

`glossaries-extra.sty v1.05: documented code`

Nicola L.C. Talbot

Dickimaw Books

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

2016-06-10

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	11
1.3 Modifications to Commands Provided by glossaries	12
1.3.1 Existence Checks	12
1.3.2 Document Definitions	14
1.3.3 Existing Glossary Style Modifications	18
1.3.4 Entry Formatting, Hyperlinks and Indexing	22
1.3.5 Entry Counting	44
1.3.6 Acronym Modifications	57
1.3.7 Indexing and Displaying Glossaries	59
1.4 Integration with glossaries-accsupp	68
1.5 Categories	79
1.6 Abbreviations	101
1.6.1 Abbreviation Styles Setup	118
1.6.2 Predefined Styles (Default Font)	121
1.6.3 Predefined Styles (Small Capitals)	133
1.6.4 Predefined Styles (Fake Small Capitals)	137
1.6.5 Predefined Styles (Emphasized)	141
1.6.6 Predefined Styles (User Parentheses Hook)	147
1.7 Using Entries in Headings	151
1.8 Multi-Lingual Support	168
2 Style Adjustments (glossaries-extra-stylemods.sty)	169
2.1 Package Initialisation	169
2.2 List-Like Styles	170
2.3 Longtable Styles	170
2.4 Long Ragged Styles	171
2.5 Supertabular Styles	173
2.6 Super Ragged Styles	174
2.7 Inline Style	175
2.8 Tree Styles	175
Glossary	187
Change History	188
Index	196

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2016/06/10 v1.05 (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 }
```

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.

```
56 \define@boolkey{glossaries-extra.sty}[@glxtr]{docdef}[true]{}
```

`indexcrossrefs` Automatically index cross references at the end of the document

```
57 \define@boolkey{glossaries-extra.sty}[@glxtr]{indexcrossrefs}[true]{%
58   \if@glxtrindexcrossrefs
59   \else
60     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
61   }
```

```
61 \fi
62 }
```

Switch off since this can increase the build time.

```
63 \@glsxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
64 \newcommand*{\@glsxtr@autoindexcrossrefs}{\@glsxtrindexcrossrefstrue}
```

iesExtraWarning

Allow users to suppress warnings.

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

raWarningNoLine

Allow users to suppress warnings.

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

```
68 \@glsxtr@declareoption{nowarn}{%
69 \let\GlossariesExtraWarning\@gobble
70 \let\GlossariesExtraWarningNoLine\@gobble
71 \glsxtr@doption{nowarn}%
72 }
```

glsxtrabbrvtype

Glossary type for abbreviations.

```
73 \newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}
```

bbreviationsdef

Set by abbreviations option.

```
74 \newcommand*{\@glsxtr@abbreviationsdef}{}%
```

bbreviationsdef

```
75 \newcommand*{\@glsxtr@doabbreviationsdef}{%
76 \@ifpackageloaded{babel}%
77 {\providecommand{\abbreviationsname}{\acronymname}}%
78 {\providecommand{\abbreviationsname}{Abbreviations}}%
79 \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
80 \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
81 \newcommand*{\printabbreviations}[1][1]{%
82 \printglossary[type=\glsxtrabbrvtype,##1]%
83 }%
84 \disable@keys{glossaries-extra.sty}{abbreviations}%
```

If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.

```
85 \ifglsacronym
86 \else
87 \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
88 \fi
89 }%
```

abbreviations If abbreviations, create a new glossary type for abbreviations.

```

90 \@glxtr@declareoption{abbreviations}{%
91   \let\@glxtr@abbreviationsdef\@glxtr@doabbreviationsdef
92 }

```

AbbreviationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses \newcommand instead of \let as a safety feature.

```

93 \newcommand*\GlsXtrDefineAbbreviationShortcuts{%
94   \newcommand*\ab{\cgl}%
95   \newcommand*\abp{\cglsp}%
96   \newcommand*\as{\glxtrshort}%
97   \newcommand*\asp{\glxtrshortpl}%
98   \newcommand*\al{\glxtrlong}%
99   \newcommand*\alp{\glxtrlongpl}%
100  \newcommand*\af{\glxtrfull}%
101  \newcommand*\afp{\glxtrfullpl}%
102  \newcommand*\Ab{\cGls}%
103  \newcommand*\Abp{\cGlspl}%
104  \newcommand*\As{\Glsxtrshort}%
105  \newcommand*\Asp{\Glsxtrshortpl}%
106  \newcommand*\Al{\Glsxtrlong}%
107  \newcommand*\Alp{\Glsxtrlongpl}%
108  \newcommand*\Af{\Glsxtrfull}%
109  \newcommand*\Afp{\Glsxtrfullpl}%
110  \newcommand*\AB{\cGLS}%
111  \newcommand*\ABP{\cGLSp}%
112  \newcommand*\AS{\GLSxtrshort}%
113  \newcommand*\ASP{\GLSxtrshortpl}%
114  \newcommand*\AL{\GLSxtrlong}%
115  \newcommand*\ALP{\GLSxtrlongpl}%
116  \newcommand*\AF{\GLSxtrfull}%
117  \newcommand*\AFP{\GLSxtrfullpl}%
118  \newcommand*\newabbr{\newabbreviation}%

```

Disable this command after it's been used.

```

119   \let\GlsXtrDefineAbbreviationShortcuts\relax
120 }

```

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

```

121 \newcommand*\GlsXtrDefineOtherShortcuts{%
122   \newcommand*\newentry{\newglossaryentry}%
123   \ifdef\printsymbols
124     {%
125       \newcommand*\newsym{\glxtrnewsymbol}%
126     }{}%
127   \ifdef\printnumbers
128     {%
129       \newcommand*\newnum{\glxtrnewnumber}%

```

```

130 }{}%
131 \let\GlsXtrDefineOtherShortcuts\relax
132 }

```

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.

```

133 \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.

```

134 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
135 {acronyms,acro,abbreviations,abbr,other,all,true,none,false}[true]{%
136   \ifcase\nr\relax % acronyms
137     \renewcommand*{\@glxtr@setupshortcuts}{%
138       \glsacrshortcutstrue
139       \DefineAcronymSynonyms
140     }%
141   \or % acro
142     \renewcommand*{\@glxtr@setupshortcuts}{%
143       \glsacrshortcutstrue
144       \DefineAcronymSynonyms
145     }%
146   \or % abbreviations
147     \renewcommand*{\@glxtr@setupshortcuts}{%
148       \GlsXtrDefineAbbreviationShortcuts
149     }%
150   \or % abbr
151     \renewcommand*{\@glxtr@setupshortcuts}{%
152       \GlsXtrDefineAbbreviationShortcuts
153     }%
154   \or % other
155     \renewcommand*{\@glxtr@setupshortcuts}{%
156       \GlsXtrDefineOtherShortcuts
157     }%
158   \or % all
159     \renewcommand*{\@glxtr@setupshortcuts}{%
160       \glsacrshortcutstrue
161       \DefineAcronymSynonyms
162       \GlsXtrDefineAbbreviationShortcuts
163       \GlsXtrDefineOtherShortcuts
164     }%
165   \or % true
166     \renewcommand*{\@glxtr@setupshortcuts}{%
167       \glsacrshortcutstrue
168       \DefineAcronymSynonyms

```



```

169      \GlsXtrDefineAbbreviationShortcuts
170      \GlsXtrDefineOtherShortcuts
171    }%
172    \else % none, false
173      \renewcommand*{\@glxtr@setupshortcuts}{}%
174    \fi
175  }

\lsxtr@doaccsupp
176 \newcommand*{\@glxtr@doaccsupp}{}

accsupp  If accsupp, load glossaries-accsupp package.
177 \@glxtr@declareoption{accsupp}{%
178   \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

GlossaryWarning  Warning text displayed in document if the external glossary file given by the argument is miss-
ing.
179 \newcommand{\glxtrNoGlossaryWarning}[1]{%
180   \@glxtr@defaultnoglossarywarning{#1}%
181 }

nomissingglsstext  If true, suppress the text produced if the external glossary file is missing.
182 \define@choicekey{glossaries-extra.sty}{nomissingglsstext}[\val\nr]%
183 {true,false}[true]{%
184   \ifcase\nr\relax % true
185     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
186       \null
187     }%
188   \else % false
189     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
190       \@glxtr@defaultnoglossarywarning{#1}%
191     }%
192   \fi
193 }

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

\lsxtr@redefstyles
194 \newcommand*{\@glxtr@redefstyles}{}

stylemods
195 \define@key{glossaries-extra.sty}{stylemods}{%
196   \ifblank{#1}%
197   {%
198     \renewcommand*{\@glxtr@redefstyles}{%
199       \RequirePackage{glossaries-extra-stylemods}}%
200   }%
201   {%

```

```

202 \renewcommand*{\@glsxtr@redefstyles}{}%
203 \@for\@glsxtr@tmp:=#1\do{%
204 \IfFileExists{glossary-\@glsxtr@tmp.sty}%
205 {%
206 \eappto\@glsxtr@redefstyles{%
207 \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
208 }%
209 {%
210 \PackageError{glossaries-extra}%
211 {Glossaries style package ‘glossary-\@glsxtr@tmp.sty’
212 doesn’t exist (did you mean to use the ‘style’ key?)}%
213 {The list of values (#1) in the ‘stylemods’ key should
214 match the glossary-xxx.sty files provided with
215 glossaries.sty}%
216 }%
217 }%
218 \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
219 }%
220 }

```

`glsxtr@do@style`

```

221 \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.

```

222 \define@key{glossaries-extra.sty}{style}{%
223 \renewcommand*{\@glsxtr@do@style}{%

```

Set this as the default style:

```

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

```

Set this style:

```

225 \setglossarystyle{#1}%
226 }%
227 }

```

Pass all other options to `glossaries`.

```

228 \DeclareOptionX*{%
229 \expandafter\glsxtr@doooption\expandafter{\CurrentOption}}

```

Process options.

```

230 \ProcessOptionsX

```

Load `glossaries` if not already loaded.

```

231 \RequirePackage{glossaries}

```

Load the `glossaries-accsupp` package if required.

```

232 \@glsxtr@doaccsupp

```

Define abbreviations `glossaries` if required.

```

233 \@glsxtr@abbreviationsdef
234 \let\@glsxtr@abbreviationsdef\relax

```

Setup shortcuts if required.

```
235 \@glstr@setupshortcuts
```

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

```
236 \renewcommand{\glstr@doption}[1]{\setupglossaries{#1}}%
```

Now define the user command:

```
237 \newcommand*\glossariesextrasetup[1]{%
238   \let\@glstr@setupshortcuts\relax
239   \setkeys{glossaries-extra.sty}{#1}%
240   \@glstr@abbreviationsdef
241   \let\@glstr@abbreviationsdef\relax
242   \@glstr@setupshortcuts
243 }
```

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

```
244 \AtBeginDocument{%
245   \disable@keys{glossaries-extra.sty}{abbreviations}%
246   \def\@glstrundeftag{\glstrundeftag}%
247 }
```

1.2 Extra Utilities

rifemptyglossary `\glstrifemptyglossary{<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.)

```
248 \newcommand{\glstrifemptyglossary}[3]{%
249   \ifglossaryexists{#1}%
250   {%
251     \ifcsstring{glolist@#1}{,}{#2}{#3}%
252   }%
253   {%
254     \glstrundefaction{Glossary type '#1' doesn't exist}{}%
255     #2%
256   }%
257 }
```

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.

```
258 \renewcommand{\glsdoifexists}[2]{%
259   \ifglentryexists{#1}{#2}%
260   {%
261     \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
262       has not been defined}{You need to define a glossary entry before
263       you can reference it.}%
264   }%
265 }
```

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

```
266 \renewcommand{\glsdoifnoexists}[2]{%
267   \ifglentryexists{#1}{%
268     \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
269       has already been defined}{}}{#2}%
270 }
```

`\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.

```
271 \ifdef\glsdoifexistsordo
272 {%
273   \renewcommand{\glsdoifexistsordo}[3]{%
274     \ifglentryexists{#1}{#2}%
275     {%
276       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
277         has not been defined}{You need to define a glossary entry
278         before you can use it.}%
279       #3%
280     }%
281   }%
282 }
283 {%
284   \glstr@warnonexistsordo\glsdoifexistsordo
285   \newcommand{\glsdoifexistsordo}[3]{%
286     \ifglentryexists{#1}{#2}%
287     {%
288       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
289         has not been defined}{You need to define a glossary entry
290         before you can use it.}%
291       #3%
292     }%
293   }%
294 }
```

```

293 }%
294 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

295 \ifdef\doifglossarynoexistsordo
296 {%
297   \renewcommand{\doifglossarynoexistsordo}[3]{%
298     \ifglossaryexists{#1}%
299     {%
300       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
301       #3%
302     }%
303     {#2}%
304   }%
305 }
306 {%
307   \glstr@warnonexistsordo\doifglossarynoexistsordo
308   \newcommand{\doifglossarynoexistsordo}[3]{%
309     \ifglossaryexists{#1}%
310     {%
311       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
312       #3%
313     }%
314     {#2}%
315   }%
316 }
317

```

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

```

318 \appto\@newglossaryentryposthook{%
319   \ifdefvoid\@glo@see
320   {\csxdef{glo@\@glo@label @see}{}}%
321   {%
322     \csxdef{glo@\@glo@label @see}{\@glo@see}%
323     \@glstr@autoindexcrossrefs
324   }%
325 }
326 \appto\@gls@keymap{,{see}{see}}

```

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

```

327 \AtEndDocument{\if@glstrindexcrossrefs\glstraddallcrossrefs\fi}

```

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

```

328 \newcommand*{\glstraddallcrossrefs}{%
329   \forallglossaries{\@glo@type}%
330   {%
331     \forglsentries[\@glo@type]{\@glo@label}%
332     {%

```

```

333     \ifglsused{\@glo@label}{\@glsxtr@addunusedxrefs{\@glo@label}}{}%
334   }%
335 }%
336 }

```

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

```

337 \newcommand*{\@glsxtr@addunusedxrefs}[1]{%
338   \letcs{\@glo@see}{glo@glsdetoklabel{#1}@see}%
339   \ifdefvoid\@glo@see
340   {%
341   {%
342     \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
343   }%
344 }

```

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

```

345 \newcommand*{\glsxtr@addunused}[1][]{%
346   \@glsxtr@addunused
347 }

```

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

```

348 \def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
349   \@for\@glsxtr@label:=#1\do
350   {%
351     \ifglsused{\@glsxtr@label}}{}%
352   {%
353     \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
354     \glsunset{\@glsxtr@label}%
355     \@glsxtr@addunusedxrefs{\@glsxtr@label}%
356   }%
357 }%
358 }

```

xtrunusedformat

```

359 \newcommand*{\glsxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

noidxglossaries Modify \makenoidxglossaries so that it automatically switches off and disables the docdef key.

```

360 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
361 \renewcommand{\makenoidxglossaries}{%
362   \glsxtr@orgmakenoidxglossaries
363   \@glsxtrdocdeffalse
364   \disable@keys{glossaries-extra.sty}{docdef}%
365 }

```

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

```
366 \renewcommand*{\gls@defdocnewglossaryentry}{%
367   \if@glxtrdocdef
```

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

```
368   \let\gls@checkseeallowed\relax
369   \let\newglossaryentry\new@glossaryentry
370 \else
371   \renewcommand*{\newglossaryentry}[2]{%
372     \PackageError{glossaries-extra}{Glossary entries must
373       be \MessageBreak defined in the preamble with \MessageBreak
374       package option 'docdef=false'}{Move your glossary definitions to
375       the preamble. You can also put them in a \MessageBreak separate file
376       and load them with \string\loadglsentries.}%
377   }%
378 \fi
379 }%
```

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

```
380 \newcommand*{\GlsXtrEnableOnTheFly}{%
381   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
382 }
```

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.)

```
383 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
384   \renewcommand*{\glsdetoklabel}[1]{%
385     \expandafter\@glxtr@ifcsstart\string##1 \@glxtr@end@
386     {%
387       \expandafter\detokenize\expandafter{##1}%
388     }%
389     {\detokenize{##1}}}%
390   }%
391   \@GlsXtrEnableOnTheFly
392 }
393 \def\@glxtr@ifcsstart#1#2\@glxtr@end@#3#4{%
394   \expandafter\if\glsbackslash#1%
395     #3%
396   \else
397     #4%
398   \fi
399 }
```

sxtrstarflywarn

```
400 \newcommand*{\glsxtrstarflywarn}{%
401   \GlossariesExtraWarning{Experimental starred version of
402   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
403   read the warnings in the glossaries-extra user manual)}}%
404 }
```

rEnableOnTheFly

```
405 \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:

`\glsxtrcat`

```
406 \newcommand*{\glsxtrcat}{general}
```

`\glsxtr`

```
407 \newcommand*{\glsxtr}[1] [] {%
408   \def\glsxtr@keylist{##1}%
409   \@glsxtr
410 }
```

`\@glsxtr`

```
411 \newcommand*{\@glsxtr}[2] [] {%
412   \ifglstryexists{##2}%
413   {%
414     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
415   }%
416   {%
417     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
418     description={\nopostdesc},##1}%
419   }%
420   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
421 }
```

`\Glsxtr`

```
422 \newcommand*{\Glsxtr}[1] [] {%
423   \def\glsxtr@keylist{##1}%
424   \@Glsxtr
425 }
```

`\@Glsxtr`

```
426 \newcommand*{\@Glsxtr}[2] [] {%
427   \ifglstryexists{##2}%
428   {%
429     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
430   }%
```



```

431  {%
432    \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
433      description={\nopostdesc},##1}%
434  }%
435  \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
436  }

```

\glsxtrpl

```

437  \newcommand*{\glsxtrpl}[1][]{%
438    \def\glsxtr@keylist{##1}%
439    \@glsxtrpl
440  }

```

\@glsxtrpl

```

441  \newcommand*{\@glsxtrpl}[2][]{%
442    \ifglsentryexists{##2}%
443    {%
444      \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
445    }%
446    {%
447      \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
448        description={\nopostdesc},##1}%
449    }%
450    \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
451  }

```

\Glsxtrpl

```

452  \newcommand*{\Glsxtrpl}[1][]{%
453    \def\glsxtr@keylist{##1}%
454    \@Glsxtrpl
455  }

```

\@Glsxtrpl

```

456  \newcommand*{\@Glsxtrpl}[2][]{%
457    \ifglsentryexists{##2}
458    {%
459      \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
460    }%
461    {%
462      \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
463        description={\nopostdesc},##1}%
464    }%
465    \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
466  }

```

\GlsXtrWarning

```

467  \newcommand*{\GlsXtrWarning}[2]{%
468    \def\@glsxtr@optlist{##1}%
469    \@onelevel@sanitize\@glsxtr@optlist

```

```

470 \GlossariesExtraWarning{The options ‘\@glsxtr@optlist’ have
471 been ignored for entry ‘##2’ as it has already been defined}%
472 }

```

Disable commands after the glossary:

```

473 \let\@glsxtr@orgprintglossary\@printglossary
474 \renewcommand\@printglossary[2]{%
475 \@glsxtr@orgprintglossary{##1}{##2}%
476 \def\@glsxtr{\@glsxtr@disabledflycommand\glsxtr}%
477 \def\@glsxtrpl{\@glsxtr@disabledflycommand\glsxtrpl}%
478 \def\@Glsxtr{\@glsxtr@disabledflycommand\Glsxtr}%
479 \def\@Glsxtrpl{\@glsxtr@disabledflycommand\Glsxtrpl}%
480 }

```

abledflycommand

```

481 \newcommand*{\@glsxtr@disabledflycommand}[1]{%
482 \PackageError{glossaries-extra}%
483 {\string##1\space can’t be used after any of the \MessageBreak
484 glossaries have been displayed}%
485 {The on-the-fly commands enabled by
486 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
487 before the glossaries. If you want to use any entries \MessageBreak
488 after any of the glossaries, you must use the standard \MessageBreak
489 method of first defining the entry and then using the \MessageBreak
490 entry with commands like \string\gls}%
491 \@glsxtr@disabledflycommand
492 }%
493 \newcommand*{\@glsxtr@disabledflycommand}[2][{}]{##2}

```

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

```

494 \let\GlsXtrEnableOnTheFly\relax
495 }
496 \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.

```

497 \newcommand*{\@glsxtr@current@style}{\@glossary@default@style}

```

Modify \setglossarystyle to set the above.

etglossarystyle

```

498 \renewcommand*{\setglossarystyle}[1]{%
499 \ifcsundef{@glsstyle@#1}%
500 {%
501 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%

```

```

502 }%
503 {%
504   \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

505   \protected@edef\@glstr@current@style{#1}%
506 }%
507 \ifx\@glossary@default@style\relax
508   \protected@edef\@glossary@default@style{#1}%
509 \fi
510 }

```

In case we have an old version of glossaries:

```

511 \ifdef\@glossary@default@style
512 {}
513 {%
514   \let\@glossary@default@style\relax
515 }

```

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

```

516 \ifdef\glslistdottedwidth
517 {%
518   \ifdim\glslistdottedwidth=.5\hsize
519     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
520     \AtBeginDocument{%
521       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
522         \setlength{\glslistdottedwidth}{.5\columnwidth}%
523       \fi
524     }%
525 \fi
526 }
527 {}%

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

528 \ifdef\glsdescwidth
529 {%
530   \ifdim\glsdescwidth=.6\hsize
531     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
532     \AtBeginDocument{%
533       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
534         \setlength{\glsdescwidth}{.6\columnwidth}%
535       \fi
536     }%
537 \fi
538 }
539 {}%

```

and for \glspagelistwidth:

lspagelistwidth

```
540 \ifdef\glspagelistwidth
541 {%
542   \ifdim\glspagelistwidth=.1\hsize
543     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
544     \AtBeginDocument{%
545       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
546         \setlength{\glspagelistwidth}{.1\columnwidth}%
547       \fi
548     }%
549   \fi
550 }
551 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```
552 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
553 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
554   \glsnonumberlistfalse
555   \renewcommand*{\glossaryentrynumbers}[1]{%
556     \ifglentryexists{\glscurrententrylabel}%
557     {%
558       \@glsxtrpreloctag
559       \GlsXtrFormatLocationList{#1}%
560       \@glsxtrpostloctag
561       \gls@save@numberlist{#1}%
562     }{}%
563   }%
564 \else
565   \glsnonumberlisttrue
566   \renewcommand*{\glossaryentrynumbers}[1]{%
567     \ifglentryexists{\glscurrententrylabel}%
568     {%
569       \gls@save@numberlist{#1}%
570     }{}%
571   }%
572 \fi
```

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

```
573 \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

```
574 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
575   \let\@glxtrpreloctag\@glxtrpreloctag
576   \let\@glxtrpostloctag\@glxtrpostloctag
577   \renewcommand*{\@glxtr@pagetag}{#1}%
578   \renewcommand*{\@glxtr@pagetag}{#2}%
579   \renewcommand*{\@glxtr@savepreloctag}[2]{%
580     \csgdef{\@glxtr@preloctag@##1}{##2}%
581   }%
582   \renewcommand*{\@glxtr@doloctag}{%
583     \ifcsundef{\@glxtr@preloctag\@glscurrententrylabel}%
584     {%
585       \GlossariesWarning{Missing pre-location tag for ‘\@glscurrententrylabel’.
586         Rerun required}%
587     }%
588     {%
589       \csuse{\@glxtr@preloctag\@glscurrententrylabel}%
590     }%
591   }%
592 }
593 \@onlypreamble\GlsXtrEnablePreLocationTag
```

glxtrpreloctag

```
594 \newcommand*{\@glxtrpreloctag}{%
595   \let\@glxtr@org@delimN\delimN
596   \let\@glxtr@org@delimR\delimR
597   \let\@glxtr@org@glsgignore\glsgignore
   \gdef is required as the delimiters may occur inside a scope.
598   \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
599   \renewcommand*{\delimN}{%
600     \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
601     \@glxtr@org@delimN}%
602   \renewcommand*{\delimR}{%
603     \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
604     \@glxtr@org@delimR}%
605   \renewcommand*{\glsgignore}[1]{%
606     \gdef\@glxtr@thisloctag{\relax}%
607     \@glxtr@org@glsgignore{##1}}%
608   \@glxtr@doloctag
609 }
```

glxtrpreloctag

```
610 \newcommand*{\@glxtrpreloctag}{}
```

@glxtr@pagetag

```
611 \newcommand*{\@glxtr@pagetag}{}
```

glxtr@pagetag

```
612 \newcommand*{\@glxtr@pagetag}{}
```

lsxtrpostloctag

```
613 \newcommand*{\@@glxtrpostloctag}{%
614   \let\delimN\@glxtr@org@delimN
615   \let\delimR\@glxtr@org@delimR
616   \let\glignore\@glxtr@org@glignore
617   \protected@write\@auxout{}{%
618     {\string\@glxtr@savepreloctag{\glscurrententrylabel}{\@glxtr@thisloctag}}%
619 }
```

lsxtrpostloctag

```
620 \newcommand*{\@glxtrpostloctag}{}
```

lsxtr@preloctag

```
621 \newcommand*{\@glxtr@savepreloctag}[2]{%
622 \protected@write\@auxout{}{%
623   \string\providecommand\string\@glxtr@savepreloctag[2]{}}
```

glxtr@doloctag

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

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

```
625 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
626   \XKV@plfalse
627   \XKV@sttrue
628   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
629   {%
630     \csname glsnonumberlist\XKV@resa\endcsname
631     \ifglsnonumberlist
632       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
633     \else
634       \def\glossaryentrynumbers##1{%
635         \@glxtrpreloctag
636         \GlsXtrFormatLocationList{##1}%
637         \@glxtrpostloctag
638         \gls@save@numberlist{##1}}%
639     \fi
640   }%
641 }
```

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.

```

642 \renewcommand*{\glentryfmt}{%
643   \ifglshasshort{\glslabel}{\glsetabbrvfmt{\glscategory{\glslabel}}}{}%
644   \glrifregular{\glslabel}%
645   {\glxtrregularfont{\glsgenentryfmt}}}%
646   {%
647     \ifglshasshort{\glslabel}%
648     {\glxtrgenabbrvfmt}%
649     {\glxtrregularfont{\glsgenentryfmt}}}%
650   }%
651 }

```

sxtrregularfont Font used for regular entries.

```

652 \newcommand*{\glxtrregularfont}[1]{#1}

```

Commands like `\glrifplural` 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, `\glfirst` or `\glsplural`. This can provide better consistency with the formatting of the `\gls`-like commands, even though they don't use `\glentryfmt`.

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

```

653 \renewcommand{\@gls@field@link}[4][]{%
654   \glsdoidexists{#3}%
655   {%
656     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
657     \def\glscustomtext{#4}%
658     \@glxtr@field@linkdefs
659     #1%
660     \@gls@link[#2]{#3}{#4}%
661   }%
662   \glspostlinkhook
663 }

```

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

```

664 \newcommand*{\@glxtr@field@linkdefs}{%
665   \let\glxtrifwasfirstuse\@secondoftwo
666   \let\glrifplural\@secondoftwo
667   \let\glscapscase\@firstofthree
668   \let\glinsert\@empty
669 }

```

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

assignfieldfont

```

670 \newcommand*{\glxtrassignfieldfont}[1]{%
671   \ifglshasshort{#1}%

```

```

672 {%
673   \glssetabbrvfmt{\glscategory{#1}}%
674   \glsifregular{#1}%
675   {\let\@gls@field@font\glsxtrregularfont}%
676   {\let\@gls@field@font\@firstofone}%
677 }%
678 {%
679   \glsifnotregular{#1}%
680   {\let\@gls@field@font\@firstofone}%
681   {\let\@gls@field@font\glsxtrregularfont}%
682 }%
683 }

```

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

```

684 \def\@glstext@#1#2[#3]{%
685   \glsxtrassignfieldfont{#2}%
686   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
687 }

```

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

```

688 \def\@GLStext@#1#2[#3]{%
689   \glsxtrassignfieldfont{#2}%
690   \@gls@field@link[\let\glsacscase\@thirdofthree]{#1}{#2}%
691   {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
692 }

```

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

```

693 \def\@Glstext@#1#2[#3]{%
694   \glsxtrassignfieldfont{#2}%
695   \@gls@field@link[\let\glsacscase\@secondofthree]{#1}{#2}%
696   {\@gls@field@font{\Glsaccesstext{#2}#3}}%
697 }

```

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

```

698 \def\@glsfirst@#1#2[#3]{%
699   \glsxtrassignfieldfont{#2}%
700   \@gls@field@link[\let\glsxtrifwasfirstuse\@firstoftwo]{#1}{#2}%
701   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
702 }

```

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

```

703 \def\@Glsfirst@#1#2[#3]{%
704   \glsxtrassignfieldfont{#2}%
705   \@gls@field@link
706   [\let\glsxtrifwasfirstuse\@firstoftwo
707   \let\glsacscase\@secondofthree
708   ]%
709   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
710 }

```


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

```
711 \def\@GLSfirst@#1#2[#3]{%
712   \glstrassignfieldfont{#2}%
713   \@gls@field@link
714   [\let\glstrifwasfirstuse\@firstoftwo
715   \let\glscapscase\@thirdofthree
716   ]%
717   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
718 }
```

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

```
719 \def\@glsplural@#1#2[#3]{%
720   \glstrassignfieldfont{#2}%
721   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
722   {\@gls@field@font{\glsaccessplural{#2}#3}}%
723 }
```

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

```
724 \def\@Glsplural@#1#2[#3]{%
725   \glstrassignfieldfont{#2}%
726   \@gls@field@link
727   [\let\glsifplural\@firstoftwo
728   \let\glscapscase\@secondofthree
729   ]%
730   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
731 }
```

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

```
732 \def\@GLSplural@#1#2[#3]{%
733   \glstrassignfieldfont{#2}%
734   \@gls@field@link
735   [\let\glsifplural\@firstoftwo
736   \let\glscapscase\@thirdofthree
737   ]%
738   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
739 }
```

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

```
740 \def\glsfirstplural@#1#2[#3]{%
741   \glstrassignfieldfont{#2}%
742   \@gls@field@link
743   [\let\glstrifwasfirstuse\@firstoftwo
744   \let\glsifplural\@firstoftwo
745   ]%
746   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
747 }
```

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

```

748 \def\@Glsfirstplural@#1#2[#3]{%
749   \glstrassignfieldfont{#2}%
750   \@gls@field@link
751   [\let\glstrifwasfirstuse\@firstoftwo
752    \let\glsifplural\@firstoftwo
753    \let\glscapscase\@secondofthree
754   ]%
755   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
756 }

```

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

```

757 \def\@GLSfirstplural@#1#2[#3]{%
758   \glstrassignfieldfont{#2}%
759   \@gls@field@link
760   [\let\glstrifwasfirstuse\@firstoftwo
761    \let\glsifplural\@firstoftwo
762    \let\glscapscase\@thirdofthree
763   ]%
764   {#1}{#2}%
765   {\@gls@field@font{\@GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
766 }

```

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

```

767 \def\@glsname@#1#2[#3]{%
768   \glstrassignfieldfont{#2}%
769   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
770 }

```

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

```

771 \def\@Glsname@#1#2[#3]{%
772   \glstrassignfieldfont{#2}%
773   \@gls@field@link
774   [\let\glscapscase\@secondoftwo]{#1}{#2}%
775   {\@gls@field@font{\Glsaccessname{#2}#3}}%
776 }

```

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

```

777 \def\@GLSname@#1#2[#3]{%
778   \glstrassignfieldfont{#2}%
779   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
780   {#1}{#2}%
781   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
782 }

```

\@glsdesc@

```

783 \def\@glsdesc@#1#2[#3]{%
784   \glstrassignfieldfont{#2}%
785   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
786 }

```



```

\@Glssymbol@ First letter uppercase version.
825 \def\@Glssymbol@#1#2[#3]{%
826   \glstrassignfieldfont{#2}%
827   \@gls@field@link
828   [\let\glscapscase\@secondoftwo]%
829   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
830 }

\@GLSsymbol@ All uppercase version.
831 \def\@GLSsymbol@#1#2[#3]{%
832   \glstrassignfieldfont{#2}%
833   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
834   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
835 }

lssymbolplural@ No case-changing version.
836 \def\@lssymbolplural@#1#2[#3]{%
837   \glstrassignfieldfont{#2}%
838   \@gls@field@link
839   [\let\glscapscase\@secondoftwo
840   \let\glsifplural\@firstoftwo
841   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
842 }

lssymbolplural@ First letter uppercase version.
843 \def\@lssymbolplural@#1#2[#3]{%
844   \glstrassignfieldfont{#2}%
845   \@gls@field@link
846   [\let\glscapscase\@secondoftwo
847   \let\glsifplural\@firstoftwo
848   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
849 }

LSsymbolplural@ All uppercase version.
850 \def\@LSsymbol@#1#2[#3]{%
851   \glstrassignfieldfont{#2}%
852   \@gls@field@link
853   [\let\glscapscase\@thirdoftwo
854   \let\glsifplural\@firstoftwo
855   ]%
856   {#1}{#2}%
857   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
858 }

\@Glsuseri@ First letter uppercase version.
859 \def\@Glsuseri@#1#2[#3]{%
860   \glstrassignfieldfont{#2}%
861   \@gls@field@link

```

```

862 [\let\glscapscase\@secondoftwo]{#1}{#2}%
863 {\@gls@field@font{\Glsentryuseri{#2}#3}}%
864 }

```

\@GLSuseri@ All uppercase version.

```

865 \def\@GLSuseri@#1#2[#3]{%
866 \glstrassignfieldfont{#2}%
867 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
868 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
869 }

```

\@Glsuserii@ First letter uppercase version.

```

870 \def\@Glsuserii@#1#2[#3]{%
871 \glstrassignfieldfont{#2}%
872 \@gls@field@link
873 [\let\glscapscase\@secondoftwo]%
874 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
875 }

```

\@GLSuserii@ All uppercase version.

```

876 \def\@GLSuserii@#1#2[#3]{%
877 \glstrassignfieldfont{#2}%
878 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
879 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
880 }

```

\@Glsuseriii@ First letter uppercase version.

```

881 \def\@Glsuseriii@#1#2[#3]{%
882 \glstrassignfieldfont{#2}%
883 \@gls@field@link
884 [\let\glscapscase\@secondoftwo]%
885 {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
886 }

```

\@GLSuseriii@ All uppercase version.

```

887 \def\@GLSuseriii@#1#2[#3]{%
888 \glstrassignfieldfont{#2}%
889 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
890 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
891 }

```

\@Glsuseriv@ First letter uppercase version.

```

892 \def\@Glsuseriv@#1#2[#3]{%
893 \glstrassignfieldfont{#2}%
894 \@gls@field@link
895 [\let\glscapscase\@secondoftwo]%
896 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
897 }

```

\@GLSuseriv@ All uppercase version.

```
898 \def\@GLSuseriv@#1#2[#3]{%
899   \glstrassignfieldfont{#2}%
900   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
901   {#1}{#2}%
902   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
903 }
```

\@Glsuserv@ First letter uppercase version.

```
904 \def\@Glsuserv@#1#2[#3]{%
905   \glstrassignfieldfont{#2}%
906   \@gls@field@link
907   [\let\glscapscase\@secondoftwo]%
908   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}}%
909 }
```

\@GLSuserv@ All uppercase version.

```
910 \def\@GLSuserv@#1#2[#3]{%
911   \glstrassignfieldfont{#2}%
912   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
913   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
914 }
```

\@Glsuservi@ First letter uppercase version.

```
915 \def\@Glsuservi@#1#2[#3]{%
916   \glstrassignfieldfont{#2}%
917   \@gls@field@link
918   [\let\glscapscase\@secondoftwo]%
919   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
920 }
```

\@GLSuservi@ All uppercase version.

```
921 \def\@GLSuservi@#1#2[#3]{%
922   \glstrassignfieldfont{#2}%
923   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
924   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
925 }
```

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

\@acrshort No case change.

```
926 \def\@acrshort#1#2[#3]{%
927   \glsdoifexists{#2}%
928   {%
929     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
930     \let\glxtrifwasfirstuse\@secondoftwo
931     \let\glsifplural\@secondoftwo
```

```

932 \let\glscapscase\@firstofthree
933 \let\glsinsert\@empty
934 \def\glscustomtext{%
935     \acronymfont{\glsaccesssshort{#2}}#3%
936 }%
937 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
938 }%
939 \glspostlinkhook
940 }

```

\@Acrshort First letter uppercase.

```

941 \def\@Acrshort#1#2[#3]{%
942     \glsdoifexists{#2}%
943     {%
944         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
945         \let\glstrifwasfirstuse\@secondoftwo
946         \let\glsifplural\@secondoftwo
947         \let\glscapscase\@secondofthree
948         \let\glsinsert\@empty
949         \def\glscustomtext{%
950             \acronymfont{\Glsaccesssshort{#2}}#3%
951         }%
952         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
953     }%
954     \glspostlinkhook
955 }

```

\@ACRshort All uppercase.

```

956 \def\@ACRshort#1#2[#3]{%
957     \glsdoifexists{#2}%
958     {%
959         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
960         \let\glstrifwasfirstuse\@secondoftwo
961         \let\glsifplural\@secondoftwo
962         \let\glscapscase\@thirdofthree
963         \let\glsinsert\@empty
964         \def\glscustomtext{%
965             \mfirstucMakeUppercase{\acronymfont{\glsaccesssshort{#2}}#3}%
966         }%
967         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
968     }%
969     \glspostlinkhook
970 }

```

\@acrshortpl No case change.

```

971 \def\@acrshortpl#1#2[#3]{%
972     \glsdoifexists{#2}%
973     {%
974         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

975 \let\glxtrifwasfirstuse\@secondoftwo
976 \let\glsifplural\@firstoftwo
977 \let\glscapscase\@firstofthree
978 \let\glsinsert\@empty
979 \def\glscustomtext{%
980 \acronymfont{\glsaccessshortpl{#2}}#3%
981 }%
982 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
983 }%
984 \glspostlinkhook
985 }

```

\@Acrshortpl First letter uppercase.

```

986 \def\@Acrshortpl#1#2[#3]{%
987 \glsdoifexists{#2}%
988 {%
989 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
990 \let\glxtrifwasfirstuse\@secondoftwo
991 \let\glsifplural\@firstoftwo
992 \let\glscapscase\@secondofthree
993 \let\glsinsert\@empty
994 \def\glscustomtext{%
995 \acronymfont{\Glsaccessshortpl{#2}}#3%
996 }%
997 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
998 }%
999 \glspostlinkhook
1000 }

```

\@ACRshortpl All uppercase.

```

1001 \def\@ACRshortpl#1#2[#3]{%
1002 \glsdoifexists{#2}%
1003 {%
1004 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1005 \let\glxtrifwasfirstuse\@secondoftwo
1006 \let\glsifplural\@firstoftwo
1007 \let\glscapscase\@thirdofthree
1008 \let\glsinsert\@empty
1009 \def\glscustomtext{%
1010 \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
1011 }%
1012 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1013 }%
1014 \glspostlinkhook
1015 }

```

\@acrlong No case change.

```

1016 \def\@acrlong#1#2[#3]{%
1017 \glsdoifexists{#2}%

```



```

1018 {%
1019   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1020   \let\glxtrifwasfirstuse\@secondoftwo
1021   \let\gl@ifplural\@secondoftwo
1022   \let\gl@scapscase\@firstofthree
1023   \let\gl@insert\@empty
1024   \def\gl@customtext{%
1025     \acronymfont{\gl@saccesslong{#2}}#3%
1026   }%
1027   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1028 }%
1029 \glspostlinkhook
1030 }

```

\@Acrlong First letter uppercase.

```

1031 \def\@Acrlong#1#2[#3]{%
1032   \gl@doifexists{#2}%
1033   {%
1034     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1035     \let\glxtrifwasfirstuse\@secondoftwo
1036     \let\gl@ifplural\@secondoftwo
1037     \let\gl@scapscase\@secondofthree
1038     \let\gl@insert\@empty
1039     \def\gl@customtext{%
1040       \acronymfont{\Glsaccesslong{#2}}#3%
1041     }%
1042     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1043   }%
1044   \glspostlinkhook
1045 }

```

\@ACRlong All uppercase.

```

1046 \def\@ACRlong#1#2[#3]{%
1047   \gl@doifexists{#2}%
1048   {%
1049     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1050     \let\glxtrifwasfirstuse\@secondoftwo
1051     \let\gl@ifplural\@secondoftwo
1052     \let\gl@scapscase\@thirdofthree
1053     \let\gl@insert\@empty
1054     \def\gl@customtext{%
1055       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslong{#2}}#3}%
1056     }%
1057     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1058   }%
1059   \glspostlinkhook
1060 }

```

\@acrlongpl No case change.

```

1061 \def\@acrlongpl#1#2[#3]{%
1062   \glsdoifexists{#2}%
1063   {%
1064     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1065     \let\glxtrifwasfirstuse\@secondoftwo
1066     \let\gl@sifplural\@firstoftwo
1067     \let\glscapscase\@firstofthree
1068     \let\gl$insert\@empty
1069     \def\glscustomtext{%
1070       \acronymfont{\gl@saccesslongpl{#2}}#3%
1071     }%
1072     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1073   }%
1074   \glspostlinkhook
1075 }

```

\@Acrlongpl First letter uppercase.

```

1076 \def\@Acrlongpl#1#2[#3]{%
1077   \glsdoifexists{#2}%
1078   {%
1079     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1080     \let\glxtrifwasfirstuse\@secondoftwo
1081     \let\gl@sifplural\@firstoftwo
1082     \let\glscapscase\@secondofthree
1083     \let\gl$insert\@empty
1084     \def\glscustomtext{%
1085       \acronymfont{\Glsaccesslongpl{#2}}#3%
1086     }%
1087     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1088   }%
1089   \glspostlinkhook
1090 }

```

\@ACrlongpl All uppercase.

```

1091 \def\@ACrlongpl#1#2[#3]{%
1092   \glsdoifexists{#2}%
1093   {%
1094     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1095     \let\glxtrifwasfirstuse\@secondoftwo
1096     \let\gl@sifplural\@firstoftwo
1097     \let\glscapscase\@thirdofthree
1098     \let\gl$insert\@empty
1099     \def\glscustomtext{%
1100       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslongpl{#2}}#3}%
1101     }%
1102     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1103   }%
1104   \glspostlinkhook
1105 }

```

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

\@glsaddkey

```

1106 \renewcommand*{\@glsaddkey}[7]{%
1107   \key@ifundefined{glossentry}{#1}%
1108   {%
1109     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1110     \appto\@gls@keymap{,{#1}{#1}}%
1111     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
1112     \appto\@newglossaryentryposthook{%
1113       \letcs{\@glo@tmp}{@glo@#1}%
1114       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1115     }%
1116     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1117     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1118   \ifcsdef{@gls@user@#1@}%
1119   {%
1120     \PackageError{glossaries}%
1121     {Can't define '\string#5' as helper command
1122     '\expandafter\string\csname @gls@user@#1@endcsname' already
1123     exists}%
1124     {}%
1125   }%
1126   {%
1127     \expandafter\newcommand\expandafter*\expandafter
1128     {\csname @gls@user@#1@endcsname}[2][ ]{%
1129       \new@ifnextchar[%
1130         {\csuse{@gls@user@#1@}{##1}{##2}}%
1131         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1132     \csdef{@gls@user@#1@}##1##2[##3]{%
1133       \@gls@field@link{##1}{##2}{#3{##2}##3}%
1134     }%
1135     \newrobustcmd*{#5}{%
1136       \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
1137   }%

```

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

```

1138   \ifcsdef{@Gls@user@#1@}%
1139   {%
1140     \PackageError{glossaries}%
1141     {Can't define '\string#6' as helper command
1142     '\expandafter\string\csname @Gls@user@#1@endcsname' already
1143     exists}%
1144     {}%
1145   }%
1146   {%
1147     \expandafter\newcommand\expandafter*\expandafter

```

```

1148     {\csname @Gls@user@#1\endcsname}[2] [] {%
1149     \new@ifnextchar[%
1150     {\csuse{@Gls@user@#1@}{##1}{##2}}}%
1151     {\csuse{@Gls@user@#1@}{##1}{##2} [] }}%
1152     \csdef{@Gls@user@#1@}##1##2[##3]{%
1153     \@gls@field@link[\let\glscapscase\secondofthree]%
1154     {##1}{##2}{#4{##2}##3}%
1155     }%
1156     \newrobustcmd*{#6}{%
1157     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
1158     }%

```

Finally the all caps version (modified):

```

1159     \ifcsdef{@GLS@user@#1@}%
1160     {%
1161     \PackageError{glossaries}%
1162     {Can't define '\string#7' as helper command
1163     '\expandafter\string\csname @GLS@user@#1\endcsname' already
1164     exists}%
1165     }%
1166     }%
1167     {%
1168     \expandafter\newcommand\expandafter*\expandafter
1169     {\csname @GLS@user@#1\endcsname}[2] [] {%
1170     \new@ifnextchar[%
1171     {\csuse{@GLS@user@#1@}{##1}{##2}}}%
1172     {\csuse{@GLS@user@#1@}{##1}{##2} [] }}%
1173     \csdef{@GLS@user@#1@}##1##2[##3]{%
1174     \@gls@field@link[\let\glscapscase@thirdofthree]%
1175     {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
1176     }%
1177     \newrobustcmd*{#7}{%
1178     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1179     }%
1180     }%
1181     {%
1182     \PackageError{glossaries-extra}{Key '#1' already exists}{}%
1183     }%
1184 }

```

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

```

1185 \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.

```

1186 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
1187 \renewcommand*{\@gls@link@checkfirsthyper}{%

```

\ifglused 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.

```

1188 \ifglused{\glslabel}%
1189 {\let\glstrifwasfirstuse\@secondoftwo}
1190 {\let\glstrifwasfirstuse\@firstoftwo}%

Store the category label for convenience.

1191 \edef\glscategorylabel{\glscategory{\glslabel}}%
1192 \ifglused{\glslabel}%
1193 {%
1194   \glscategoryattribute{\glscategorylabel}{nohypernext}{true}%
1195   {\KV@glslink@hyperfalse}{}}%
1196 }%
1197 {%
1198   \glscategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
1199   {\KV@glslink@hyperfalse}{}}%
1200 }%
1201 \glslinkcheckfirsthyperhook
1202 }

```

`\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.

```

1203 \ifdefined\do@glsglsdisablehyperinlist
1204 {%
1205   \let\@glstr@do@glsglsdisablehyperinlist\do@glsglsdisablehyperinlist
1206   \renewcommand*{\do@glsglsdisablehyperinlist}{%
1207     \@glstr@do@glsglsdisablehyperinlist
1208     \glscategoryattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}}%
1209   }
1210 }
1211 {}

```

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

```

1212 \define@boolkey{glslink}{noindex}[true]{}
1213 \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`

```

1214 \ifdefined\@gls@setdefault@glslink@opts
1215 {
1216   \renewcommand*{\@gls@setdefault@glslink@opts}{%
1217     \KV@glslink@noindexfalse
1218   }
1219 }
1220 {

```

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

```

1221 \newcommand*{\@gls@setdefault@glslink@opts}{%

```

```

1222 \KV@glslink@noindexfalse
1223 }
1224 \pretodo@glsl@disablehyperinlist{\@glsl@setdefault@glslink@opts}
1225 }

```

DefaultGlsOpts Set the default options for `\glslink` etc.

```

1226 \newcommand*\GlsXtrSetDefaultGlsOpts}[1]{%
1227 \renewcommand*\@glsl@setdefault@glslink@opts{\setkeys{glslink}{#1}}%
1228 }

```

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

```

1229 \newcommand*\glstrifindexing}[2]{%
1230 \ifKV@glslink@noindex #2\else #1\fi
1231 }

```

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

```

1232 \renewcommand*\glswriteentry}[2]{%
1233 \glstrifindexing
1234 {%
1235 \ifglslindexonlyfirst
1236 \ifglslused{#1}
1237 {\glstrdoautoindexname{#1}{dualindex}}%
1238 {#2}%
1239 \else
1240 \glslifattribute{#1}{indexonlyfirst}{true}%
1241 {\ifglslused{#1}
1242 {\glstrdoautoindexname{#1}{dualindex}}%
1243 {#2}}%
1244 {#2}%
1245 \fi
1246 }%
1247 {}%
1248 }

```

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

```

1249 \appto@do@wrglossary{\@glstrdo@wrindex
1250 \glstrdowrglossaryhook{\@glsl@label}%
1251 }

```

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

Similarly for the “`noidx`” version:

s@noidxglossary

```

1252 \appto\glsl@s@noidxglossary{\@glstrdo@wrindex
1253 \glstrdowrglossaryhook{\@glsl@label}%
1254 }

```

xtr@do@@wrindex

```
1255 \newcommand*{\@glsxtr@do@@wrindex}{%
1256   \glsxtrdoautoindexname{\@gls@label}{dualindex}%
1257 }
```

owrglossaryhook Allow user to hook into indexing code. (Always used by \glsadd. Used by \gls when indexing, which may or may not occur depending on the indexing settings.)

```
1258 \newcommand*{\glsxtrdowrglossaryhook}[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.

```
1259 \newcommand*{\@gls@alt@hyp@opt}[1]{%
1260   \let\glslinkvar\@firstofthree
1261   \let\@gls@hyp@opt@cs#1\relax
1262   \@ifstar{\s@gls@hyp@opt}%
1263   {\@ifnextchar+%
1264     {\@firstoftwo{\p@gls@hyp@opt}}%
1265     {%
1266       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
1267       {\@firstoftwo{\@alt@gls@hyp@opt}}%
1268       {#1}%
1269     }%
1270   }%
1271 }
```

alt@gls@hyp@opt User version

```
1272 \newcommand*{\@alt@gls@hyp@opt}[1] []{%
1273   \let\glslinkvar\@firstofthree
1274   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
1275 \newcommand*{\@gls@alt@hyp@opt@char}{}
```

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

```
1276 \newcommand*{\@gls@alt@hyp@opt@keys}{}
```

rSetAltModifier

```
1277 \newcommand*{\GlsXtrSetAltModifier}[2]{%
1278   \let\@gls@hyp@opt\@gls@alt@hyp@opt
1279   \def\@gls@alt@hyp@opt@char{#1}%
1280   \def\@gls@alt@hyp@opt@keys{#2}%
1281 }
```

\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 `\glxtrshort` to use `\glstryshort` and, similarly, the long form commands like `\acrlong` and `\glxtrlong` to use `\glstrylong`.

```
1282 \renewcommand*{\glsdohyperlink}[2]{%
1283   \hyperlink{#1}{\glxtrprotectlinks#2}}
```

`gl:disablehyper` Redefine in case we have an old version of glossaries.

```
1284 \ifundef\glsdonohyperlink
1285 {%
1286   \renewcommand{\gl:disablehyper}{%
1287     \KV@glslink@hyperfalse
1288     \let\@glslink\glsdonohyperlink
1289     \let\@glstarget\@secondoftwo
1290   }
1291 }
1292 {}
```

`gl:donohyperlink` 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.

```
1293 \def\glsdonohyperlink#1#2{\glxtrprotectlinks #2}
```

Reset `\@glslink` with patched versions:

```
1294 \ifcsundef{hyperlink}%
1295 {%
1296   \let\@glslink\glsdonohyperlink
1297 }%
1298 {%
1299   \let\@glslink\glsdohyperlink
1300 }
```

`gl:trprotectlinks` Make `\gls` (and variants) behave like the corresponding `\glstext` (and variants) with hyperlinking and indexing off.

```
1301 \newcommand*{\glxtrprotectlinks}{%
1302   \KV@glslink@hyperfalse
1303   \KV@glslink@noindextrue
1304   \let\@gls@\@glxtrp@text@
1305   \let\@Gls@\@Glsxtrp@text@
1306   \let\@GLS@\@GLSxtrp@text@
1307   \let\@glspl@\@glxtrp@plural@
1308   \let\@Glspl@\@Glsxtrp@plural@
1309   \let\@GLSpl@\@GLSxtrp@plural@
1310   \let\@glxtrshort@\@glxtrp@short@
1311   \let\@Glsxtrshort@\@Glsxtrp@short@
1312   \let\@GLSxtrshort@\@GLSxtrp@short@
1313   \let\@glxtrlong@\@glxtrp@long@
1314   \let\@Glsxtrlong@\@Glsxtrp@long@
1315   \let\@GLSxtrlong@\@GLSxtrp@long@}
```



```

1316 \let\@glxstrshortpl\@glxstrp@shortpl@
1317 \let\@Glsxtrshortpl\@Glsxtrp@shortpl@
1318 \let\@GLSxtrshortpl\@GLSxtrp@shortpl@
1319 \let\@glxtrlongpl\@glxstrp@longpl@
1320 \let\@Glsxtrlongpl\@Glsxtrp@longpl@
1321 \let\@GLSxtrlongpl\@GLSxtrp@longpl@
1322 \let\@acrshort\@glxstrp@acrshort@
1323 \let\@Acrshort\@Glsxtrp@acrshort@
1324 \let\@ACRshort\@GLSxtrp@acrshort@
1325 \let\@acrshortpl\@glxstrp@acrshortpl@
1326 \let\@Acrshortpl\@Glsxtrp@acrshortpl@
1327 \let\@ACRshortpl\@GLSxtrp@acrshortpl@
1328 \let\@acrlong\@glxstrp@acrlong@
1329 \let\@Acrlong\@Glsxtrp@acrlong@
1330 \let\@ACRlong\@GLSxtrp@acrlong@
1331 \let\@acrlongpl\@glxstrp@acrlongpl@
1332 \let\@Acrlongpl\@Glsxtrp@acrlongpl@
1333 \let\@ACRlongpl\@GLSxtrp@acrlongpl@
1334 }

```

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

@glxstrp@text@

```
1335 \def\@glxstrp@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtrp@text@

```
1336 \def\@Glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtrp@text@

```
1337 \def\@GLSxtrp@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtrp@plural@

```
1338 \def\@glxstrp@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

Glsxtrp@plural@

```
1339 \def\@Glsxtrp@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

GLSxtrp@plural@

```
1340 \def\@GLSxtrp@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxstrp@short@

```

1341 \def\@glxstrp@short@#1#2[#3]{%
1342 {%
1343   \glsssetabbrvfmt{\glscategory{#2}}%
1344   \glssabbrvfont{\glssentryshort{#2}}#3%
1345 }%
1346 }

```

Glsxtr@p@short@

```
1347 \def\@Glsxtr@p@short@#1#2[#3]{%
1348 {%
1349   \glsetabbrvfmt{\glscategory{#2}}%
1350   \glsabbrvfont{\Glsentryshort{#2}}#3%
1351 }%
1352 }
```

GLSxtr@p@short@

```
1353 \def\@GLSxtr@p@short@#1#2[#3]{%
1354 {%
1355   \glsetabbrvfmt{\glscategory{#2}}%
1356   \mfirstucMakeUppercase{\glsabbrvfont{\Glsentryshort{#2}}#3}%
1357 }%
1358 }
```

sxtr@p@shortpl@

```
1359 \def\@glxtr@p@shortpl@#1#2[#3]{%
1360 {%
1361   \glsetabbrvfmt{\glscategory{#2}}%
1362   \glsabbrvfont{\Glsentryshortpl{#2}}#3%
1363 }%
1364 }
```

Sxtr@p@shortpl@

```
1365 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
1366 {%
1367   \glsetabbrvfmt{\glscategory{#2}}%
1368   \glsabbrvfont{\Glsentryshortpl{#2}}#3%
1369 }%
1370 }
```

Sxtr@p@shortpl@

```
1371 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
1372 {%
1373   \glsetabbrvfmt{\glscategory{#2}}%
1374   \mfirstucMakeUppercase{\glsabbrvfont{\Glsentryshortpl{#2}}#3}%
1375 }%
1376 }
```

@glxtr@p@long@

```
1377 \def\@glxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
1378 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
1379 \def\@GLSxtr@p@long@#1#2[#3]{%
1380   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@
1381 \def\@glsxtr@p@longpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

lsxtr@p@longpl@
1382 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsentrylongpl{#2}}#3}}

LSxtr@p@longpl@
1383 \def\@GLSxtr@p@longpl@#1#2[#3]{%
1384 {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
1385 \def\@glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
1386 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
1387 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
1388 {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
1389 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1390 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1391 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
1392 {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
1393 \def\@glsxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
1394 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
1395 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
1396 {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}

tr@p@acrlongpl@
1397 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
1398 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
1399 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
1400 {\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.
1401 \renewcommand*{\@glsunset}[1]{%
1402   \@glsunset{#1}%
1403   \glsxtrpostunset{#1}%
1404 }%

glsxtrpostunset
1405 \newcommand*{\glsxtrpostunset}[1]{%

\@glslocalunset   Local unset.
1406 \renewcommand*{\@glslocalunset}[1]{%
1407   \@glslocalunset{#1}%
1408   \glsxtrpostlocalunset{#1}%
1409 }%

rpostlocalunset
1410 \newcommand*{\glsxtrpostlocalunset}[1]{%

\@glsreset   Global reset.
1411 \renewcommand*{\@glsreset}[1]{%
1412   \@glsreset{#1}%
1413   \glsxtrpostreset{#1}%
1414 }%

glsxtrpostreset
1415 \newcommand*{\glsxtrpostreset}[1]{%

\@glslocalreset   Local reset.
1416 \renewcommand*{\@glslocalreset}[1]{%
1417   \@glslocalreset{#1}%
1418   \glsxtrpostlocalreset{#1}%
1419 }%

rpostlocalreset
1420 \newcommand*{\glsxtrpostlocalreset}[1]{%

leEntryCounting   The first argument is the list of categories and the second argument is the value of the en-
                    trycount attribute.
1421 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
1422 \glsenableentrycount
```

Redefine \gls etc:

```
1423 \renewcommand*{\gls}{\cglsl}%
1424 \renewcommand*{\Gls}{\cGls}%
1425 \renewcommand*{\glspl}{\cglspl}%
1426 \renewcommand*{\Glspl}{\cGlspl}%
1427 \renewcommand*{\GLS}{\cGLS}%
1428 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
1429 \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
1430 \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
1431 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
1432   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
1433     can't be used with \string\GlsXtrEnableEntryCounting}%
1434   {Use one or other but not both commands}}%
1435 }
```

ycountunsetattr

```
1436 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
1437   \@for\@glsxtr@cat:=#1\do
1438   {%
1439     \ifdefempty{\@glsxtr@cat}{}%
1440     {%
1441       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
1442     }%
1443   }%
1444 }
```

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

nableentrycount

```
1445 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
1446 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
1447 \renewcommand*{\gls@defdocnewglossaryentry}{%
1448   \renewcommand*{\newglossaryentry}[2]{%
1449     \PackageError{glossaries}{\string\newglossaryentry\space
1450       may only be used in the preamble when entry counting has
1451       been activated}{If you use \string\glsenableentrycount\space
1452       you must place all entry definitions in the preamble not in
1453       the document environment}%
1454   }%
1455 }
```

New commands to access new fields:

```

1456 \newcommand*\glsentrycurrcount}[1]{%
1457   \ifcsundef{glo@glsdetoklabel{##1}@currcount}%
1458   {0}{\@gls@entry@field{##1}{currcount}}%
1459 }%
1460 \newcommand*\glsentryprevcount}[1]{%
1461   \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
1462   {0}{\@gls@entry@field{##1}{prevcount}}%
1463 }%

```

Adjust post unset and reset:

```

1464 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
1465 \renewcommand*\glsxtrpostunset}[1]{%
1466   \@glsxtr@entrycount@org@unset{##1}%
1467   \@gls@increment@currcount{##1}%
1468 }%
1469 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
1470 \renewcommand*\glsxtrpostlocalunset}[1]{%
1471   \@glsxtr@entrycount@org@localunset{##1}%
1472   \@gls@local@increment@currcount{##1}%
1473 }%
1474 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
1475 \renewcommand*\glsxtrpostreset}[1]{%
1476   \@glsxtr@entrycount@org@reset{##1}%
1477   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
1478 }%
1479 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
1480 \renewcommand*\glsxtrpostlocalreset}[1]{%
1481   \@glsxtr@entrycount@org@localreset{##1}%
1482   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
1483 }%

```

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

```

1484 \let\@cgl@s\@cgl@s
1485 \let\@cgl spl\@cgl spl
1486 \let\@cGLS\@cGLS
1487 \let\@cGL spl\@cGL spl
1488 \let\@cGLS\@cGLS
1489 \let\@cGLSpl\@cGLSpl

```

The rest is as the original definition.

```

1490 \AtEndDocument{\@gls@write@entrycounts}%
1491 \renewcommand*\@gls@entry@count}[2]{%
1492   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
1493 }%
1494 \let\glsenableentrycount\relax
1495 \renewcommand*\glsenableentryunitcount{%
1496   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
1497     can't be used with \string\glsenableentrycount}%

```

```

1498     {Use one or other but not both commands}%
1499   }%
1500 }

```

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.

```

1501 \renewcommand*{\@gls@write@entrycounts}{%
1502   \immediate\write\@auxout
1503     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
1504   \count@=0\relax
1505   \forallglsentries{\@glsentry}{%
1506     \glshasattribute{\@glsentry}{entrycount}%
1507     {%
1508       \ifglsused{\@glsentry}%
1509       {%
1510         \immediate\write\@auxout
1511           {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
1512       }%
1513     }%
1514     \advance\count@ by \@ne
1515   }%
1516 }%
1517 }%
1518 \ifnum\count@=0
1519   \GlossariesExtraWarningNoLine{Entry counting has been enabled
1520     \MessageBreak with \string\glsenableentrycount\space but the
1521     \MessageBreak attribute 'entrycount' hasn't
1522     \MessageBreak been assigned to any of the defined
1523     \MessageBreak entries}%
1524 \fi
1525 }

```

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

```

1526 \newcommand*{\glxtrifcounttrigger}[3]{%
1527   \glshasattribute{#1}{entrycount}%
1528   {%
1529     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
1530     #3%
1531   \else
1532     #2%
1533   \fi
1534 }%
1535 {#3}%
1536 }

```

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

\@@cgl's@

```

1537 \def\@@cgl's@#1#2[#3]{%
1538   \gl'sxtrifcounttrigger{#2}%
1539   {%
1540     \cgl'sformat{#2}{#3}%
1541     \gl'sunset{#2}%
1542   }%
1543   {%
1544     \@gl's@{#1}{#2}[#3]%
1545   }%
1546 }%
```

\@@cgl'spl@

```

1547 \def\@@cgl'spl@#1#2[#3]{%
1548   \gl'sxtrifcounttrigger{#2}%
1549   {%
1550     \cgl'splformat{#2}{#3}%
1551     \gl'sunset{#2}%
1552   }%
1553   {%
1554     \@gl'spl@{#1}{#2}[#3]%
1555   }%
1556 }%
```

\@@cGl's@

```

1557 \def\@@cGl's@#1#2[#3]{%
1558   \gl'sxtrifcounttrigger{#2}%
1559   {%
1560     \cGl'sformat{#2}{#3}%
1561     \gl'sunset{#2}%
1562   }%
1563   {%
1564     \@Gl's@{#1}{#2}[#3]%
1565   }%
1566 }%
```

\@@cGl'spl@

```

1567 \def\@@cGl'spl@#1#2[#3]{%
1568   \gl'sxtrifcounttrigger{#2}%
1569   {%
1570     \cGl'splformat{#2}{#3}%
1571     \gl'sunset{#2}%
1572   }%
1573   {%
1574     \@Gl'spl@{#1}{#2}[#3]%
1575   }%
1576 }%
```



```

\@cGLS@
1577 \def\@cGLS@#1#2[#3]{%
1578   \glxtrifcounttrigger{#2}%
1579   {%
1580     \cGLSformat{#2}{#3}%
1581     \glset{#2}%
1582   }%
1583   {%
1584     \@GLS@{#1}{#2}[#3]%
1585   }%
1586 }%

\@cGLSpl@
1587 \def\@cGLSpl@#1#2[#3]{%
1588   \glxtrifcounttrigger{#2}%
1589   {%
1590     \cGLSplformat{#2}{#3}%
1591     \glset{#2}%
1592   }%
1593   {%
1594     \@GLSpl@{#1}{#2}[#3]%
1595   }%
1596 }%
1597 %
1598 % Remove default warnings from \cs{cgl} etc so that it can be used
1599 % interchangeable with \cs{gl} etc.
1600 %\begin{macro}\@cgl@
1601 %   \begin{macrocode}
1602 \def\@cgl@#1#2[#3]{\@gl@{#1}{#2}[#3]}

\@cGl@
1603 \def\@cGl@#1#2[#3]{\@gl@{#1}{#2}[#3]}

\@cglspl@
1604 \def\@cglspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

\@cGlspl@
1605 \def\@cGlspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

    Add all upper case versions not provided by glossaries.

\cGLS
1606 \newrobustcmd*{\cGLS}{\@glshyp@opt\@cGLS}

\@cGLS   Defined the un-starred form. Need to determine if there is a final optional argument
1607 \newcommand*{\@cGLS}[2][ ]{%
1608   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[ ]}%
1609 }

```

```

\@cGLS@
1610 \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.
1611 \newcommand*\cGLSformat}[2]{%
1612   \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
1613 }

\cGLSpl
1614 \newrobustcmd*\cGLSpl{\@gls@hyp@opt\@cGLSpl}

\@cGLSpl  Defined the un-starred form. Need to determine if there is a final optional argument
1615 \newcommand*\@cGLSpl}[2][{}]{%
1616   \new@ifnextchar[\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[{}]}%
1617 }

\@cGLSpl@
1618 \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.
1619 \newcommand*\cGLSplformat}[2]{%
1620   \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
1621 }

        Modify the trigger formats to check for the regular attribute.

\cglformat
1622 \renewcommand*\cglformat}[2]{%
1623   \glsifregular{#1}
1624   {\glsentryfirst{#1}}%
1625   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
1626 }

\cGlsformat
1627 \renewcommand*\cGlsformat}[2]{%
1628   \glsifregular{#1}
1629   {\Glsentryfirst{#1}}%
1630   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
1631 }

\cglsplformat
1632 \renewcommand*\cglsplformat}[2]{%
1633   \glsifregular{#1}
1634   {\glsentryfirstplural{#1}}%
1635   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
1636 }

```

\cGlsplformat

```
1637 \renewcommand*{\cGlsplformat}[2]{%
1638   \glsifregular{#1}
1639   {\Glsentryfirstplural{#1}}%
1640   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2}%
1641 }
```

New code similar to above for unit counting.

defunitcounters

```
1642 \newcommand*{\@@newglossaryentry@defunitcounters}{%
1643   \edef\@glo@countunit{\csuse{\glstr@categoryattr@@\@glo@category @unitcount}}%
1644   \ifvoid\@glo@countunit
1645     {}%
1646     {%
1647       \@glstr@ifunitcounter{\@glo@countunit}%
1648       {}%
1649       {\expandafter\@glstr@addunitcounter\expandafter{\@glo@countunit}}%
1650     }%
1651 }
```

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

```
1652 \newcommand*{\@glstr@unitcountlist}{}%
```

@addunitcounter

```
1653 \newcommand*{\@glstr@addunitcounter}[1]{%
1654   \listadd{\@glstr@unitcountlist}{#1}%
1655   \ifcsundef{glstr@theunit@#1}
1656     {%
1657       \ifcsdef{theH#1}%
1658       {\csdef{glstr@theunit@#1}{\csuse{theH#1}}}%
1659       {\csdef{glstr@theunit@#1}{\csuse{the#1}}}%
1660     }%
1661   {}%
1662 }
```

r@ifunitcounter

```
1663 \newcommand*{\@glstr@ifunitcounter}[3]{%
1664   \xifinlist{#1}{\@glstr@unitcountlist}{#2}{#3}%
1665 }
```

urrentunitcount

```
1666 \newcommand*{\@glstr@currentunitcount}[1]{%
1667   glo@\glstetoklabel{#1}@currunit@\glstetattribute{#1}{unitcount}.%
1668   \csuse{glstr@theunit@\glstetattribute{#1}{unitcount}}%
1669 }
```

previousunitcount

```
1670 \newcommand*\@glsxtr@previousunitcount[1]{%
1671   glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
1672   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
1673 }
```

t@currunitcount

```
1674 \newcommand*\@gls@increment@currunitcount[1]{%
1675   \gls@hasattribute{#1}{unitcount}%
1676   {%
1677     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
1678     \ifcsundef{\@glsxtr@csname}%
1679     {%
1680       \csgdef{\@glsxtr@csname}{1}%
1681       \listcsxadd
1682         {glo@\glsdetoklabel{#1}@unitlist}%
1683         {\glsgetattribute{#1}{unitcount}.%
1684         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
1685       }%
1686     }%
1687     {%
1688       \csxdef{\@glsxtr@csname}%
1689       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
1690     }%
1691   }%
1692   {}%
1693 }
```

t@currunitcount

```
1694 \newcommand*\@gls@local@increment@currunitcount[1]{%
1695   \gls@hasattribute{#1}{unitcount}%
1696   {%
1697     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
1698     \ifcsundef{\@glsxtr@csname}%
1699     {%
1700       \csdef{\@glsxtr@csname}{1}%
1701       \listcseadd
1702         {glo@\glsdetoklabel{#1}@unitlist}%
1703         {\glsgetattribute{#1}{unitcount}.%
1704         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
1705       }%
1706     }%
1707     {%
1708       \csedef{\@glsxtr@csname}%
1709       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
1710     }%
1711   }%
1712   {}%
1713 }
```

r@currunitcount

```
1714 \newcommand*{\@glxstr@currunitcount}[2]{%
1715   \ifcsundef
1716     {glo@\glstoklabel{#1}@currunit@#2}%
1717     {0}%
1718     {\csuse{glo@\glstoklabel{#1}@currunit@#2}}%
1719 }%
```

r@prevunitcount

```
1720 \newcommand*{\@glxstr@prevunitcount}[2]{%
1721   \ifcsundef
1722     {glo@\glstoklabel{#1}@prevunit@#2}%
1723     {0}%
1724     {\csuse{glo@\glstoklabel{#1}@prevunit@#2}}%
1725 }%
```

entryunitcount

```
1726 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
1727   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
1728   \renewcommand*{\gls@defdocnewglossaryentry}{%
1729     \renewcommand*\newglossaryentry[2]{%
1730       \PackageError{glossaries}{\string\newglossaryentry\space
1731         may only be used in the preamble when entry counting has
1732         been activated}{If you use \string\glsenableentryunitcount\space
1733         you must place all entry definitions in the preamble not in
1734         the document environment}%
1735     }%
1736   }%
```

New commands to access new fields:

```
1737   \newcommand*{\glsentrycurrcount}[1]{%
1738     \@glxstr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
1739     \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
1740   }%
1741   \newcommand*{\glsentryprevcount}[1]{%
1742     \@glxstr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
1743     \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
1744   }%
```

Access total count:

```
1745   \newcommand*{\glsentryprevtotalcount}[1]{%
1746     \ifcsundef{glo@\glstoklabel{##1}@prevunittotal}%
1747     {0}%
1748     {%
1749       \number\csuse{glo@\glstoklabel{##1}@prevunittotal}
1750     }%
1751   }%
```

Access max value:

```

1752 \newcommand*{\glstentryprevmaxcount}[1]{%
1753   \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
1754   {0}%
1755   {%
1756     \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
1757   }%
1758 }%

```

Adjust post unset and reset:

```

1759 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
1760 \renewcommand*{\glstxtrpostunset}[1]{%
1761   \@glstxtr@entryunitcount@org@unset{##1}%
1762   \@glst@increment@currunitcount{##1}%
1763 }%
1764 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
1765 \renewcommand*{\glstxtrpostlocalunset}[1]{%
1766   \@glstxtr@entryunitcount@org@localunset{##1}%
1767   \@glst@local@increment@currunitcount{##1}%
1768 }%
1769 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
1770 \renewcommand*{\glstxtrpostreset}[1]{%
1771   \glshasattribute{##1}{unitcount}%
1772   {%
1773     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
1774     \ifcsundef{\@glstxtr@csname}%
1775     {}%
1776     {\csgdef{\@glstxtr@csname}{0}}%
1777   }%
1778   {}%
1779 }%
1780 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
1781 \renewcommand*{\glstxtrpostlocalreset}[1]{%
1782   \@glstxtr@entryunitcount@org@localreset{##1}%
1783   \glshasattribute{##1}{unitcount}%
1784   {%
1785     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
1786     \ifcsundef{\@glstxtr@csname}%
1787     {}%
1788     {\csdef{\@glstxtr@csname}{0}}%
1789   }%
1790   {}%
1791 }%

```

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

```

1792 \let\@cglst@\@cglst@
1793 \let\@cglstpl@\@cglstpl@
1794 \let\@cGLS@\@cGLS@
1795 \let\@cGLstpl@\@cGLstpl@

```

```

1796 \let\@cGLS@ \@cGLS@
1797 \let\@cGLSpl@ \@cGLSpl@

Write information to the aux file.

1798 \AtEndDocument{\@gls@write@entryunitcounts}%
1799 \renewcommand*{\@gls@entry@unitcount}[3]{%
1800   \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
1801   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
1802   {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
1803   {%
1804     \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
1805       \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
1806     }%
1807     \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
1808     {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
1809     {%
1810       \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
1811       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
1812       \fi
1813     }%
1814   }%
1815 \let\glsenableentryunitcount\relax
1816 \renewcommand*{\glsenableentrycount}{%
1817   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
1818     can't be used with \string\glsenableentryunitcount}%
1819   {Use one or other but not both commands}%
1820 }%
1821 }
1822 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

1823 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

1824 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
1825   \immediate\write\@auxout
1826   {\string\@gls@entry@unitcount
1827     {\@glsentry}%
1828     {\@glsxtr@currunitcount{\@glsentry}{#1}%
1829     }%
1830     {#1}}%
1831 }

```

entryunitcounts

```

1832 \newcommand*{\@gls@write@entryunitcounts}{%
1833   \immediate\write\@auxout
1834   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
1835   \count@=0\relax
1836   \forallglsentries{\@glsentry}{%

```

```

1837 \glshasattribute{\@gl Sentry}{unitcount}%
1838 {%
1839 \ifgl sused{\@gl Sentry}%
1840 {%
1841 \forlistcsloop
1842 {\@gl s@write@entryunitcounts@do}%
1843 {glo@\gl sdetoklabel{\@gl Sentry}@unitlist}%
1844 }%
1845 {}%
1846 \advance\count@ by \@ne
1847 }%
1848 {}%
1849 }%
1850 \ifnum\count@=0
1851 \GlossariesExtraWarningNoLine{Entry counting has been enabled
1852 \MessageBreak with \string\gl senableentryunitcount\space but the
1853 \MessageBreak attribute ‘unitcount’ hasn’t
1854 \MessageBreak been assigned to any of the defined
1855 \MessageBreak entries}%
1856 \fi
1857 }

```

`tryUnitCounting` 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.

```

1858 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

1859 \gl senableentryunitcount

```

Redefine `\gl s` etc:

```

1860 \renewcommand*{\gl s}{\cgl s}%
1861 \renewcommand*{\Gls}{\cGls}%
1862 \renewcommand*{\gl spl}{\cgl spl}%
1863 \renewcommand*{\Glspl}{\cGlspl}%
1864 \renewcommand*{\GLS}{\cGLS}%
1865 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

1866 \@gl sxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

1867 \let\GlsXtrEnableEntryUnitCounting\@gl sxtr@setentryunitcountunsetattr
1868 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
1869 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
1870 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
1871 {Use one or other but not both commands}}%
1872 }

```

`tcountunsetattr`

```

1873 \newcommand*{\@gl sxtr@setentryunitcountunsetattr}[3]{%
1874 \@for\@gl sxtr@cat:=#1\do

```



```

1875 {%
1876   \ifdefempty{\@glstr@cat}{}%
1877   {%
1878     \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
1879     \glsssetcategoryattribute{\@glstr@cat}{unitcount}{#3}%
1880   }%
1881 }%
1882 }

```

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`

```

1883 \renewcommand*{\SetGenericNewAcronym}{%
1884   \let\@Gls@entryname\@Gls@acrenryname
1885   \renewcommand{\newacronym}[4][]{%
1886     \ifdefempty{\@glssacronymlists}%
1887     {%
1888       \def\@glo@type{\acronymtype}%
1889       \setkeys{glossentry}{##1}%
1890       \DeclareAcronymList{\@glo@type}%
1891     }%
1892   }%
1893   \glsskeylisttok{##1}%
1894   \glsslabeltok{##2}%
1895   \glssshorttok{##3}%
1896   \glsslongtok{##4}%
1897   \newacronymhook
1898   \protected@edef\@do@newglossaryentry{%
1899     \noexpand\newglossaryentry{\the\glsslabeltok}%
1900     {%
1901       type=\acronymtype,%
1902       name={\expandonce{\acronymentry{##2}}},%
1903       sort={\acronymssort{\the\glssshorttok}{\the\glsslongtok}},%
1904       text={\the\glssshorttok},%
1905       short={\the\glssshorttok},%
1906       shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
1907       long={\the\glsslongtok},%
1908       longplural={\the\glsslongtok\noexpand\acrpluralsuffix},%
1909       category=acronym,%
1910       \GenericAcronymFields,%
1911       \the\glsskeylisttok

```

```

1912 }%
1913 }%
1914 \@do@newglossaryentry
1915 }%
1916 \renewcommand*{\acrfullfmt}[3]{%
1917   \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
1918 \renewcommand*{\Acrfullfmt}[3]{%
1919   \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
1920 \renewcommand*{\ACRfullfmt}[3]{%
1921   \glslink{##1}{##2}{%
1922     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
1923 \renewcommand*{\acrfullplfmt}[3]{%
1924   \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
1925 \renewcommand*{\Acrfullplfmt}[3]{%
1926   \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
1927 \renewcommand*{\ACRfullplfmt}[3]{%
1928   \glslink{##1}{##2}{%
1929     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
1930 \renewcommand*{\glstryfull}[1]{\genacrfullformat{##1}{}}}%
1931 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
1932 \renewcommand*{\glstryfullpl}[1]{\genplacrfullformat{##1}{}}}%
1933 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
1934 }

```

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:

```

1935 \let\@glstr@org@setacronymstyle\setacronymstyle
1936 \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`.

```

1937 \newcommand*{\MakeAcronymsAbbreviations}{%
1938   \renewcommand*{\newacronym}[4][{}]{%
1939     \newabbreviation[type=\acronymtype,category=acronym,##1]{##2}{##3}{##4}%
1940   }%
1941   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
1942   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
1943   \renewcommand*{\setacronymstyle}[1]{%
1944     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
1945     unavailable.
1946     Use \string\setabbreviationstyle\space instead.
1947     The original acronym interface can be restored with
1948     \string\RestoreAcronyms}{}%
1949 }%
1950 \renewcommand*{\newacronymstyle}[1]{%
1951   \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
1952   available unless you restore the original acronym interface with

```

```

1953 \string\RestoreAcronyms}%
1954 \@glstr@org@newacronymstyle{##1}%
1955 }%
1956 }

```

Switch acronyms to abbreviations:

```
1957 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```

1958 \newcommand*{\RestoreAcronyms}{%
1959 \SetGenericNewAcronym
1960 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
1961 \renewcommand{\acronymfont}[1]{##1}%
1962 \let\setacronymstyle\@glstr@org@setacronymstyle
1963 \let\newacronymstyle\@glstr@org@newacronymstyle
1964 \let\@gls@link@checkfirsthyper\@glstr@org@checkfirsthyper
1965 \glssetcategoryattribute{acronym}{regular}{false}%
1966 \setacronymstyle{long-short}%
1967 }

```

\glsacspace Allow the user to customise the maximum value.

```

1968 \renewcommand*{\glsacspace}[1]{%
1969 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
1970 \ifdim\dimen@<\glsacspacemax~\else\space\fi
1971 }

```

\glsacspacemax Value used in the above.

```
1972 \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

```
1973 \newcommand*{\@glstr@reg@glosslist}{}
```

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

```
1974 \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

```

1975 \renewcommand*{\makeglossaries}[1][]{%
1976   \ifblank{#1}%
1977   {\@glsxtr@org@makeglossaries}%
1978   {%
1979     \edef\@glsxtr@reg@glosslist{#1}%
1980     \ifundef{\glswrite}{\newwrite\glswrite}{}%
1981     \protected@write\@auxout{}{\string\providecommand
1982       \string\@glsorder[1]}%
1983     \protected@write\@auxout{}{\string\providecommand
1984       \string\@istfilename[1]}%
1985     \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
1986     \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
1987     \write\@auxout{\string\providecommand\string\@gls@reference[3]}%

```

Iterate through each supplied glossary type and activate it.

```

1988   \@for\@glo@type:=#1\do{%
1989     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
1990   }%

```

New glossaries must be created before \makeglossaries:

```

1991   \renewcommand*{\newglossary}[4][]{%
1992     \PackageError{glossaries}{New glossaries
1993       must be created before \string\makeglossaries}{You need
1994       to move \string\makeglossaries\space after all your
1995       \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

1996   \let\@makeglossary\relax
1997   \let\makeglossary\relax
1998   \let\makeglossaries\relax

```

Disable all commands that have no effect after \makeglossaries

```

1999   \@disable@onlypremakeg

```

Allow see key:

```

2000   \let\gls@checkseeallowed\relax

```

Suppress warning about no \makeglossaries

```

2001   \let\warn@nomakeglossaries\relax
2002   \def\warn@noprintglossary{%
2003     \GlossariesWarningNoLine{No \string\printglossary\space
2004       or \string\printglossaries\space
2005       found.^^J(Remove \string\makeglossaries\space if you don't
2006 want
2007       any glossaries.)^^JThis document will not have a glossary}%
2008   }%

```

Adjust display number list to check for type:

```

2009   \renewcommand*{\glsdisplaynumberlist}[1]{%
2010     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2011     {\@glsxtr@idx@displaynumberlist{##1}}%

```

```

2012   {\@glsxtr@noidx@displaynumberlist{##1}}}%
2013 }%

```

Adjust entry list:

```

2014   \renewcommand*{\glsentrynumberlist}[1]{%
2015     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2016     {\@glsxtr@idx@entrynumberlist{##1}}}%
2017     {\@glsxtr@noidx@entrynumberlist{##1}}}%
2018 }%

```

Adjust number list loop

```

2019   \renewcommand*{\glsnumberlistloop}[2]{%
2020     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2021     {%
2022       \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
2023       not available for glossary ‘##1’}{}%
2024     }%
2025     {\@glsxtr@noidx@numberlistloop{##1}{##2}}}%
2026 }%

```

Only sanitize sort for normal indexing glossaries.

```

2027   \renewcommand*{\glsprestandardsort}[3]{%
2028     \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
2029     {%
2030       \glsdosanitizesort
2031     }%
2032     {%
2033       \ifglssanitizesort
2034       \@gls@noidx@sanitizesort
2035       \else
2036       \@gls@noidx@nosanitizesort
2037       \fi
2038     }%
2039 }%

```

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

```

2040   \renewcommand*{\new@glossaryentry}[2]{%
2041     \PackageError{glossaries-extra}{Glossary entries must be defined
2042     in the preamble\MessageBreak when you use the optional argument
2043     of \string\makeglossaries}{Either move your definitions to the
2044     preamble or don't use the optional argument of
2045     \string\makeglossaries}%
2046 }%

```

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

```

2047   \renewcommand*{\@printgloss@setsort}{%
2048     \renewcommand*{\@glo@assign@sortkey}{%
2049       \edef\@glo@type{\@glo@type}%
2050       \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%

```

```

2051      {%
2052      \@@glo@no@assign@sortkey
2053      }%
2054      {%
2055      \@@glo@assign@sortkey
2056      }%
2057      }%
2058      \def\@glo@sorttype{\@glo@default@sorttype}%
2059      }%

```

Check automake setting:

```

2060      \ifglsautomake
2061      \renewcommand*{\@gls@doautomake}{%
2062      \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
2063      \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
2064      }%
2065      }%
2066      \fi
2067      }%
2068      }

```

Display number list for the regular version:

splaynumberlist

```

2069 \let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

2070 \newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
2071 \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
2072 \ifdef\@gls@loclist
2073 {%
2074 \def\@gls@noidxloclist@sep{%
2075 \def\@gls@noidxloclist@sep{%
2076 \def\@gls@noidxloclist@sep{%
2077 \glsnumlistsep
2078 }%
2079 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
2080 }%
2081 }%
2082 \def\@gls@noidxloclist@finalsep{}%
2083 \def\@gls@noidxloclist@prev{}%
2084 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
2085 \@gls@noidxloclist@finalsep
2086 \@gls@noidxloclist@prev
2087 }%
2088 {%
2089 ??\glsdoifexists{#1}%
2090 {%
2091 \GlossariesWarning{Missing location list for ‘#1’. Either

```

```

2092      a rerun is required or you haven't referenced the entry.}%
2093  }%
2094 }%
2095 }%
2096

```

And for the number list loop:

@numberlistloop

```

2097 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
2098   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2099   \let\@gls@org@glsnoidxdisplayloc@glsnoidxdisplayloc
2100   \let\@gls@org@glsseeformat@glsseeformat
2101   \let@glsnoidxdisplayloc#2\relax
2102   \let@glsseeformat#3\relax
2103   \ifdef\@gls@loclist
2104   {%
2105     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
2106   }%
2107   {%
2108     ??\glsdoifexists{#1}%
2109     {%
2110       \GlossariesWarning{Missing location list for ‘##1’. Either
2111       a rerun is required or you haven't referenced the entry.}%
2112     }%
2113   }%
2114   \let@glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
2115   \let@glsseeformat\@gls@org@glsseeformat
2116 }%

```

Same for entry number list.

entrynumberlist

```

2117 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
2118   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2119   \ifdef\@gls@loclist
2120   {%
2121     \glsnoidxloclist{\@gls@loclist}%
2122   }%
2123   {%
2124     ??\glsdoifexists{#1}%
2125     {%
2126       \GlossariesWarning{Missing location list for ‘#1’. Either
2127       a rerun is required or you haven't referenced the entry.}%
2128     }%
2129   }%
2130 }%

```

entrynumberlist

```

2131 \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

```

2132 \renewcommand{\@print@glossary}{%
2133   \makeatletter
2134   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2135   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2136   {}%
2137   {\glstrNoGlossaryWarning{\@glo@type}}%
2138   \ifglxindy
2139     \ifcsundef{@xdy@\@glo@type @language}%
2140     {%
2141       \edef\@do@auxoutstuff{%
2142         \noexpand\AtEndDocument{%
2143           \noexpand\immediate\noexpand\write\@auxout{%
2144             \string\providecommand\string\@xdylanguage[2]{}}%
2145           \noexpand\immediate\noexpand\write\@auxout{%
2146             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
2147         }%
2148       }%
2149     }%
2150     {%
2151       \edef\@do@auxoutstuff{%
2152         \noexpand\AtEndDocument{%
2153           \noexpand\immediate\noexpand\write\@auxout{%
2154             \string\providecommand\string\@xdylanguage[2]{}}%
2155           \noexpand\immediate\noexpand\write\@auxout{%
2156             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
2157               @language\endcsname}}%
2158         }%
2159       }%
2160     }%
2161     \@do@auxoutstuff
2162     \edef\@do@auxoutstuff{%
2163       \noexpand\AtEndDocument{%
2164         \noexpand\immediate\noexpand\write\@auxout{%
2165           \string\providecommand\string\@gls@codepage[2]{}}%
2166         \noexpand\immediate\noexpand\write\@auxout{%
2167           \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
2168       }%
2169     }%
2170     \@do@auxoutstuff
2171   \fi
2172   \renewcommand*{\@warn@nomakeglossaries}{%
2173     \GlossariesWarningNoLine{\string\makeglossaries\space
2174       hasn't been used,^^Jthe glossaries will not be updated}%
2175   }%
2176 }

```


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

`oGlsWarningHead` Header message.

```
2177 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
2178   This document is incomplete. The external file associated with
2179   the glossary ‘#1’ (which should be called \texttt{#2})
2180   hasn’t been created.%
2181 }
```

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

```
2182 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
2183   This has probably happened because there are no entries defined
2184   in this glossary.%
2185 }
```

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

```
2186 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
2187   If you don’t want this glossary,
2188   add \texttt{nomain} to your package option list when you load
2189   \texttt{glossaries-extra.sty}. For example:%
2190 }
```

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

```
2191 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
2192   Did you forget to use \texttt{type=#1} when you defined your
2193   entries? If you tried to load entries into this glossary with
2194   \texttt{\string\loadglsentries} did you remember to use
2195   \texttt{[#1]} as the optional argument? If you did, check that
2196   the definitions in the file you loaded all had the type set
2197   to \texttt{\string\glsdefaulttype}.%
2198 }
```

`arningCheckFile` Advisory message to check the file contents.

```
2199 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
2200   Check the contents of the file \texttt{#1}. If
2201   it’s empty, that means you haven’t indexed any of your entries in this
2202   glossary (using commands like \texttt{\string\gls} or
2203   \texttt{\string\glsadd}) so this list can’t be generated.
2204   If the file isn’t empty, the document build process hasn’t been
2205   completed.%
2206 }
```

`WarningAutoMake` Message when automake option has been used.

```
2207 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
2208   You may need to rerun \LaTeX. If you already have, it may be that
2209   \TeX’s shell escape doesn’t allow you to run
2210   \ifglxindy xindy\else makeindex\fi. Check the
2211   transcript file \texttt{\jobname.log}. If the shell escape is
```

```

2212 disabled, try one of the following:
2213
2214 \begin{itemize}
2215   \item Run the external (Lua) application:
2216
2217     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2218
2219   \item Run the external (Perl) application:
2220
2221     \texttt{makeglossaries \string"\jobname\string"}
2222 \end{itemize}
2223
2224 Then rerun \LaTeX\ on this document.
2225 \GlossariesExtraWarning{Rerun required to build the
2226 glossary ‘#1’ or check TeX’s shell escape allows
2227 you to run \ifglxindy xindy\else makeindex\fi}%
2228 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

2229 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
2230   You need to either replace \texttt{\string\makenoidxglossaries}
2231   with \texttt{\string\makeglossaries} or replace
2232   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
2233   \texttt{\string\printnoidxglossary}
2234   (or \texttt{\string\printnoidxglossaries}) and then rebuild
2235   this document.%
2236 }

```

arningBuildInfo Build advice.

```

2237 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
2238   Try one of the following:
2239   \begin{itemize}
2240     \item Add \texttt{automake} to your package option list when you load
2241       \texttt{glossaries-extra.sty}. For example:
2242
2243       \texttt{\string\usepackage[automake]%
2244         \glsoopenbrace glossaries-extra\glsclosebrace}
2245
2246     \item Run the external (Lua) application:
2247
2248       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2249
2250     \item Run the external (Perl) application:
2251
2252       \texttt{makeglossaries \string"\jobname\string"}
2253   \end{itemize}
2254
2255   Then rerun \LaTeX\ on this document.%
2256 }

```

oGlsWarningTail Final paragraph.

```
2257 \newcommand{\GlsXtrNoGlsWarningTail}{%
2258   This message will be removed once the problem has been fixed.%
2259 }
```

GlsWarningNoOut No out file created. Build advice.

```
2260 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
2261   The file \texttt{#1} doesn't exist. This most likely means you haven't used
2262   \texttt{\string\makeglossaries} or you have used
2263   \texttt{\string\nofiles}. If this is just a draft version of the
2264   document, you can suppress this message using the
2265   \texttt{nomissingglstext} package option.%
2266 }
```

glossarywarning

```
2267 \newcommand*{@\glstr@defaultnoglossarywarning}[1]{%
2268   \glossarysection[\glossarytoctitle]{\glossarytitle}
2269   \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
2270   \par
2271   \glstrifemptyglossary{#1}%
2272   {%
2273     \GlsXtrNoGlsWarningEmptyStart\space
2274     \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
2275     \medskip
2276     \noindent\texttt{\string\usepackage[nomain\ifglstracronym ,acronym\fi]%
2277       \glstrbrace glossaries-extra\glstrclosebrace}
2278     \medskip
2279     }%
2280     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
2281   }%
2282   {%
2283     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
2284     {%
2285       \GlsXtrNoGlsWarningCheckFile
2286       {\jobname.\csname @glotype@\@glo@type @out\endcsname}
2287
2288       \ifglstrautomake
2289
2290       \GlsXtrNoGlsWarningAutoMake{#1}
2291
2292     \else
2293
2294       \ifthenelse{\equal{#1}{main}}{%
2295       {%
2296         \GlsXtrNoGlsWarningEmptyMain\par
2297         \medskip
2298         \noindent\texttt{\string\usepackage[nomain]%
2299           \glstrbrace glossaries-extra\glstrclosebrace}
2300         \medskip
```

```

2301      }%
2302      {}%
2303
2304      \ifdefequal\makeglossaries\@no@makeglossaries
2305      {%
2306          \GlsXtrNoGlsWarningMisMatch
2307      }%
2308      {%
2309          \GlsXtrNoGlsWarningBuildInfo
2310      }%
2311      \fi
2312      }%
2313      {%
2314          \GlsXtrNoGlsWarningNoOut
2315          {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
2316      }%
2317      }%
2318      \par
2319      \GlsXtrNoGlsWarningTail
2320      }

```

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.

```

2321 \@ifpackageloaded{glossaries-accsupp}
2322 {

```

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

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

```

2323 \newcommand*\glsaccessname}[1]{%
2324     \glsnameaccessdisplay
2325     {%
2326         \glsentryname{#1}%
2327     }%
2328     {#1}%
2329 }

```

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

```

2330 \newcommand*\Glsaccessname}[1]{%
2331     \glsnameaccessdisplay
2332     {%
2333         \Glsentryname{#1}%
2334     }%

```

```

2335     {#1}%
2336 }

```

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

```

2337 \newcommand*{\GLSaccessname}[1]{%
2338   \glsnameaccessdisplay
2339   {%
2340     \mfirstucMakeUppercase{\glstryname{#1}}%
2341   }%
2342   {#1}%
2343 }

```

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

```

2344 \newcommand*{\glsaccessstext}[1]{%
2345   \glstextaccessdisplay
2346   {%
2347     \glstrytext{#1}%
2348   }%
2349   {#1}%
2350 }

```

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

```

2351 \newcommand*{\Glsaccessstext}[1]{%
2352   \glstextaccessdisplay
2353   {%
2354     \Glsstrytext{#1}%
2355   }%
2356   {#1}%
2357 }

```

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

```

2358 \newcommand*{\GLSAccessstext}[1]{%
2359   \glstextaccessdisplay
2360   {%
2361     \mfirstucMakeUppercase{\glstrytext{#1}}%
2362   }%
2363   {#1}%
2364 }

```

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

```

2365 \newcommand*{\glsaccessplural}[1]{%
2366   \glspluralaccessdisplay
2367   {%
2368     \glstryplural{#1}%
2369   }%
2370   {#1}%
2371 }

```

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

```

2372 \newcommand*{\Glsaccessplural}[1]{%
2373   \glspluralaccessdisplay
2374   {%
2375     \Glsentryplural{#1}%
2376   }%
2377   {#1}%
2378 }

```

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

```

2379 \newcommand*{\GLSaccessplural}[1]{%
2380   \glspluralaccessdisplay
2381   {%
2382     \mfirstucMakeUppercase{\glsentryplural{#1}}%
2383   }%
2384   {#1}%
2385 }

```

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

```

2386 \newcommand*{\glsaccessfirst}[1]{%
2387   \glsfirstaccessdisplay
2388   {%
2389     \glsentryfirst{#1}%
2390   }%
2391   {#1}%
2392 }

```

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

```

2393 \newcommand*{\GLSaccessfirst}[1]{%
2394   \glsfirstaccessdisplay
2395   {%
2396     \Glsentryfirst{#1}%
2397   }%
2398   {#1}%
2399 }

```

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

```

2400 \newcommand*{\GLSaccessfirst}[1]{%
2401   \glsfirstaccessdisplay
2402   {%
2403     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
2404   }%
2405   {#1}%
2406 }

```

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

```

2407 \newcommand*{\glsaccessfirstplural}[1]{%
2408   \glsfirstpluralaccessdisplay
2409   {%
2410     \glsentryfirstplural{#1}%
2411   }%
2412   {#1}%
2413 }

```

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

```

2414 \newcommand*{\Glsaccessfirstplural}[1]{%
2415   \glsfirstpluralaccessdisplay
2416   {%
2417     \Glsentryfirstplural{#1}%
2418   }%
2419   {#1}%
2420 }

```

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

```

2421 \newcommand*{\GLSaccessfirstplural}[1]{%
2422   \glsfirstpluralaccessdisplay
2423   {%
2424     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
2425   }%
2426   {#1}%
2427 }

```

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

```

2428 \newcommand*{\glsaccesssymbol}[1]{%
2429   \glssymbolaccessdisplay
2430   {%
2431     \glsentrysymbol{#1}%
2432   }%
2433   {#1}%
2434 }

```

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

```

2435 \newcommand*{\Glsaccesssymbol}[1]{%
2436   \glssymbolaccessdisplay
2437   {%
2438     \Glsentrysymbol{#1}%
2439   }%
2440   {#1}%
2441 }

```

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

```

2442 \newcommand*{\GLSaccesssymbol}[1]{%

```

```

2443 \glssymbolaccessdisplay
2444 {%
2445 \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
2446 }%
2447 {#1}%
2448 }

```

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

```

2449 \newcommand*{\glaccesssymbolplural}[1]{%
2450 \glssymbolpluralaccessdisplay
2451 {%
2452 \glsentrysymbolplural{#1}%
2453 }%
2454 {#1}%
2455 }

```

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

```

2456 \newcommand*{\Glsaccesssymbolplural}[1]{%
2457 \glssymbolpluralaccessdisplay
2458 {%
2459 \Glsentrysymbolplural{#1}%
2460 }%
2461 {#1}%
2462 }

```

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

```

2463 \newcommand*{\GLSaccesssymbolplural}[1]{%
2464 \glssymbolpluralaccessdisplay
2465 {%
2466 \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
2467 }%
2468 {#1}%
2469 }

```

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

```

2470 \newcommand*{\glaccessdesc}[1]{%
2471 \glsdescriptionaccessdisplay
2472 {%
2473 \glentrydesc{#1}%
2474 }%
2475 {#1}%
2476 }

```

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

```

2477 \newcommand*{\Glsaccessdesc}[1]{%
2478 \glsdescriptionaccessdisplay

```



```

2479     {%
2480         \Glsentrydesc{#1}%
2481     }%
2482     {#1}%
2483 }

```

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

```

2484 \newcommand*{\GLSaccessdesc}[1]{%
2485     \glsdescriptionaccessdisplay
2486     {%
2487         \mfirstucMakeUppercase{\glsentrydesc{#1}}%
2488     }%
2489     {#1}%
2490 }

```

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

```

2491 \newcommand*{\glsaccessdescplural}[1]{%
2492     \glsdescriptionpluralaccessdisplay
2493     {%
2494         \glsentrydescplural{#1}%
2495     }%
2496     {#1}%
2497 }

```

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

```

2498 \newcommand*{\Glsaccessdescplural}[1]{%
2499     \glsdescriptionpluralaccessdisplay
2500     {%
2501         \Glsentrydescplural{#1}%
2502     }%
2503     {#1}%
2504 }

```

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

```

2505 \newcommand*{\GLSaccessdescplural}[1]{%
2506     \glsdescriptionpluralaccessdisplay
2507     {%
2508         \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
2509     }%
2510     {#1}%
2511 }

```

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

```

2512 \newcommand*{\glsaccessshort}[1]{%
2513     \glsshortaccessdisplay
2514     {%
2515         \glsentryshort{#1}%

```

```

2516     }%
2517     {#1}%
2518 }

```

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

```

2519 \newcommand*\Glsaccessshort}[1]{%
2520   \glsshortaccessdisplay
2521   {%
2522     \Glsentryshort{#1}%
2523   }%
2524   {#1}%
2525 }

```

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

```

2526 \newcommand*\GLSaccessshort}[1]{%
2527   \glsshortaccessdisplay
2528   {%
2529     \mfirstucMakeUppercase{\Glsentryshort{#1}}%
2530   }%
2531   {#1}%
2532 }

```

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

```

2533 \newcommand*\lsaccessshortpl}[1]{%
2534   \glsshortpluralaccessdisplay
2535   {%
2536     \Glsentryshortpl{#1}%
2537   }%
2538   {#1}%
2539 }

```

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

```

2540 \newcommand*\lsaccessshortpl}[1]{%
2541   \glsshortpluralaccessdisplay
2542   {%
2543     \Glsentryshortpl{#1}%
2544   }%
2545   {#1}%
2546 }

```

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

```

2547 \newcommand*\LSaccessshortpl}[1]{%
2548   \glsshortpluralaccessdisplay
2549   {%
2550     \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
2551   }%

```

```

2552     {#1}%
2553 }

```

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

```

2554 \newcommand*\glsaccesslong[1]{%
2555     \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
2556 }

```

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

```

2557
2558 \newcommand*\Glsaccesslong[1]{%
2559     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
2560 }

```

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

```

2561 \newcommand*\GLSaccesslong[1]{%
2562     \glslongaccessdisplay
2563     {%
2564         \mfirstucMakeUppercase{\glsentrylong{#1}}%
2565     }%
2566     {#1}%
2567 }

```

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

```

2568 \newcommand*\glsaccesslongpl[1]{%
2569     \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
2570 }

```

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

```

2571
2572 \newcommand*\Glsaccesslongpl[1]{%
2573     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
2574 }

```

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

```

2575 \newcommand*\GLSaccesslongpl[1]{%
2576     \glslongpluralaccessdisplay
2577     {%
2578         \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
2579     }%
2580     {#1}%
2581 }

```

End of if part

```

2582 }
2583 {

```

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

`\glsaccessname` Display the name value (no link and no check for existence).
2584 `\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.
2585 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.
2586 `\newcommand*{\GLSaccessname}[1]{%`
2587 `\protect\mfirstucMakeUppercase{\glsentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).
2588 `\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.
2589 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
2590 `\newcommand*{\GLSaccesstext}[1]{%`
2591 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
2592 `\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.
2593 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
2594 `\newcommand*{\GLSaccessplural}[1]{%`
2595 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
2596 `\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.
2597 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
2598 `\newcommand*{\GLSaccessfirst}[1]{%`
2599 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
2600 `\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.

2601 \newcommand*{\GLsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}

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

2602 \newcommand*{\GLSaccessfirstplural}[1]{%

2603 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}

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

2604 \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.

2605 \newcommand*{\GLsaccesssymbol}[1]{\Glsentrysymbol{#1}}

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

2606 \newcommand*{\GLSaccesssymbol}[1]{%

2607 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}

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

2608 \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.

2609 \newcommand*{\GLsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}

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

2610 \newcommand*{\GLSaccesssymbolplural}[1]{%

2611 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}

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

2612 \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.

2613 \newcommand*{\GLsaccessdesc}[1]{\Glsentrydesc{#1}}

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

2614 \newcommand*{\GLSaccessdesc}[1]{%

2615 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}

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

2616 \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.
2617 \newcommand*{\GLsaccessdesclplural}[1]{\Glsentrydesclplural{#1}}

ccessdesclplural Display the desclplural value (no link and no check for existence). converted to upper case.
2618 \newcommand*{\GLSaccessdesclplural}[1]{%
2619 \protect\mfirstucMakeUppercase{\glsentrydesclplural{#1}}}

\glsaccessshort Display the short form (no link and no check for existence).
2620 \newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}

\GLsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).
2621 \newcommand*{\GLsaccessshort}[1]{\Glsentryshort{#1}}

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.
2622 \newcommand*{\GLSaccessshort}[1]{%
2623 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}

lsaccessshortpl Display the short plural form (no link and no check for existence).
2624 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).
2625 \newcommand*{\GLsaccessshortpl}[1]{\Glsentryshortpl{#1}}

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.
2626 \newcommand*{\GLSaccessshortpl}[1]{%
2627 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}

\glsaccesslong Display the long form (no link and no check for existence).
2628 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}

\GLsaccesslong Display the long form (no link and no check for existence).
2629 \newcommand*{\GLsaccesslong}[1]{\Glsentrylong{#1}}

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.
2630 \newcommand*{\GLSaccesslong}[1]{%
2631 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}

glsaccesslongpl Display the long plural form (no link and no check for existence).
2632 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}

GLsaccesslongpl Display the long plural form (no link and no check for existence).
2633 \newcommand*{\GLsaccesslongpl}[1]{\Glsentrylongpl{#1}}

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

```
2634 \newcommand*{\GLSaccesslongpl}[1]{%
2635 \protect\mfirstucMakeUppercase{\glstrylongpl{#1}}}
```

End of else part

```
2636 }
```

1.5 Categories

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

```
2637 \glssaddstoragekey{category}{general}{\glscategory}
```

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

```
2638 \newcommand{\glisifcategory}[4]{%
2639 \ifglsfieq{#1}{category}{#2}{#3}{#4}%
2640 }
```

Categories can have attributes.

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

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

```
2641 \newcommand*{\glissetcategoryattribute}[3]{%
2642 \csdef{@glstr@categoryattr@@#1@#2}{#3}%
2643 }
```

`categoryattribute` `\glsggetcategoryattribute{<category>}{<attribute-label>}`

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

```
2644 \newcommand*{\glsggetcategoryattribute}[2]{%
2645 \csuse{@glstr@categoryattr@@#1@#2}%
2646 }
```

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

Tests if the category has the given attribute set.

```
2647 \newcommand*{\glshascategoryattribute}[4]{%
2648 \ifcsvoid{@glstr@categoryattr@@#1@#2}{#4}{#3}%
2649 }
```

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

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

```
2650 \newcommand*{\glsssetattribute}[3]{%
2651   \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
2652 }
```

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

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

```
2653 \newcommand*{\glssgetattribute}[2]{%
2654   \glssgetcategoryattribute{\glscategory{#1}}{#2}%
2655 }
```

`\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.

```
2656 \newcommand*{\glsshasattribute}[4]{%
2657   \ifglssentryexists{#1}%
2658   {\glsshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
2659   {#4}%
2660 }
```

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

True if category has the attribute with the given value.

```
2661 \newcommand{\glssifcategoryattribute}[5]{%
2662   \ifcsundef{@glssxtr@categoryattr@#1@#2}%
2663   {#5}%
2664   {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
2665 }
```

`\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.

```
2666 \newcommand{\glsifattribute}[5]{%
2667   \ifglstryexists{#1}%
2668   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
2669   {#5}%
2670 }
```

Set attributes for the default general category:

```
2671 \glssetcategoryattribute{general}{regular}{true}
```

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

```
2672 \glssetcategoryattribute{acronym}{regular}{true}
```

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

```
2673 \newcommand*{\glssetregularcategory}[1]{%
2674   \glssetcategoryattribute{#1}{regular}{true}%
2675 }
```

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

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

```
2676 \newcommand{\glsifregularcategory}[3]{%
2677   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
2678 }
```

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

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

```
2679 \newcommand{\glsifnotregularcategory}[3]{%
2680   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
2681 }
```

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

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

```
2682 \newcommand{\glsifregular}[3]{%
2683   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
2684 }
```

\glsifnotregular

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

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

```
2685 \newcommand{\glsifnotregular}[3]{%
2686   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
2687 }
```

\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.

```
2688 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
2689   \forallglossaries[#1]{#3}%
2690   {%
2691     \forlgsentries[#3]{#4}%
2692     {%
2693       \glsifcategory{#4}{#2}{#5}{}%
2694     }%
2695   }%
2696 }
```

\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.

```
2697 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
2698   \forallglossaries[#1]{#4}%
2699   {%
2700     \forlgsentries[#4]{#5}%
2701     {%
2702       \glsifattribute{#5}{#2}{#3}{#6}{}%
2703     }%
2704   }%
2705 }
```

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

```
2706 \ifdef\newterm
2707 {%
```

`\newterm`

```
2708 \renewcommand*\newterm}[2] [] {%
2709 \newglossaryentry{#2}%
2710 {type={index},category=index,name={#2},%
2711 description={\glstrpostdescription\nopostdesc},#1}%
2712 }
```

Indexed terms are regular by default.

```
2713 \glsssetcategoryattribute{index}{regular}{true}
```

`trpostdescindex`

```
2714 \newcommand*\glstrpostdescindex{}
2715 }
2716 {}
```

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`.

```
2717 \ifdef\printsymbols
2718 {%
```

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

```
2719 \newcommand*\glstrnewsymbol}[3] [] {%
2720 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
2721 }
```

Symbols are regular by default.

```
2722 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
2723 \newcommand*\glstrpostdescsymbol{}
2724 }
2725 {}
```

Similar for the `numbers` option.

```
2726 \ifdef\printnumbers
2727 {%
```

glsxtrnewnumber

```
2728 \ifdef\printnumbers
2729   \newcommand*{\glsxtrnewnumber}[3] [] {%
2730     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
2731   }
```

Numbers are regular by default.

```
2732   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
2733   \newcommand*{\glsxtrpostdescnumber}{}

2734 }
2735 {}
```

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.

```
2736 \newcommand*{\glsxtrsetcategory}[2] {%
2737   \@for\@glsxtr@label:=#1\do
2738   {%
2739     \glsfieldxddef{\@glsxtr@label}{category}{#2}%
2740   }%
2741 }
```

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.

```
2742 \newcommand*{\glsxtrsetcategoryforall}[2] {%
2743   \forallglossaries[#1]{\@glsxtr@type}{%
2744     \forglentries[\@glsxtr@type]{\@glsxtr@label}%
2745     {%
2746       \glsfieldxddef{\@glsxtr@label}{category}{#2}%
2747     }%
2748   }%
2749 }
```

glsxtrfieldtitlecase

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

Apply title casing to the contents of the given field.

```
2750 \newcommand*{\glsxtrfieldtitlecase}[2] {%
2751   \expandafter\xcapitalisewords\expandafter
2752     {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
2753 }
```

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.

```

2754 \@ifpackageloaded{glossaries-accsupp}
2755 {
2756   \renewcommand*{\glossentrydesc}[1]{%
2757     \glsdoifexistsorwarn{#1}%
2758     {%
2759       \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

2760     \glshasattribute{#1}{glossdescfont}%
2761     {%
2762       \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
2763       \ifcsdef{\@glxtr@attrval}%
2764       {%
2765         \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
2766       }%
2767       {%
2768         \GlossariesExtraWarning{Unknown control sequence name
2769           ‘\@glxtr@attrval’ supplied in glossdescfont attribute
2770           for entry ‘#1’. Ignoring}%
2771         \let\@glxtr@glossdescfont\@firstofone
2772       }%
2773     }%
2774     {\let\@glxtr@glossdescfont\@firstofone}%
2775     \gl@ifattribute{#1}{glossdesc}{firstuc}%
2776     {%
2777       \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
2778     }%
2779     {%
2780       \gl@ifattribute{#1}{glossdesc}{title}%
2781       {%
2782         \@glxtr@do@titlecaps@warn
2783         \glsdescriptionaccessdisplay
2784         {%
2785           \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
2786         }%
2787         {#1}%
2788       }%
2789     }%
2790     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
2791     }%
2792   }%
2793 }%
2794 }
2795 }
2796 {
2797   \renewcommand*{\glossentrydesc}[1]{%
2798     \glsdoifexistsorwarn{#1}%

```

```

2799  {%
2800    \glsetabbrvfmt{\glscategory{#1}}%
2801    \glshasattribute{#1}{glossdescfont}%
2802    {%
2803      \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
2804      \ifcsdef{\@glxtr@attrval}%
2805        {%
2806          \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
2807        }%
2808        {%
2809          \GlossariesExtraWarning{Unknown control sequence name
2810            '\@glxtr@attrval' supplied in glossdescfont attribute
2811            for entry '#1'. Ignoring}%
2812          \let\@glxtr@glossdescfont\@firstofone
2813        }%
2814      }%
2815      {\let\@glxtr@glossdescfont\@firstofone}%
2816      \gl@ifattribute{#1}{glossdesc}{firstuc}%
2817      {%
2818        \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
2819      }%
2820      {%
2821        \gl@ifattribute{#1}{glossdesc}{title}%
2822        {%
2823          \@glxtr@do@titlecaps@warn
2824          \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
2825        }%
2826        {%
2827          \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
2828        }%
2829      }%
2830    }%
2831  }
2832 }

```

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

```

2833 \ifpackageloaded{glossaries-accsupp}
2834 {
2835   \renewcommand*{\glossentryname}[1]{%
2836     \glsoifexistsorwarn{#1}%
2837     {%
2838       \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

2839   \glshasattribute{#1}{glossnamefont}%
2840   {%
2841     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
2842     \ifcsdef{\@glxtr@attrval}%
2843       {%

```

```

2844 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
2845 }%
2846 {%
2847 \GlossariesExtraWarning{Unknown control sequence name
2848 '\@glxtr@attrval' supplied in glossnamefont attribute
2849 for entry '#1'. Reverting to default \string\glsnamefont}%
2850 \let\@glxtr@glossnamefont\glsnamefont
2851 }%
2852 }%
2853 {\let\@glxtr@glossnamefont\glsnamefont}%
2854 \glsifattribute{#1}{glossname}{firstuc}%
2855 {%
2856 \glsnameaccessdisplay
2857 {%
2858 \@glxtr@glossnamefont{\Glsentryname{#1}}%
2859 }%
2860 {#1}%
2861 }%
2862 {%
2863 \glsifattribute{#1}{glossname}{title}%
2864 {%
2865 \@glxtr@do@titlecaps@warn
2866 \glsnameaccessdisplay
2867 {%
2868 \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
2869 }%
2870 {#1}%
2871 }%
2872 {%
2873 \glsifattribute{#1}{glossname}{uc}%
2874 {%
2875 \glsnameaccessdisplay
2876 {%

```

Hide the label from the upper-casing command.

```

2877 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
2878 \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
2879 }%
2880 {#1}%
2881 }%
2882 {%
2883 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
2884 \glsnameaccessdisplay
2885 {%
2886 \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
2887 }%
2888 {#1}%
2889 }%
2890 }%
2891 }%

```

Do post-name hook:

```

2892     \glxtrpostnamehook{#1}%
2893   }%
2894 }
2895 }
2896 {
2897   \renewcommand*{\glossentryname}[1]{%
2898     \glsoifexistsorwarn{#1}%
2899     {%
2900       \glsetabbrvfmt{\glscategory{#1}}%
2901       \glshasattribute{#1}{glossnamefont}%
2902       {%
2903         \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
2904         \ifcsdef{\@glxtr@attrval}%
2905         {%
2906           \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
2907         }%
2908         {%
2909           \GlossariesExtraWarning{Unknown control sequence name
2910             '\@glxtr@attrval' supplied in glossnamefont attribute
2911             for entry '#1'. Reverting to default \string\glnamefont}%
2912           \let\@glxtr@glossnamefont\glnamefont
2913         }%
2914       }%
2915       {\let\@glxtr@glossnamefont\glnamefont}%
2916       \glsoifattribute{#1}{glossname}{firstuc}%
2917       {%
2918         \@glxtr@glossnamefont{\Glsentryname{#1}}%
2919       }%
2920       {%
2921         \glsoifattribute{#1}{glossname}{title}%
2922         {%
2923           \@glxtr@do@titlecaps@warn
2924           \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
2925         }%
2926         {%
2927           \glsoifattribute{#1}{glossname}{uc}%
2928           {%

```

Hide the label from the upper-casing command.

```

2929     \letcs{\glo@name}{glo@\glsoifdetoklabel{#1}@name}%
2930     \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
2931   }%
2932   {%

```

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.

```

2933     \letcs{\glo@name}{glo@\glsoifdetoklabel{#1}@name}%
2934     \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
2935   }%

```



```

2936     }%
2937 }%
    Do post-name hook.
2938     \glsxtrpostnamehook{#1}%
2939 }%
2940 }
2941 }

```

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

```

2942 \@ifpackageloaded{glossaries-accsupp}
2943 {
2944     \renewcommand*{\Glossentryname}[1]{%
2945         \glsdoifexistsorwarn{#1}%
2946         {%
2947             \glssetabbrvfmt{\glscategory{#1}}%
2948             \glsattribute{#1}{glossnamefont}%
2949             {%
2950                 \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
2951                 \ifcsdef{\@glsxtr@attrval}%
2952                 {%
2953                     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
2954                 }%
2955                 {%
2956                     \GlossariesExtraWarning{Unknown control sequence name
2957                     '\@glsxtr@attrval' supplied in glossnamefont attribute
2958                     for entry '#1'. Reverting to default \string\glsnamefont}%
2959                     \let\@glsxtr@glossnamefont\glsnamefont
2960                 }%
2961             }%
2962             {\let\@glsxtr@glossnamefont\glsnamefont}%
2963             \glsnameaccessdisplay
2964             {%
2965                 \@glsxtr@glossnamefont{\Glsentryname{#1}}%
2966             }%
2967             {#1}%

```

Do post-name hook:

```

2968     \glsxtrpostnamehook{#1}%
2969 }%
2970 }
2971 }
2972 {
2973     \renewcommand*{\Glossentryname}[1]{%
2974         \glsdoifexistsorwarn{#1}%
2975         {%
2976             \glssetabbrvfmt{\glscategory{#1}}%
2977             \glsattribute{#1}{glossnamefont}%

```

```

2978    {%
2979        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
2980        \ifcsdef{\@glsxtr@attrval}%
2981        {%
2982            \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
2983        }%
2984        {%
2985            \GlossariesExtraWarning{Unknown control sequence name
2986                '\@glsxtr@attrval' supplied in glossnamefont attribute
2987                for entry '#1'. Reverting to default \string\glsnamefont}%
2988            \let\@glsxtr@glossnamefont\glsnamefont
2989        }%
2990    }%
2991    {\let\@glsxtr@glossnamefont\glsnamefont}%
2992    \@glsxtr@glossnamefont{\Glsentryname{#1}}}%

```

Do post-name hook:

```

2993        \glsxtrpostnamehook{#1}%
2994    }%
2995 }
2996 }

```

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.

```

2997 \newcommand*{\glsxtrpostnamehook}[1]{%
2998     \def\@glsnumberformat{glsnumberformat}%
2999     \glsxtrdoautoindexname{#1}{indexname}%

```

Allow categories to hook in here.

```

3000     \csuse{glsxtrpostname\glscategory{\glscurrententrylabel}}%
3001 }

```

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

```

3002 \newif\if@glsxtr@format@override
3003 \@glsxtr@format@overridefalse

```

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

`xFormatOverride`

```

3004 \@ifpackageloaded{hyperref}
3005 {

```

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

```

3006     \ifHy@hyperindex
3007     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%

```

```

3008      \@glxtr@format@overridetrue
3009      \appto\theindex{\let\glshypernumber\@firstofone}%
3010    }
3011  \else
3012    \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3013      \@glxtr@format@overridetrue
3014      \appto\theindex{\let\glshypernumber\hyperpage}%
3015    }
3016  \fi
3017 }
3018 {
3019   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3020     \@glxtr@format@overridetrue
3021   }
3022 }
3023 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

3024 \newcommand*{\glxtrdoautoindexname}[2]{%
3025   \glshasattribute{#1}{#2}%
3026   {%

```

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.

```

3027   \@glxtr@autoindex@setname{#1}%

```

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

```

3028   \protected@edef\@glxtr@attrval{\glsggetattribute{#1}{#2}}%
3029   \if@glxtr@format@override
3030     \ifdefstring{\@glsnumberformat}{\glsgnumberformat}{}%
3031     {\let\@glxtr@attrval\@glsnumberformat}%
3032   \fi
3033   \ifdefstring{\@glxtr@attrval}{true}%
3034   {%
3035     {\eappto\@glo@name{\@glxtr@autoindex@encap\@glxtr@attrval}}%
3036     \expandafter\index\expandafter{\@glo@name}%
3037   }%
3038   {}%
3039 }

```

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

```

3040 \newcommand*{\@glxtr@autoindex@setname}[1]{%
3041   \def\@glo@name{\string\glssentryname{#1}}%
3042   \glsetentryfield{\@glo@sort}{#1}{sort}%
3043   \@gls@checkmkidxchars\@glo@sort
3044   \@glxtr@autoindex@doextra@esc\@glo@sort
3045   \epreto\@glo@name{\@glo@sort\@glxtr@autoindex@at}%
3046 }

```

dex@doextra@esc

```
3047 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
3048 \ifx\@glsxtr@autoindex@esc\@gls@quotechar
3049 \else
3050 \def\@gls@checkedmkidx{}%
3051 \edef\@glsxtr@checkspch{%
3052 \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
3053 \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
3054 \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3055 \@glsxtr@checkspch
3056 \let#1\@gls@checkedmkidx\relax
3057 \fi
```

Escape actual character unless it has already been escaped.

```
3058 \ifx\@glsxtr@autoindex@at\@gls@actualchar
3059 \else
3060 \def\@gls@checkedmkidx{}%
3061 \edef\@glsxtr@checkspch{%
3062 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
3063 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
3064 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3065 \@glsxtr@checkspch
3066 \let#1\@gls@checkedmkidx\relax
3067 \fi
```

Escape level character unless it has already been escaped.

```
3068 \ifx\@glsxtr@autoindex@level\@gls@levelchar
3069 \else
3070 \def\@gls@checkedmkidx{}%
3071 \edef\@glsxtr@checkspch{%
3072 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
3073 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
3074 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3075 \@glsxtr@checkspch
3076 \let#1\@gls@checkedmkidx\relax
3077 \fi
```

Escape encap character unless it has already been escaped.

```
3078 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
3079 \else
3080 \def\@gls@checkedmkidx{}%
3081 \edef\@glsxtr@checkspch{%
3082 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
3083 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
3084 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
3085 \@glsxtr@checkspch
3086 \let#1\@gls@checkedmkidx\relax
3087 \fi
3088 }
```

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`.

```
3089 \newcommand*{\@glxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```
3090 \newcommand*{\GlsXtrSetActualChar}[1]{%
3091   \gdef\@glxtr@autoindex@at{#1}%
3092   \def\@glxtr@autoindex@escat##1##2##3\@glxtr@endescspch{%
3093     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escat}{##1}{##2}{##3}%
3094   }%
3095 }
3096 \@onlypreamble\GlsXtrSetActualChar
3097 \makeatother
3098 \GlsXtrSetActualChar{@}
3099 \makeatletter

```

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

```
3100 \newcommand*{\@glxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```
3101 \newcommand*{\GlsXtrSetEncapChar}[1]{%
3102   \gdef\@glxtr@autoindex@encap{#1}%
3103   \def\@glxtr@autoindex@escencap##1##2##3\@glxtr@endescspch{%
3104     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
3105   }%
3106 }
3107 \GlsXtrSetEncapChar{|}
3108 \@onlypreamble\GlsXtrSetEncapChar

```

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

```
3109 \newcommand*{\@glxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```
3110 \newcommand*{\GlsXtrSetLevelChar}[1]{%
3111   \gdef\@glxtr@autoindex@level{#1}%
3112   \def\@glxtr@autoindex@esclevel##1##2##3\@glxtr@endescspch{%
3113     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
3114   }%
3115 }
3116 \GlsXtrSetLevelChar{!}
3117 \@onlypreamble\GlsXtrSetLevelChar

```

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

```
3118 \newcommand*{\@glxtr@autoindex@esc}{"}

```

`\GlsXtrSetEscChar` Set the escape character.

```
3119 \newcommand*{\GlsXtrSetEscChar}[1]{%
3120   \gdef\@glstr@autoindex@esc{#1}%
3121   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
3122     \@glstr@autoindex@escspch{#1}\@glstr@autoindex@escquote}{##1}{##2}{##3}%
3123   }%
3124 }
3125 \GlsXtrSetEscChar{"}
3126 \@onlypreamble\GlsXtrSetEscChar
```

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

```
3127 \ifdef\actualchar
3128   {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
3129   {}
```

Quote character `\quotechar`:

```
3130 \ifdef\quotechar
3131   {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
3132   {}
```

Level character `\levelchar`:

```
3133 \ifdef\levelchar
3134   {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
3135   {}
```

Encap character `\encapchar`:

```
3136 \ifdef\encapchar
3137   {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
3138   {}
```

`\leto@endescspch`

```
3139 \def\@glstr@gobbleto@endescspch#1\@glstr@endescspch{}
```

`\toindex@esc@spch`

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

```
3140 \newcommand*{\@glstr@autoindex@escspch}[5]{%
3141   \@glstr@tmpb=\expandafter{\@glstr@checkedmkidx}%
3142   \toks@={#3}%
3143   \ifx\@nnil#3\relax
3144     \def\@glstr@checkspch{\@glstr@gobbleto@endescspch#5\@glstr@endescspch}%
3145   \else
3146     \ifx\@nnil#4\relax
3147       \edef\@glstr@checkedmkidx{\the\@glstr@tmpb\the\toks@}%
3148       \def\@glstr@checkspch{\@glstr@gobbleto@endescspch
3149         #4#5\@glstr@endescspch}%
3150     \else
3151       \edef\@glstr@checkedmkidx{\the\@glstr@tmpb\the\toks@
```

```

3152      \@glsxtr@autoindex@esc#1}%
3153      \def\@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
3154      \fi
3155      \fi
3156      \@glsxtr@checkspch
3157 }

```

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

```

3158 \renewcommand*{\Glossentrydesc}[1]{%
3159   \glsdoifexistsorwarn{#1}%
3160   {%
3161     \glssetabbrvfmt{\glscategory{#1}}%
3162     \Glsaccessdesc{#1}%
3163   }%
3164 }

```

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

```

3165 \renewcommand*{\glossentrysymbol}[1]{%
3166   \glsdoifexistsorwarn{#1}%
3167   {%
3168     \glssetabbrvfmt{\glscategory{#1}}%
3169     \Glsaccesssymbol{#1}%
3170   }%
3171 }

```

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

```

3172 \renewcommand*{\Glossentrysymbol}[1]{%
3173   \glsdoifexistsorwarn{#1}%
3174   {%
3175     \glssetabbrvfmt{\glscategory{#1}}%
3176     \Glsaccesssymbol{#1}%
3177   }%
3178 }

```

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

`\enableInitialTagging` 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.

```

3179 \newcommand*{\GlsXtrEnableInitialTagging}{%
3180   \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
3181 }
3182 \@onlypreamble\GlsXtrEnableInitialTagging

```

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

```

3183 \newcommand*{\s@glsxtr@enabletagging}[2]{%
3184   \undef#2%
3185   \@glsxtr@enabletagging{#1}{#2}%
3186 }

```

r@enabletagging Internal command.

```
3187 \newcommand*{\@glsxtr@enabletagging}[2]{%  
    Set attributes for categories given in the first argument.  
3188   \@for\@glsxtr@cat:=#1\do  
3189   {%  
3190     \ifdefempty\@glsxtr@cat  
3191     {}%  
3192     {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%  
3193   }%  
3194   \newrobustcmd*#2[1]{##1}%  
3195   \def\@glsxtr@taggingcs{#2}%  
3196   \renewcommand*\@glsxtr@activate@initialtagging{%  
3197     \let#2\@glsxtr@tag  
3198   }%  
3199   \ifundef\@gls@preglossaryhook  
3200   {\GlossariesExtraWarning{Initial tagging requires at least  
3201     glossaries.sty v4.19 to work correctly}}%  
3202   {}%  
3203 }
```

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

```
3204 \ifundef\mfu@checkword@do  
3205 {  
3206   \newcommand*{\mfu@checkword@do}[1]{%  
3207     \ifdefstring{\mfu@checkword@arg}{#1}%  
3208     {%  
3209       \let\@mfu@domakefirstuc\@firstofone  
3210       \listbreak  
3211     }%  
3212   }%  
3213 }
```

\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.

```
3214 \ifundef\mfu@checkword  
3215 {  
3216   \newcommand{\@glsxtr@do@titlecaps@warn}{%  
3217     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps  
3218       support not available}}%  
3219   \let\@glsxtr@do@titlecaps@warn\relax  
3220 }  
3221 }  
3222 {  
3223   \renewcommand*{\mfu@checkword}[1]{%
```

One warning should suffice.

```
3219     \let\@glsxtr@do@titlecaps@warn\relax  
3220 }  
3221 }  
3222 {  
3223   \renewcommand*{\mfu@checkword}[1]{%
```



```

3224      \def\mfu@checkword@arg{#1}%
3225      \let\@mfu@domakefirstuc\makefirstuc
3226      \forlistloop\mfu@checkword@do\@mfu@nocaplist
3227    }
3228  }
3229 }
3230 {}% no patch required

```

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

```

3231 \newcommand*{\@glxtr@do@titlecaps@warn}{}

```

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

```

3232 \newcommand*{\@glxtr@activate@initialtagging}{}

```

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

```

3233 \newrobustcmd*{\@glxtr@tag}[1]{%
3234   \gl@ifattribute{\gl@currententrylabel}{tagging}{true}%
3235   {\gl@xtrtagfont{#1}}{#1}%
3236 }

```

`\gl@xtrtagfont` Used in the glossary.

```

3237 \newcommand*{\gl@xtrtagfont}[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.

```

3238 \ifdef\@gl@preglossaryhook
3239 {
3240   \renewcommand*{\@gl@preglossaryhook}{%
3241     \@glxtr@activate@initialtagging
3242     \let\@glxtr@org@postdescription\gl@postdescription
3243     \renewcommand*{\gl@postdescription}{%
3244       \ifgl@entryexists{\gl@currententrylabel}%
3245       {%
3246         \gl@xtrpostdescription
3247         \@glxtr@org@postdescription
3248       }{}%
3249     }%
3250   }%
3251 }
3252 {}

```

`postdescription` This command will only be used if `\@gl@preglossaryhook` is available *and* the glossary style uses `\gl@postdescription` 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.

```

3253 \newcommand*{\glxtrpostdescription}{%
3254   \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
3255 }

postdescgeneral
3256 \newcommand*{\glxtrpostdescgeneral}{}

xtrpostdescterm
3257 \newcommand*{\glxtrpostdescterm}{}

postdescacronym
3258 \newcommand*{\glxtrpostdescacronym}{}

escabbreviation
3259 \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.
3260 \renewcommand*{\glspostlinkhook}{%
3261   \ifglseentryexists{\glslabel}{\glxtrpostlinkhook}{}%
3262 }

xtrpostlinkhook  The entry label should already be stored in \glslabel by \@gls@link.
3263 \newcommand*{\glxtrpostlinkhook}{%
3264   \glxtrdiscardperiod{\glslabel}%
3265   {\glxtrpostlinkendsentence}%
3266   {\glxtrpostlink}%
3267 }

\glxtrpostlink
3268 \newcommand*{\glxtrpostlink}{%
3269   \csuse{glxtrpostlink\glscategory{\glslabel}}%
3270 }

linkendsentence  Done by \glxtrpostlinkhook if a full stop is discarded.
3271 \newcommand*{\glxtrpostlinkendsentence}{%
3272   \ifcsdef{glxtrpostlink\glscategory{\glslabel}}
3273   {%
3274     \csuse{glxtrpostlink\glscategory{\glslabel}}%
    Put the full stop back.
3275     .\spacefactor\sffcode'\. \relax
3276   }%
3277   {%

```

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

```
3278 \spacefactor\sfcode'\. \relax
3279 }%
3280 }
```

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.

```
3281 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
3282 \glxtrifwasfirstuse{\space(\glssaccessdesc{\glslabel})}{}}%
3283 }
```

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.

```
3284 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
3285 \glxtrifwasfirstuse
3286 {%
3287 \ifglshassymbol{\glslabel}{\space(\glssaccesssymbol{\glslabel})}{}}%
3288 }%
3289 {}%
3290 }
```

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).

```
3291 \newcommand*{\glxtrdiscardperiod}[3]{%
3292 \glxtrifwasfirstuse
3293 {%
3294 \glssifattribute{#1}{retainfirstuseperiod}{true}%
3295 {#3}%
3296 {%
3297 \glssifattribute{#1}{discardperiod}{true}%
3298 {%
3299 \glssifplural
3300 {%
3301 \glssifattribute{#1}{pluraldiscardperiod}{true}%
3302 {\glxtrifperiod{#2}{#3}}%
3303 {#3}%
3304 }%
3305 {%
3306 \glxtrifperiod{#2}{#3}%
3307 }%
3308 }%
3309 {#3}%
3310 }%
3311 }%
3312 {%
```

```

3313 \glsifattribute{#1}{discardperiod}{true}%
3314 {%
3315 \glsifplural
3316 {%
3317 \glsifattribute{#1}{pluraldiscardperiod}{true}%
3318 {\glsxtrifperiod{#2}{#3}}%
3319 {#3}%
3320 }%
3321 {%
3322 \glsxtrifperiod{#2}{#3}%
3323 }%
3324 }%
3325 {#3}%
3326 }%
3327 }

```

`\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`

```
3328 \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 ' ').

```
3329 \newcommand*{\glsxtr@punclist}{.,:;?!}
```

`\punctuationmark` Add character to punctuation list.

```
3330 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`\punctuationmarks` Reset the punctuation list.

```
3331 \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`.)

```

3332 \newcommand*{\glsxtrifnextpunc}[2]{%
3333 \def\reserved@a{#1}%
3334 \def\reserved@b{#2}%
3335 \futurelet\@glspunc@token\glsxtr@ifnextpunc
3336 }

```

`\glsxtr@ifnextpunc`

```

3337 \newcommand*{\glsxtr@ifnextpunc}{%
3338 \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{-}%
3339 \reserved@b
3340 }

```

xtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
3341 \newcommand*{\glxtr@ifpunctoken}[1]{%
3342   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
3343 }
```

xtr@ifpunctoken

```
3344 \def\@glxtr@ifpunctoken#1#2{%
3345   \let\reserved@d=#2%
3346   \ifx\reserved@d\@nnil
3347     \let\glxtr@next\@glxtr@notfoundinlist
3348   \else
3349     \ifx#1\reserved@d
3350       \let\glxtr@next\@glxtr@foundinlist
3351     \else
3352       \let\glxtr@next\@glxtr@ifpunctoken
3353     \fi
3354   \fi
3355   \glxtr@next#1%
3356 }
```

xtr@foundinlist

```
3357 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
3358 \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.

```
3359 \newcommand{\glxtrdopostpunc}[1]{%
3360   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
3361 }
```

@glxtr@swaptwo

```
3362 \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.

```

3363 \define@key{glsxtrabbrv}{category}{%
3364 \edef\glscategorylabel{#1}%
3365 \ifcsdef{@glsabbrv@current@#1}%
3366 {%

```

Warning should already have been issued.

```

3367 \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
3368 \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
3369 \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
3370 \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
3371 }%
3372 }%
3373 }

```

Save the short plural form. This may be needed before the entry is defined.

```

3374 \define@key{glsxtrabbrv}{shortplural}{%
3375 \def\@gls@shortpl{#1}%
3376 }

```

Similarly for the long plural form.

```

3377 \define@key{glsxtrabbrv}{longplural}{%
3378 \def\@gls@longpl{#1}%
3379 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```

3380 \newtoks\glsshortpltok

```

\glslongpltok

```

3381 \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.

```

3382 \newcommand*{\@glsxtr@insertdots}[2]{%
3383 \def#1{%
3384 \@glsxtr@insert@dots#1#2\@nnil
3385 }

```

glsxtr@insert@dots

```

3386 \newcommand*{\@glsxtr@insert@dots}[2]{%
3387 \ifx\@nnil#2\relax
3388 \let\@glsxtr@insert@dots@next\@gobble
3389 \else
3390 \ifx\relax#2\relax

```

```

3391 \else
3392 \appto#1{#2.}%
3393 \fi
3394 \let\@glstr@insert@dots@next\@glstr@insert@dots
3395 \fi
3396 \@glstr@insert@dots@next#1%
3397 }

```

newabbreviation Define a new generic abbreviation.

```

3398 \newcommand*\newabbreviation[4][{}]{%
3399 \glskeylisttok{#1}%
3400 \glslabeltok{#2}%
3401 \glsshorttok{#3}%
3402 \glslongtok{#4}%

```

Get the category.

```

3403 \def\glscategorylabel{abbreviation}%
3404 \glstr@applyabbrvstyle{\@glstr@current@abbreviation}%
3405 \setkeys*{glstrabbrv}[shortplural,longplural]{#1}%

```

Set the default long plural

```

3406 \def\@gls@longpl{#4\glspluralsuffix}%

```

Has the insertdots attribute been set?

```

3407 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
3408 {%
3409 \@glstr@insertdots\@gls@short{#3}%
3410 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
3411 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3412 {%
3413 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3414 '\abbrvpluralsuffix}%
3415 }%
3416 {%
3417 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
3418 {%
3419 \let\@gls@shortpl\@gls@short
3420 }%
3421 {%
3422 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3423 \abbrvpluralsuffix}%
3424 }%
3425 }%
3426 }%
3427 {%

```

insertdots not true.

```

3428 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3429 {%
3430 \def\@gls@shortpl{#3'\abbrvpluralsuffix}%
3431 }%

```

```

3432   {%
3433     \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
3434     {%
3435       \def\@gls@shortpl{#3}%
3436     }%
3437     {%
3438       \def\@gls@shortpl{#3\abbrvpluralsuffix}%
3439     }%
3440   }%
3441 }%

```

Hook for further customisation if required:

```

3442 \glsxtrnewabbrvpresetkeyhook{#1}{#2}{#3}%

```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary.

```

3443 \setkeys*{\glsxtrabbrv}[category]{#1}%

```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```

3444 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
3445 \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Do any extra setup provided by hook:

```

3446 \newabbreviationhook

```

Define this entry:

```

3447 \protected@edef\@do@newglossaryentry{%
3448   \noexpand\newglossaryentry{\the\glslabeltok}%
3449   {%
3450     type=\glsxtrabbrvtype,%
3451     category=abbreviation,%
3452     short={\the\glsshorttok},%
3453     shortplural={\the\glsshortpltok},%
3454     long={\the\glslongtok},%
3455     longplural={\the\glslongpltok},%
3456     name={\the\glsshorttok},%
3457     \CustomAbbreviationFields,%
3458     \the\glskeylisttok
3459   }%
3460 }%
3461 \@do@newglossaryentry
3462 \GlsXtrPostNewAbbreviation
3463 }

```

`\evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```

3464 \newcommand*{\glsxtrnewabbrvpresetkeyhook}[3]{}

```

`\NewAbbreviation` Hook used by abbreviation styles.

```

3465 \newcommand*{\GlsXtrPostNewAbbreviation}{}

```

`\bbreviationhook` Hook for use with `\newabbreviation`.

```

3466 \newcommand*{\newabbreviationhook}{}

```


reviousFields

```
3467 \newcommand*{\CustomAbbreviationFields}{}
```

lsxtrfullformat Full format without case change.

```
3468 \newcommand*{\glxtrfullformat}[2]{%
3469   \glsfirstlongfont{\glsaccesslong{#1}}#2\glxtrfullsep{#1}%
3470   (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
3471 }
```

lsxtrfullformat Full format with case change.

```
3472 \newcommand*{\Glsxtrfullformat}[2]{%
3473   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glxtrfullsep{#1}%
3474   (\protect\glsfirstabbrvfont{\Glsaccessshort{#1}})%
3475 }
```

xtrfullplformat Plural full format without case change.

```
3476 \newcommand*{\glxtrfullplformat}[2]{%
3477   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
3478   (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
3479 }
```

xtrfullplformat Plural full format with case change.

```
3480 \newcommand*{\Glsxtrfullplformat}[2]{%
3481   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
3482   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
3483 }
```

\glxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
3484 \newcommand*{\glxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with \gl Sentryfull (for example, first use suppresses the long form or uses a footnote).

inlinefullformat Full format without case change.

```
3485 \newcommand*{\glxtrininlinefullformat}{\glxtrfullformat}
```

inlinefullformat Full format with case change.

```
3486 \newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
3487 \newcommand*{\glxtrininlinefullplformat}{\glxtrfullplformat}
```

inefullplformat Plural full format with case change.

```
3488 \newcommand*{\Glsxtrininlinefullplformat}{\Glsxtrfullplformat}
```

Redefine \gl Sentryfull 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`
3489 `\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}`

`\Glsentryfull`
3490 `\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}`

`\glsentryfullpl`
3491 `\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}`

`\Glsentryfullpl`
3492 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
3493 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
3494 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.
3495 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`
3496 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
3497 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.
3498 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.
3499 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`
3500 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix.
3501 `\newcommand*{\abbrvpluralsuffix}{\glspluralsuffix}`

`\glsxtrfull` Full form (no case-change).
3502 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`
3503 `\newcommand*{\ns@glsxtrfull}[2][{}]{%`
3504 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%`
3505 `{\@glsxtr@full{#1}{#2}[]}%`
3506 `}`

\@glxtr@full Low-level macro:

```
3507 \def\@glxtr@full#1#2[#3]{%
3508   \glsdoifexists{#2}%
3509   {%
3510     \glssetabbrvfmt{\glscategory{#2}}%
3511     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3512     \let\glsifplural\@secondoftwo
3513     \let\glscapscase\@firstofthree
3514     \let\glsinsert\@empty
3515     \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.

```
3516   \glxtrsetupfulldefs
3517   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3518   }%
3519   \glspostlinkhook
3520 }
```

trsetupfulldefs

```
3521 \newcommand*{\glxtrsetupfulldefs}{%
3522   \let\glxtrifwasfirstuse\@firstoftwo
3523 }
```

\Glsxtrfull Full form (first letter uppercase).

```
3524 \newrobustcmd*{\Glsxtrfull}{\@gl@hyp@opt\@ns@Glsxtrfull}
3525 \newcommand*\ns@Glsxtrfull[2][ ]{%
3526   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
3527   {\@Glsxtr@full{#1}{#2} [ ]}%
3528 }
```

\@Glsxtr@full Low-level macro:

```
3529 \def\@Glsxtr@full#1#2[#3]{%
3530   \glsdoifexists{#2}%
3531   {%
3532     \glssetabbrvfmt{\glscategory{#2}}%
3533     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3534     \let\glsifplural\@secondoftwo
3535     \let\glscapscase\@secondofthree
3536     \let\glsinsert\@empty
3537     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
3538     \glxtrsetupfulldefs
3539     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3540   }%
3541   \glspostlinkhook
3542 }
```

`\GLSxtrfull` Full form (all uppercase).

```
3543 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
3544 \newcommand*\ns@GLSxtrfull[2][\%
3545   \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{%
3546     {\@GLSxtr@full{#1}{#2}[]}%
3547 }
```

`\@GLSxtr@full` Low-level macro:

```
3548 \def\@GLSxtr@full#1#2[#3]{%
3549   \glsdoifexists{#2}%
3550   {%
3551     \glssetabbrvfmt{\glscategory{#2}}%
3552     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3553     \let\glsifplural\@secondoftwo
3554     \let\glsapscase\@thirdofthree
3555     \let\glsinsert\@empty
3556     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
3557     \glsxtrsetupfulldefs
3558     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3559   }%
3560   \glspostlinkhook
3561 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
3562 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
3563 \newcommand*\ns@glsxtrfullpl[2][\%
3564   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{%
3565     {\@glsxtr@fullpl{#1}{#2}[]}%
3566 }
```

`\@glsxtr@fullpl` Low-level macro:

```
3567 \def\@glsxtr@fullpl#1#2[#3]{%
3568   \glsdoifexists{#2}%
3569   {%
3570     \glssetabbrvfmt{\glscategory{#2}}%
3571     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3572     \let\glsifplural\@firstoftwo
3573     \let\glsapscase\@firstofthree
3574     \let\glsinsert\@empty
3575     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
3576     \glsxtrsetupfulldefs
3577     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3578   }%
3579   \glspostlinkhook
3580 }
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
3581 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
3582 \newcommand*\ns@Glsxtrfullpl[2][\%
```

```

3583 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
3584           {\@Glsxtr@fullpl{#1}{#2}[]}%
3585 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

3586 \def\@Glsxtr@fullpl#1#2[#3]{%
3587   \glsdoifexists{#2}%
3588   {%
3589     \glssetabbrvfmt{\glscategory{#2}}%
3590     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3591     \let\glsifplural\@firstoftwo
3592     \let\glsupcase\@secondofthree
3593     \let\glsinsert\@empty
3594     \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
3595     \glsxtrsetupfulldefs
3596     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3597   }%
3598   \glspostlinkhook
3599 }

```

`\Glsxtrfullpl` Plural full form (all upper case).

```

3600 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
3601 \newcommand*\ns@Glsxtrfullpl[2][]{%
3602   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
3603   {\@Glsxtr@fullpl{#1}{#2}[]}%
3604 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

3605 \def\@Glsxtr@fullpl#1#2[#3]{%
3606   \glsdoifexists{#2}%
3607   {%
3608     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3609     \let\glsifplural\@firstoftwo
3610     \let\glsupcase\@thirdofthree
3611     \let\glsinsert\@empty
3612     \def\glscustomtext{%
3613       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
3614     \glsxtrsetupfulldefs
3615     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3616   }%
3617   \glspostlinkhook
3618 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

3619 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

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

```

3620 \newcommand*{\ns@glstrshort}[2] [] {%
3621   \new@ifnextchar[{\@glstrshort{#1}{#2}}{\@glstrshort{#1}{#2} []}%
3622 }

```

Read in the final optional argument:

```

3623 \def\@glstrshort#1#2[#3] {%
3624   \glstoifexists{#2}%
3625   {%

```

Need to make sure \glabbrvfont is set correctly.

```

3626     \glsetabbrvfmt{\glscategory{#2}}%
3627     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3628     \let\glstrifwasfirstuse\@secondoftwo
3629     \let\gl@ifplural\@secondoftwo
3630     \let\glscapscase\@firstofthree
3631     \let\glinsert\@empty
3632     \def\glscustomtext{%
3633       \glabbrvfont{\glaccessshort{#2}\ifglstrinsertinside#3\fi}%
3634       \ifglstrinsertinside\else#3\fi
3635     }%
3636     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3637   }%
3638   \glspostlinkhook
3639 }

```

\Glsxtrshort

```

3640 \newrobustcmd*{\Glsxtrshort}{\@gl@hyp@opt\ns@Glsxtrshort}

```

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

```

3641 \newcommand*{\ns@Glsxtrshort}[2] [] {%
3642   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
3643 }

```

Read in the final optional argument:

```

3644 \def\@Glsxtrshort#1#2[#3] {%
3645   \glstoifexists{#2}%
3646   {%
3647     \glsetabbrvfmt{\glscategory{#2}}%
3648     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3649     \let\glstrifwasfirstuse\@secondoftwo
3650     \let\gl@ifplural\@secondoftwo
3651     \let\glscapscase\@secondofthree
3652     \let\glinsert\@empty
3653     \def\glscustomtext{%
3654       \glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
3655       \ifglstrinsertinside\else#3\fi
3656     }%
3657     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3658   }%
3659   \glspostlinkhook
3660 }

```

\GLSxtrshort

```
3661 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
3662 \newcommand*{\ns@GLSxtrshort}[2] [] {%
3663   \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} []}]%
3664 }

    Read in the final optional argument:
3665 \def\@GLSxtrshort#1#2[#3] {%
3666   \glsdoifexists{#2}%
3667   {%
3668     \glssetabbrvfmt{\glscategory{#2}}%
3669     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3670     \let\glsxtrifwasfirstuse\@secondoftwo
3671     \let\glsifplural\@secondoftwo
3672     \let\glsapscase\@thirdofthree
3673     \let\glsinsert\@empty
3674     \def\glscustomtext{%
3675       \mfirstucMakeUppercase
3676       {\glsabbrvfont{\glsaccessshort{#2}}\ifglsxtrininsertinside#3\fi}%
3677       \ifglsxtrininsertinside\else#3\fi
3678     }%
3679   }%
3680   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3681   }%
3682   \glspostlinkhook
3683 }
```

\glsxtrlong

```
3684 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\ns@glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
3685 \newcommand*{\ns@glsxtrlong}[2] [] {%
3686   \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2} []}]%
3687 }

    Read in the final optional argument:
3688 \def\@glsxtrlong#1#2[#3] {%
3689   \glsdoifexists{#2}%
3690   {%
3691     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3692     \let\glsxtrifwasfirstuse\@secondoftwo
3693     \let\glsifplural\@secondoftwo
3694     \let\glsapscase\@firstofthree
3695     \let\glsinsert\@empty
3696     \def\glscustomtext{%
3697       \glsfont{\glsaccesslong{#2}}\ifglsxtrininsertinside#3\fi}%
3698       \ifglsxtrininsertinside\else#3\fi
3699     }%
3700     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

3701 }%
3702 \glspostlinkhook
3703 }

```

\Glsxtrlong

```

3704 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
3705 \newcommand*{\ns@Glsxtrlong}[2] [] {%
3706   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
3707 }

    Read in the final optional argument:
3708 \def\@Glsxtrlong#1#2[#3]{%
3709   \glsdoifexists{#2}%
3710   {%
3711     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3712     \let\glstrifwasfirstuse\@secondoftwo
3713     \let\glsifplural\@secondoftwo
3714     \let\glscapscase\@thirdofthree
3715     \let\glsinsert\@empty
3716     \def\glscustomtext{%
3717       \glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
3718       \ifglstrinsertinside\else#3\fi
3719     }%
3720     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3721   }%
3722   \glspostlinkhook
3723 }

```

\GLSxtrlong

```

3724 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
3725 \newcommand*{\ns@GLSxtrlong}[2] [] {%
3726   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
3727 }

    Read in the final optional argument:
3728 \def\@GLSxtrlong#1#2[#3]{%
3729   \glsdoifexists{#2}%
3730   {%
3731     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3732     \let\glstrifwasfirstuse\@secondoftwo
3733     \let\glsifplural\@secondoftwo
3734     \let\glscapscase\@thirdofthree
3735     \let\glsinsert\@empty
3736     \def\glscustomtext{%
3737       \mfirstucMakeUppercase
3738       {\glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
3739       \ifglstrinsertinside\else#3\fi

```



```

3740     }%
3741     }%
3742     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3743     }%
3744     \glspostlinkhook
3745 }

```

Plural short forms:

\glsxtrshortpl

```

3746 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

```

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

```

3747 \newcommand*{\ns@glsxtrshortpl}[2][ ]{%
3748   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2}[]}%
3749 }

```

Read in the final optional argument:

```

3750 \def\@glsxtrshortpl#1#2[#3]{%
3751   \glsdoifexists{#2}%
3752   {%
3753     \glssetabbrvfmt{\glscategory{#2}}%
3754     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3755     \let\glsxtrifwasfirstuse\@secondoftwo
3756     \let\glsifplural\@firstoftwo
3757     \let\glscapscase\@firstofthree
3758     \let\glsinsert\@empty
3759     \def\glscustomtext{%
3760       \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
3761       \ifglsxtrininsertinside\else#3\fi
3762     }%
3763     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3764     }%
3765     \glspostlinkhook
3766 }

```

\Glsxtrshortpl

```

3767 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

3768 \newcommand*{\ns@Glsxtrshortpl}[2][ ]{%
3769   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
3770 }

```

Read in the final optional argument:

```

3771 \def\@Glsxtrshortpl#1#2[#3]{%
3772   \glsdoifexists{#2}%
3773   {%
3774     \glssetabbrvfmt{\glscategory{#2}}%
3775     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3776     \let\glsxtrifwasfirstuse\@secondoftwo

```

```

3777 \let\glsifplural\@firstoftwo
3778 \let\glscapscase\@secondofthree
3779 \let\glsinsert\@empty
3780 \def\glscustomtext{%
3781   \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
3782   \ifglsxtrinsertinside\else#3\fi
3783 }%
3784 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3785 }%
3786 \glspostlinkhook
3787 }

```

\GLSxtrshortpl

```

3788 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3789 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
3790   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
3791 }

```

Read in the final optional argument:

```

3792 \def\@GLSxtrshortpl#1#2[#3] {%
3793   \glsdoifexists{#2}%
3794   {%
3795     \glssetabbrvfmt{\glscategory{#2}}%
3796     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3797     \let\glsxtrifwasfirstuse\@secondoftwo
3798     \let\glsifplural\@firstoftwo
3799     \let\glsapscale\@thirdofthree
3800     \let\glsinsert\@empty
3801     \def\glscustomtext{%
3802       \mfirstucMakeUppercase
3803       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
3804       \ifglsxtrinsertinside\else#3\fi
3805     }%
3806   }%
3807   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3808 }%
3809 \glspostlinkhook
3810 }

```

Plural long forms:

\glsxtrlongpl

```

3811 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3812 \newcommand*{\ns@glsxtrlongpl}[2] [] {%
3813   \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%
3814 }

```

Read in the final optional argument:

```
3815 \def\@glxstrlongpl#1#2[#3]{%
3816   \glstoifexists{#2}%
3817   {%
3818     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3819     \let\glxstrifwasfirstuse\@secondoftwo
3820     \let\gl@ifplural\@firstoftwo
3821     \let\glscapscase\@firstofthree
3822     \let\glinsert\@empty
3823     \def\glscustomtext{%
3824       \glslongfont{\glaccesslongpl{#2}\ifglxstrinsertinside#3\fi}%
3825       \ifglxstrinsertinside\else#3\fi
3826     }%
3827     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3828   }%
3829   \glspostlinkhook
3830 }
```

\Glsxtrlongpl

```
3831 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3832 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
3833   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
3834 }
```

Read in the final optional argument:

```
3835 \def\@Glsxtrlongpl#1#2[#3]{%
3836   \glstoifexists{#2}%
3837   {%
3838     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3839     \let\glxstrifwasfirstuse\@secondoftwo
3840     \let\gl@ifplural\@firstoftwo
3841     \let\glscapscase\@secondofthree
3842     \let\glinsert\@empty
3843     \def\glscustomtext{%
3844       \glslongfont{\Glsaccesslongpl{#2}\ifglxstrinsertinside#3\fi}%
3845       \ifglxstrinsertinside\else#3\fi
3846     }%
3847     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3848   }%
3849   \glspostlinkhook
3850 }
```

\GLSxtrlongpl

```
3851 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
3852 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
3853   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
3854 }
```

Read in the final optional argument:

```

3855 \def\@GLSxtrlongpl#1#2[#3]{%
3856   \glsdoifexists{#2}%
3857   {%
3858     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3859     \let\glxtrifwasfirsttuse\@secondoftwo
3860     \let\glsifplural\@firstoftwo
3861     \let\glscapscase\@thirdofthree
3862     \let\glsinsert\@empty
3863     \def\glscustomtext{%
3864       \mfirstucMakeUppercase
3865       {\glslongfont{\glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
3866       \ifglxtrininsertinside\else#3\fi
3867     }%
3868   }%
3869   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3870 }%
3871 \glspostlinkhook
3872 }

```

`\glssetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```

3873 \newcommand*{\glssetabbrvfmt}[1]{%
3874   \ifcsdef{@glsabbrv@current@#1}%
3875   {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
3876   {\glxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
3877 }

```

`sxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```

3878 \newcommand*{\glsxtrgenabbrvfmt}{%
3879   \ifdefempty\glscustomtext
3880   {%
3881     \ifgl@sused\glslabel
3882     {%

```

Subsequent use:

```

3883     \glsifplural
3884     {%

```

Subsequent plural form:

```

3885     \glscapscase
3886     {%

```

Subsequent plural form, don't adjust case:

```

3887     \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
3888     }%
3889     {%

```

Subsequent plural form, make first letter upper case:

```

3890     \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
3891     }%
3892     {%

```

Subsequent plural form, all caps:

```
3893      \mfirstucMakeUppercase
3894      {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
3895      }%
3896      }%
3897      {%
```

Subsequent singular form

```
3898      \glscapscase
3899      {%
```

Subsequent singular form, don't adjust case:

```
3900      \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
3901      }%
3902      {%
```

Subsequent singular form, make first letter upper case:

```
3903      \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
3904      }%
3905      {%
```

Subsequent singular form, all caps:

```
3906      \mfirstucMakeUppercase
3907      {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
3908      }%
3909      }%
3910      }%
3911      {%
```

First use:

```
3912      \glsifplural
3913      {%
```

First use plural form:

```
3914      \glscapscase
3915      {%
```

First use plural form, don't adjust case:

```
3916      \glstrfullplformat{\glslabel}{\glsinsert}%
3917      }%
3918      {%
```

First use plural form, make first letter upper case:

```
3919      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
3920      }%
3921      {%
```

First use plural form, all caps:

```
3922      \mfirstucMakeUppercase
3923      {\glstrfullplformat{\glslabel}{\glsinsert}}%
3924      }%
3925      }%
3926      {%
```

First use singular form

```
3927      \glscapscase
3928      {%
```

First use singular form, don't adjust case:

```
3929      \glsxtrfullformat{\glslabel}{\glsinsert}%
3930      }%
3931      {%
```

First use singular form, make first letter upper case:

```
3932      \Glsxtrfullformat{\glslabel}{\glsinsert}%
3933      }%
3934      {%
```

First use singular form, all caps:

```
3935      \mfirstucMakeUppercase
3936      {\glsxtrfullformat{\glslabel}{\glsinsert}}%
3937      }%
3938      }%
3939      }%
3940      }%
3941      {%
```

User supplied text.

```
3942      \glscustomtext
3943      }%
3944 }
```

1.6.1 Abbreviation Styles Setup

breivationstyle

```
3945 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
3946   \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
3947   {%
3948     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
3949   }%
3950   {%
```

Have abbreviations already been defined for this category?

```
3951   \ifcsstring{@glsabbrv@current@#1}{#2}%
3952   {%
```

Style already set.

```
3953   }%
3954   {%
3955     \def\@glsxtr@dostylewarn{%
3956       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
3957       {%
3958         \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
3959           style has been switched \MessageBreak
3960           for category ‘#1’, \MessageBreak
```

```

3961         but there have already been entries \MessageBreak
3962         defined for this category. Unwanted \MessageBreak
3963         side-effects may result}}}%
3964     \@endfortrue
3965 }%
3966 \@glxtr@dostylewarn

Set up the style for the given category.
3967     \csdef{@glxabbrv@current@#1}{#2}%
3968     \glxtr@applyabbrvstyle{#2}%
3969 }%
3970 }%
3971 }

```

`\glxtr@applyabbrvstyle` Apply the abbreviation style without existence check.

```

3972 \newcommand*{\glxtr@applyabbrvstyle}[1]{%
3973     \csuse{@glxabbrv@dispstyle@setup@#1}%
3974     \csuse{@glxabbrv@dispstyle@fmts@#1}%
3975 }

```

`\glxtr@applyabbrvfmt` Only apply the style formats.

```

3976 \newcommand*{\glxtr@applyabbrvfmt}[1]{%
3977     \csuse{@glxabbrv@dispstyle@fmts@#1}%
3978 }

```

`\glxtr@newabbreviationstyle` 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.

```

3979 \newcommand*{\newabbreviationstyle}[3]{%
3980     \ifcsdef{@glxabbrv@dispstyle@setup@#1}
3981     {%
3982         \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
3983         defined}}}%
3984     }%
3985     {%
3986         \csdef{@glxabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

3987     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
3988     #2}%
3989     \csdef{@glxabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

3990     \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
3991     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
3992     \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
3993     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
3994     #3}%
3995 }%
3996 }

```

renewabbreviationstyle

```

3997 \newcommand*{\renewabbreviationstyle}[3]{%
3998   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
3999   {%
4000     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
4001   }%
4002   {%
4003     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
4004       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
4005       #2}%
4006     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
4007       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
4008       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
4009       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
4010       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
4011       #3}%
4012     }%
4013 }

```

renewabbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

4014 \newcommand*{\letabbreviationstyle}[2]{%
4015   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
4016   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4017 }

```

renewdeprecated@abbrstyle

```
\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

4018 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
4019   \csdef{@glsabbrv@dispstyle@setup@#1}{%
4020     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
4021     \csuse{@glsabbrv@dispstyle@setup@#2}%
4022   }%
4023   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4024 }

```

renewdeprecatedAbbrStyle Generate warning for deprecated style use.

```

4025 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
4026   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
4027   use ‘#2’ instead}%
4028 }

```


seAbbrStyleSetup

```

4029 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
4030   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
4031   {%
4032     \PackageError{glossaries-extra}%
4033     {Unknown abbreviation style definitions ‘#1’}{}%
4034   }%
4035   {%
4036     \csname @glsabbrv@dispstyle@setup@#1\endcsname
4037   }%
4038 }

```

seAbbrStyleFmts

```

4039 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
4040   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
4041   {%
4042     \PackageError{glossaries-extra}%
4043     {Unknown abbreviation style formats ‘#1’}{}%
4044   }%
4045   {%
4046     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
4047   }%
4048 }

```

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.

```

4049 \newif\ifglsxtrinsertinside
4050 \glsxtrinsertinsidefalse

```

long-short

```

4051 \newabbreviationstyle{long-short}%
4052 {%
4053   \renewcommand*{\CustomAbbreviationFields}{%
4054     name={\protect\glsabbrvfont{\the\glsshorttok}},
4055     sort={\the\glsshorttok},
4056     first={\protect\glsfirstlongfont{\the\glslongtok}}%
4057     \protect\glsxtrfullsep{\the\glslabeltok}}%
4058     (\protect\glsfirstabbrvfont{\the\glsshorttok}}),%

```

```

4059   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
4060   \protect\glxtrfullsep{\the\glslabeltok}%
4061   (\protect\glsfirstabbrvfont{\the\glsshortpltok}}),%
4062   plural={\protect\glsabbvfont{\the\glsshortpltok}},%
4063   description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

4064   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4065     \glshasattribute{\the\glslabeltok}{regular}%
4066     {%
4067       \glssetAttribute{\the\glslabeltok}{regular}{false}%
4068     }%
4069   }%
4070 }%
4071 }%
4072 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4073   \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4074   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4075   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4076   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4077   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

4078   \renewcommand*{\glxtrfullformat}[2]{%
4079     \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
4080     \ifglxtrininsertinside\else##2\fi
4081     \glxtrfullsep{##1}%
4082     (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4083   }%
4084   \renewcommand*{\glxtrfullplformat}[2]{%
4085     \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
4086     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
4087     (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4088   }%
4089   \renewcommand*{\Glsxtrfullformat}[2]{%
4090     \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
4091     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
4092     (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4093   }%
4094   \renewcommand*{\Glsxtrfullplformat}[2]{%
4095     \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
4096     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
4097     (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4098   }%
4099 }

```

Set this as the default style for general abbreviations:

```

4100 \setabbreviationstyle{long-short}

```

ngshortdescsort

```
4101 \newcommand*{\glxtrlongshortdescsort}{\the\glslongtok\space(\the\glsshorttok)}
```

long-short-desc User supplies description. The long form is included in the name.

```
4102 \newabbreviationstyle{long-short-desc}%
4103 {%
4104   \renewcommand*{\CustomAbbreviationFields}{%
4105     name={\protect\glxtrfullformat{\the\glslabeltok}{}},
4106     sort={\glxtrlongshortdescsort},%
4107     first={\protect\glsfirstlongfont{\the\glslongtok}%
4108       \protect\glxtrfullsep{\the\glslabeltok}%
4109       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4110     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4111       \protect\glxtrfullsep{\the\glslabeltok}%
4112       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
4113     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4114   }%
4115   \glshasattribute{\the\glslabeltok}{regular}%
4116   {%
4117     \glissetattribute{\the\glslabeltok}{regular}{false}%
4118   }%
4119 }%
4120 }%
4121 }%
4122 {%
4123   \GlsXtrUseAbbrStyleFmts{long-short}%
4124 }
```

Unset the regular attribute if it has been set.

```
4114 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4115   \glshasattribute{\the\glslabeltok}{regular}%
4116   {%
4117     \glissetattribute{\the\glslabeltok}{regular}{false}%
4118   }%
4119 }%
4120 }%
4121 }%
4122 {%
4123   \GlsXtrUseAbbrStyleFmts{long-short}%
4124 }
```

short-long Short form followed by long form in parenthesis on first use.

```
4125 \newabbreviationstyle{short-long}%
4126 {%
4127   \renewcommand*{\CustomAbbreviationFields}{%
4128     name={\protect\glsabbvfont{\the\glsshorttok}},
4129     sort={\the\glsshorttok},
4130     description={\the\glslongtok},%
4131     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
4132       \protect\glxtrfullsep{\the\glslabeltok}%
4133       (\protect\glsfirstlongfont{\the\glslongtok})},%
4134     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
4135       \protect\glxtrfullsep{\the\glslabeltok}%
4136       (\protect\glsfirstlongfont{\the\glslongpltok})},%
4137     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4138   }%
4139   \glshasattribute{\the\glslabeltok}{regular}%
4140   {%
4141     \glissetattribute{\the\glslabeltok}{regular}{false}%
4142   }%
4143 }
```

Unset the regular attribute if it has been set.

```
4138 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4139   \glshasattribute{\the\glslabeltok}{regular}%
4140   {%
4141     \glissetattribute{\the\glslabeltok}{regular}{false}%
4142   }%
4143 }
```

```

4142 }%
4143 {}%
4144 }%
4145 }%
4146 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4147 \renewcommand*\abbrvpluralsuffix{\glspluralsuffix}%
4148 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
4149 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
4150 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
4151 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

4152 \renewcommand*\glsxtrfullformat[2]{%
4153   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4154   \ifglsxtrininsertinside\else##2\fi
4155   \glsxtrfullsep{##1}%
4156   (\glsfirstlongfont{\glsaccesslong{##1}})%
4157 }%
4158 \renewcommand*\glsxtrfullplformat[2]{%
4159   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4160   \ifglsxtrininsertinside\else##2\fi
4161   \glsxtrfullsep{##1}%
4162   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4163 }%
4164 \renewcommand*\Glsxtrfullformat[2]{%
4165   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4166   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4167   (\glsfirstlongfont{\glsaccesslong{##1}})%
4168 }%
4169 \renewcommand*\Glsxtrfullplformat[2]{%
4170   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4171   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4172   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4173 }%
4174 }

```

`short-long-desc` User supplies description. The long form is included in the name.

```

4175 \newabbreviationstyle{short-long-desc}%
4176 {%
4177   \renewcommand*\CustomAbbreviationFields{%
4178     name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
4179     sort={\the\glsshorttok},%
4180     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
4181     \protect\glsxtrfullsep{\the\glslabeltok}%
4182     (\protect\glsfirstlongfont{\the\glslongtok}}),%
4183     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
4184     \protect\glsxtrfullsep{\the\glslabeltok}%
4185     (\protect\glsfirstlongfont{\the\glslongpltok}}),%

```

4186 plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

Unset the regular attribute if it has been set.

```
4187 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4188   \glshasattribute{\the\glslabeltok}{regular}%
4189   {%
4190     \glssetattribute{\the\glslabeltok}{regular}{false}%
4191   }%
4192   {}%
4193 }%
4194 }%
4195 {%
4196   \GlsXtrUseAbbrStyleFmts{short-long}%
4197 }
```

ongfootnotefont Only used by the “footnote” styles.

```
4198 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
4199 \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?)

```
4200 \newabbreviationstyle{footnote}%
4201 {%
4202   \renewcommand*{\CustomAbbreviationFields}{%
4203     name={\protect\glsabbvfont{\the\glsshortttok}},
4204     sort={\the\glsshortttok},
4205     description={\the\glslongtok},%
4206     first={\protect\glsfirstabbvfont{\the\glsshortttok}%
4207       \protect\footnote{\protect\glsfirstlongfont{\the\glslongtok}}},%
4208     firstplural={\protect\glsfirstabbvfont{\the\glsshortpltok}%
4209       \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%
4210     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
4211 }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
4211 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4212   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
4213   \glshasattribute{\the\glslabeltok}{regular}%
4214   {%
4215     \glssetattribute{\the\glslabeltok}{regular}{false}%
4216   }%
4217   {}%
4218 }%
4219 }%
4220 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

4221 \renewcommand*\abbrvpluralsuffix{\glspluralsuffix}%
4222 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
4223 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
4224 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
4225 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

4226 \renewcommand*\glsxtrfullformat[2]{%
4227   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4228   \ifglsxtrininsertinside\else##2\fi
4229   \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
4230 }%
4231 \renewcommand*\glsxtrfullplformat[2]{%
4232   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4233   \ifglsxtrininsertinside\else##2\fi
4234   \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
4235 }%
4236 \renewcommand*\Glsxtrfullformat[2]{%
4237   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4238   \ifglsxtrininsertinside\else##2\fi
4239   \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
4240 }%
4241 \renewcommand*\Glsxtrfullplformat[2]{%
4242   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4243   \ifglsxtrininsertinside\else##2\fi
4244   \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
4245 }%

```

The first use full form and the inline full form use the short (long) style.

```

4246 \renewcommand*\glsxtrinlinefullformat[2]{%
4247   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4248   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4249   (\glsfirstlongfont{\glsaccesslong{##1}})%
4250 }%
4251 \renewcommand*\glsxtrinlinefullplformat[2]{%
4252   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4253   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4254   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4255 }%
4256 \renewcommand*\Glsxtrinlinefullformat[2]{%
4257   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4258   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4259   (\glsfirstlongfont{\glsaccesslong{##1}})%
4260 }%
4261 \renewcommand*\Glsxtrinlinefullplformat[2]{%
4262   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4263   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4264   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4265 }%

```

4266 }

short-footnote

```
4267 \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.

```
4268 \newabbreviationstyle{postfootnote}%
4269 {%
4270   \renewcommand*{\CustomAbbreviationFields}{%
4271     name={\protect\glsabbrvfont{\the\glsshorttok}},
4272     sort={\the\glsshorttok},
4273     description={\the\glslongtok},%
4274     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
4275       \protect\footnote{\protect\glsfirstlongfont{\the\glslongtok}}},%
4276     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
4277       \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%
4278     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
4279   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4280     \csdef{glsxtrpostlink\glscategorylabel}{%
4281       \glsxtrifwasfirstuse
4282       {%
4283         \glsxtrdopostpunc{\protect\footnote
4284           {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
4285       }%
4286     }%
4287   }%
4288   \glshasattribute{\the\glslabeltok}{regular}%
4289   {%
4290     \glssetattribute{\the\glslabeltok}{regular}{false}%
4291   }%
4292   {%
4293   }%
```

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.

```
4279   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4280     \csdef{glsxtrpostlink\glscategorylabel}{%
4281       \glsxtrifwasfirstuse
4282       {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
4283         \glsxtrdopostpunc{\protect\footnote
4284           {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
4285       }%
4286     }%
4287   }%
4288   \glshasattribute{\the\glslabeltok}{regular}%
4289   {%
4290     \glssetattribute{\the\glslabeltok}{regular}{false}%
4291   }%
4292   {%
4293   }%
```

The footnote needs to be suppressed in the inline form, so \glsxtrfull must set the first use switch off.

```
4294   \renewcommand*{\glsxtrsetupfulldefs}{%
4295     \let\glsxtrifwasfirstuse\@secondoftwo
4296   }%
4297 }%
4298 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

4299 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4300 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4301 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4302 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
4303 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

4304 \renewcommand*{\glsxtrfullformat}[2]{%
4305   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4306   \ifglsxtrininsertinside\else##2\fi
4307 }%
4308 \renewcommand*{\glsxtrfullplformat}[2]{%
4309   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4310   \ifglsxtrininsertinside\else##2\fi
4311 }%
4312 \renewcommand*{\Glsxtrfullformat}[2]{%
4313   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4314   \ifglsxtrininsertinside\else##2\fi
4315 }%
4316 \renewcommand*{\Glsxtrfullplformat}[2]{%
4317   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4318   \ifglsxtrininsertinside\else##2\fi
4319 }%

```

The first use full form and the inline full form use the short (long) style.

```

4320 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4321   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4322   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4323   (\glsfirstlongfont{\glsaccesslong{##1}})%
4324 }%
4325 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4326   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4327   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4328   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4329 }%
4330 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4331   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4332   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4333   (\glsfirstlongfont{\glsaccesslong{##1}})%
4334 }%
4335 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4336   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4337   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4338   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4339 }%
4340 }

```

rt-postfootnote

```

4341 \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.

```

4342 \newabbreviationstyle{short}%
4343 {%
4344   \renewcommand*{\CustomAbbreviationFields}{%
4345     name={\protect\glsabbrvfont{\the\glsshorttok}},
4346     sort={\the\glsshorttok},
4347     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4348     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4349     text={\protect\glsabbrvfont{\the\glsshorttok}},
4350     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4351     description={\the\glslongtok}}%
4352   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4353     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4354 }%
4355 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4356 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4357 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4358 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4359 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4360 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

4361 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4362   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4363   \ifglsxtrininsertinside##2\fi}%
4364   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4365   (\glsfirstlongfont{\glsaccesslong{##1}})%
4366 }%
4367 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4368   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4369   \ifglsxtrininsertinside##2\fi}%
4370   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4371   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4372 }%
4373 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4374   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4375   \ifglsxtrininsertinside##2\fi}%
4376   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4377   (\glsfirstlongfont{\Glsaccesslong{##1}})%
4378 }%
4379 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4380   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4381   \ifglsxtrininsertinside##2\fi}%
4382   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4383   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%

```

4384 }%

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.

```
4385 \renewcommand*{\glxtrfullformat}[2]{%
4386   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4387   \ifglxtrinsertinside\else##2\fi
4388 }%
4389 \renewcommand*{\glxtrfullplformat}[2]{%
4390   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4391   \ifglxtrinsertinside\else##2\fi
4392 }%
4393 \renewcommand*{\Glsxtrfullformat}[2]{%
4394   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4395   \ifglxtrinsertinside\else##2\fi
4396 }%
4397 \renewcommand*{\Glsxtrfullplformat}[2]{%
4398   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4399   \ifglxtrinsertinside\else##2\fi
4400 }%
4401 }
```

Set this as the default style for acronyms:

```
4402 \setabbreviationstyle[acronym]{short}
```

short-nolong

```
4403 \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.

```
4404 \newabbreviationstyle{short-desc}%
4405 {%
4406   \renewcommand*{\CustomAbbreviationFields}{%
4407     name={\protect\glxtrinlinefullformat{\the\glslabeltok}{}},
4408     sort={\the\glsshorttok},
4409     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4410     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4411     text={\protect\glsabbrvfont{\the\glsshorttok}},
4412     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4413     description={\the\glslongtok}}%
4414   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4415     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4416 }%
4417 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4418 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4419 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4420 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
```

```

4421 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4422 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

4423 \renewcommand*\glsxtrinlinefullformat}[2]{%
4424   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4425   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4426   (\glsfirstlongfont{\glsaccesslong{##1}})%
4427 }%
4428 \renewcommand*\glsxtrinlinefullplformat}[2]{%
4429   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4430   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4431   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4432 }%
4433 \renewcommand*\Glsxtrinlinefullformat}[2]{%
4434   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4435   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4436   (\glsfirstlongfont{\Glsaccesslong{##1}})%
4437 }%
4438 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
4439   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4440   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4441   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
4442 }%

```

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.

```

4443 \renewcommand*\glsxtrfullformat}[2]{%
4444   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4445   \ifglsxtrininsertinside\else##2\fi
4446 }%
4447 \renewcommand*\glsxtrfullplformat}[2]{%
4448   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4449   \ifglsxtrininsertinside\else##2\fi
4450 }%
4451 \renewcommand*\Glsxtrfullformat}[2]{%
4452   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4453   \ifglsxtrininsertinside\else##2\fi
4454 }%
4455 \renewcommand*\Glsxtrfullplformat}[2]{%
4456   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4457   \ifglsxtrininsertinside\else##2\fi
4458 }%
4459 }

```

ort-nolong-desc

```

4460 \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.

```

4461 \newabbreviationstyle{long-desc}%
4462 {%
4463   \renewcommand*{\CustomAbbreviationFields}{%
4464     name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
4465     sort={\the\glslongtok},
4466     first={\protect\glsfirstlongfont{\the\glslongtok}},
4467     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4468     text={\the\glslongtok},
4469     plural={\the\glslongpltok}%
4470   }%
4471   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4472     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4473 }%
4474 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4475   \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4476   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4477   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4478   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4479   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the long format followed by the short form in parentheses.

```

4480   \renewcommand*{\glsxtrinlinefullformat}[2]{%
4481     \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4482     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4483     (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
4484   }%
4485   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4486     \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
4487     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4488     (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4489   }%
4490   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4491     \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4492     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4493     (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
4494   }%
4495   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4496     \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
4497     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4498     (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4499   }%

```

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.

```

4500   \renewcommand*{\glsxtrfullformat}[2]{%
4501     \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
4502     \ifglsxtrininsertinside\else##2\fi

```

```

4503 }%
4504 \renewcommand*{\glxtrfullplformat}[2]{%
4505   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4506   \ifglxtrinsertinside\else##2\fi
4507 }%
4508 \renewcommand*{\Glsxtrfullformat}[2]{%
4509   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4510   \ifglxtrinsertinside\else##2\fi
4511 }%
4512 \renewcommand*{\Glsxtrfullplformat}[2]{%
4513   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4514   \ifglxtrinsertinside\else##2\fi
4515 }%
4516 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

4517 \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.

```

4518 \newabbreviationstyle{long}%
4519 {%
4520   \renewcommand*{\CustomAbbreviationFields}{%
4521     name={\protect\glsabbrvfont{\the\glsshorttok}},
4522     sort={\the\glsshorttok},
4523     first={\protect\glsfirstlongfont{\the\glslongtok}},
4524     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4525     text={\the\glslongtok},
4526     plural={\the\glslongpltok},%
4527     description={\the\glslongtok}%
4528   }%
4529   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4530     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4531 }%
4532 {%
4533   \GlsXtrUseAbbrStyleFmts{long-desc}%
4534 }

```

long-noshort Provide a synonym that matches similar styles.

```

4535 \letabbreviationstyle{long-noshort}{long}

```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

\glxtrscfont

```

4536 \newcommand*{\glxtrscfont}[1]{\textsc{#1}}

```

sxtrfirstscfont

```
4537 \newcommand*{\glxtrfirstscfont}[1]{\glxtrscfont{#1}}
```

and for the default short form suffix:

\glxtrscsuffix

```
4538 \newcommand*{\glxtrscsuffix}{\glstextup{\glspluralsuffix}}
```

long-short-sc

```
4539 \newabbreviationstyle{long-short-sc}%  
4540 {%  
4541   \GlsXtrUseAbbrStyleSetup{long-short}%  
4542 }%  
4543 {%
```

Mostly as long-short style:

```
4544   \GlsXtrUseAbbrStyleFmts{long-short}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4545   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4546   \renewcommand*{\glabbrvfont[1]{\glxtrscfont{##1}}}%  
4547   \renewcommand*{\glfirstabbrvfont[1]{\glxtrfirstscfont{##1}}}%  
4548 }
```

g-short-sc-desc

```
4549 \newabbreviationstyle{long-short-sc-desc}%  
4550 {%  
4551   \GlsXtrUseAbbrStyleSetup{long-short-desc}%  
4552 }%  
4553 {%
```

Mostly as long-short-desc style:

```
4554   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4555   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4556   \renewcommand*{\glabbrvfont[1]{\glxtrscfont{##1}}}%  
4557   \renewcommand*{\glfirstabbrvfont[1]{\glxtrfirstscfont{##1}}}%  
4558 }
```

Now the short (long) version

```
4559 \newabbreviationstyle{short-sc-long}%  
4560 {%  
4561   \GlsXtrUseAbbrStyleSetup{short-long}%  
4562 }%  
4563 {%
```

Mostly as short-long style:

```
4564   \GlsXtrUseAbbrStyleFmts{short-long}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4565 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4566 \renewcommand*{\glssabbrvfont[1]{\glxtrscfont{##1}}}%
4567 \renewcommand*{\glssfirstabbrvfont[1]{\glxtrfirstscfont{##1}}}%
4568 }
```

As before but user provides description

```
4569 \newabbreviationstyle{short-sc-long-desc}%
4570 {%
4571   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4572 }%
4573 {%
```

Mostly as short-long-desc style:

```
4574 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4575 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4576 \renewcommand*{\glssabbrvfont[1]{\glxtrscfont{##1}}}%
4577 \renewcommand*{\glssfirstabbrvfont[1]{\glxtrfirstscfont{##1}}}%
4578 }
```

short-sc

```
4579 \newabbreviationstyle{short-sc}%
4580 {%
4581   \GlsXtrUseAbbrStyleSetup{short-nolong}%
4582 }%
4583 {%
```

Mostly as short style:

```
4584 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4585 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4586 \renewcommand*{\glssabbrvfont[1]{\glxtrscfont{##1}}}%
4587 \renewcommand*{\glssfirstabbrvfont[1]{\glxtrfirstscfont{##1}}}%
4588 }
```

short-sc-nolong

```
4589 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
4590 \newabbreviationstyle{short-sc-desc}%
4591 {%
4592   \GlsXtrUseAbbrStyleSetup{short-desc}%
4593 }%
4594 {%
```

Mostly as short style:

```
4595 \GlsXtrUseAbbrStyleFmts{short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4596 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4597 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4598 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4599 }
```

-sc-nolong-desc

```
4600 \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.

```
4601 \newabbreviationstyle{long-noshort-sc}%
4602 {%
4603   \GlsXtrUseAbbrStyleSetup{long-noshort}%
4604 }%
4605 {%
```

Mostly as long style:

```
4606 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4607 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4608 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4609 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4610 }
```

long-sc Backward compatibility:

```
4611 \@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.

```
4612 \newabbreviationstyle{long-noshort-sc-desc}%
4613 {%
4614   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4615 }%
4616 {%
```

Mostly as long style:

```
4617 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4618 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
4619 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
4620 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
4621 }
```

long-desc-sc Backward compatibility:

```
4622 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```


ort-sc-footnote

```
4623 \newabbreviationstyle{short-sc-footnote}%  
4624 {%  
4625   \GlsXtrUseAbbrStyleSetup{short-footnote}%  
4626 }%  
4627 {%
```

Mostly as long style:

```
4628   \GlsXtrUseAbbrStyleFmts{short-footnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4629   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4630   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%  
4631   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%  
4632 }
```

footnote-sc Backward compatibility:

```
4633 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

```
4634 \newabbreviationstyle{short-sc-postfootnote}%  
4635 {%  
4636   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%  
4637 }%  
4638 {%
```

Mostly as long style:

```
4639   \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
4640   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%  
4641   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%  
4642   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%  
4643 }
```

postfootnote-sc Backward compatibility:

```
4644 \@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`

```
4645 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}
```

`sxtrfirstsmfont`

```
4646 \newcommand*{\glxtrfirstsmfont}[1]{\glxtrsmfont{#1}}
```

and for the default short form suffix:

\glxtrmsuffix

```
4647 \newcommand*{\glxtrmsuffix}{\glspluralsuffix}
```

long-short-sm

```
4648 \newabbreviationstyle{long-short-sm}%
```

```
4649 {%
```

```
4650   \GlsXtrUseAbbrStyleSetup{long-short}%
```

```
4651 }%
```

```
4652 {%
```

Mostly as long-short style:

```
4653   \GlsXtrUseAbbrStyleFmts{long-short}%
```

```
4654   \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
```

```
4655   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
```

```
4656   \renewcommand*\abbrvpluralsuffix{\protect\glxtrmsuffix}%
```

```
4657 }
```

g-short-sm-desc

```
4658 \newabbreviationstyle{long-short-sm-desc}%
```

```
4659 {%
```

```
4660   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
```

```
4661 }%
```

```
4662 {%
```

Mostly as long-short-desc style:

```
4663   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

```
4664   \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
```

```
4665   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
```

```
4666   \renewcommand*\abbrvpluralsuffix{\protect\glxtrmsuffix}%
```

```
4667 }
```

short-sm-long Now the short (long) version

```
4668 \newabbreviationstyle{short-sm-long}%
```

```
4669 {%
```

```
4670   \GlsXtrUseAbbrStyleSetup{short-long}%
```

```
4671 }%
```

```
4672 {%
```

Mostly as short-long style:

```
4673   \GlsXtrUseAbbrStyleFmts{short-long}%
```

```
4674   \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
```

```
4675   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
```

```
4676   \renewcommand*\abbrvpluralsuffix{\protect\glxtrmsuffix}%
```

```
4677 }
```

rt-sm-long-desc As before but user provides description

```
4678 \newabbreviationstyle{short-sm-long-desc}%
```

```
4679 {%
```

```
4680   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
```

```
4681 }%
```

```
4682 {%
```

Mostly as short-long-desc style:

```
4683 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4684 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4685 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4686 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4687 }
```

short-sm

```
4688 \newabbreviationstyle{short-sm}%
4689 {%
4690 \GlsXtrUseAbbrStyleSetup{short-nolong}%
4691 }%
4692 {%
```

Mostly as short style:

```
4693 \GlsXtrUseAbbrStyleFmts{short-nolong}%
4694 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4695 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4696 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4697 }
```

short-sm-nolong

```
4698 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
4699 \newabbreviationstyle{short-sm-desc}%
4700 {%
4701 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
4702 }%
4703 {%
```

Mostly as short style:

```
4704 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
4705 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
4706 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
4707 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
4708 }
```

-sm-nolong-desc

```
4709 \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.

```
4710 \newabbreviationstyle{long-noshort-sm}%
4711 {%
4712 \GlsXtrUseAbbrStyleSetup{long-noshort}%
4713 }%
4714 {%
```

Mostly as long style:

```
4715 \GlsXtrUseAbbrStyleFmts{long-noshort}%
4716 \renewcommand*{\glsabbrvfont[1]{\glsxtrsmfont{##1}}}%
4717 \renewcommand*{\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}}%
4718 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4719 }
```

long-sm Backward compatibility:

```
4720 \@glsxtr@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.

```
4721 \newabbreviationstyle{long-noshort-sm-desc}%
4722 {%
4723 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4724 }%
4725 {%
```

Mostly as long style:

```
4726 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4727 \renewcommand*{\glsabbrvfont[1]{\glsxtrsmfont{##1}}}%
4728 \renewcommand*{\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}}%
4729 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4730 }
```

long-desc-sm Backward compatibility:

```
4731 \@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}
```

short-sm-footnote

```
4732 \newabbreviationstyle{short-sm-footnote}%
4733 {%
4734 \GlsXtrUseAbbrStyleSetup{short-footnote}%
4735 }%
4736 {%
```

Mostly as long style:

```
4737 \GlsXtrUseAbbrStyleFmts{short-footnote}%
4738 \renewcommand*{\glsabbrvfont[1]{\glsxtrsmfont{##1}}}%
4739 \renewcommand*{\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}}%
4740 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4741 }
```

footnote-sm Backward compatibility:

```
4742 \@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

sm-postfootnote

```
4743 \newabbreviationstyle{short-sm-postfootnote}%
4744 {%
4745 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
```

4746 }%
 4747 {%

Mostly as long style:

4748 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
 4749 \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
 4750 \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
 4751 \renewcommand*\abbrvpluralsuffix{\protect\glstrsmsuffix}%
 4752 }

postfootnote-sm Backward compatibility:

4753 \@glstr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

1.6.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont

4754 \newcommand*\glsabbrvemfont[1]{\emph{#1}}%

irstabbrvemfont

4755 \newcommand*\glsfirstabbrvemfont[1]{\glsabbrvemfont{#1}}%

firstlongemfont Only used by the “long-em” styles.

4756 \newcommand*\glsfirstlongemfont[1]{\glslongemfont{#1}}%

\glslongemfont Only used by the “long-em” styles.

4757 \newcommand*\glslongemfont[1]{\emph{#1}}%

long-short-em

4758 \newabbreviationstyle{long-short-em}%
 4759 {%
 4760 \GlsXtrUseAbbrStyleSetup{long-short}%
 4761 }%
 4762 {%

Mostly as long-short style:

4763 \GlsXtrUseAbbrStyleFmts{long-short}%
 4764 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
 4765 }

g-short-em-desc

4766 \newabbreviationstyle{long-short-em-desc}%
 4767 {%
 4768 \GlsXtrUseAbbrStyleSetup{long-short-desc}%
 4769 }%
 4770 {%

Mostly as long-short-desc style:

```
4771 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
4772 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4773 }
```

ong-em-short-em

```
4774 \newabbreviationstyle{long-em-short-em}%
4775 {%
```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```
4776 \renewcommand*{\CustomAbbreviationFields}{%
4777   name={\protect\glsabbrvfont{\the\glsshorttok}},
4778   sort={\the\glsshorttok},
4779   first={\protect\glsfirstlongfont{\the\glslongtok}%
4780     \protect\glsxtrfullsep{\the\glslabeltok}%
4781     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4782   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4783     \protect\glsxtrfullsep{\the\glslabeltok}%
4784     (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
4785   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
4786   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```
4787 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4788   \glshasattribute{\the\glslabeltok}{regular}%
4789   {%
4790     \glssetattribute{\the\glslabeltok}{regular}{false}%
4791   }%
4792   {}%
4793 }%
4794 }%
4795 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4796 \GlsXtrUseAbbrStyleFmts{long-short}%
4797 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
4798 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
4799 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4800 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4801 }
```

m-short-em-desc

```
4802 \newabbreviationstyle{long-em-short-em-desc}%
4803 {%
4804   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
4805 }%
4806 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4807 \GlsXtrUseAbbrStyleFmts{long-short-desc}%

```

```

4808 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4809 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4810 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4811 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4812 }

```

short-em-long Now the short (long) version

```

4813 \newabbreviationstyle{short-em-long}%
4814 {%
4815 \GlsXtrUseAbbrStyleSetup{short-long}%
4816 }%
4817 {%

```

Mostly as short-long style:

```

4818 \GlsXtrUseAbbrStyleFmts{short-long}%
4819 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4820 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4821 }

```

rt-em-long-desc As before but user provides description

```

4822 \newabbreviationstyle{short-em-long-desc}%
4823 {%
4824 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4825 }%
4826 {%

```

Mostly as short-long-desc style:

```

4827 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4828 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4829 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4830 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4831 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4832 }

```

hort-em-long-em

```

4833 \newabbreviationstyle{short-em-long-em}%
4834 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

4835 \renewcommand*\CustomAbbreviationFields{%
4836 name={\protect\glsabbrvfont{\the\glsshorttok}},
4837 sort={\the\glsshorttok},
4838 description={\protect\glslongemfont{\the\glslongtok}},%
4839 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
4840 \protect\glsxtrfullsep{\the\glslabeltok}%
4841 (\protect\glsfirstlongfont{\the\glslongtok}}),%
4842 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
4843 \protect\glsxtrfullsep{\the\glslabeltok}%
4844 (\protect\glsfirstlongfont{\the\glslongpltok}}),%
4845 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```
4846 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4847   \glshasattribute{\the\glslabeltok}{regular}%
4848   {%
4849     \glsselattribute{\the\glslabeltok}{regular}{false}%
4850   }%
4851   {}%
4852 }%
4853 }%
4854 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4855 \GlsXtrUseAbbrStyleFmts{short-long}%
4856 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4857 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4858 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4859 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4860 }
```

em-long-em-desc

```
4861 \newabbreviationstyle{short-em-long-em-desc}%
4862 {%
4863   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4864 }%
4865 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4866 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
4867 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4868 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4869 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
4870 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
4871 }
```

short-em

```
4872 \newabbreviationstyle{short-em}%
4873 {%
4874   \GlsXtrUseAbbrStyleSetup{short-nolong}%
4875 }%
4876 {%
```

Mostly as short style:

```
4877 \GlsXtrUseAbbrStyleFmts{short-nolong}%
4878 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
4879 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
4880 }
```

short-em-nolong

```
4881 \letabbreviationstyle{short-em-nolong}{short-em}
```



```

short-em-desc
4882 \newabbreviationstyle{short-em-desc}%
4883 {%
4884   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
4885 }%
4886 {%

  Mostly as short style:
4887   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
4888   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4889   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4890 }

-em-nolong-desc
4891 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

long-noshort-em  The short form is explicitly invoked through commands like \glssshort.
4892 \newabbreviationstyle{long-noshort-em}%
4893 {%
4894   \GlsXtrUseAbbrStyleSetup{long-noshort}%
4895 }%
4896 {%

  Mostly as long-noshort style:
4897   \GlsXtrUseAbbrStyleFmts{long-noshort}%
4898   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4899   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4900 }

long-em  Backward compatibility:
4901 \@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

g-em-noshort-em  The short form is explicitly invoked through commands like \glssshort.
4902 \newabbreviationstyle{long-em-noshort-em}%
4903 {%
4904   \renewcommand*\CustomAbbreviationFields{%
4905     name={\protect\glsabbrvfont{\the\glssshorttok}},
4906     sort={\the\glssshorttok},
4907     first={\protect\glsfirstlongfont{\the\glslongtok}},
4908     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
4909     text={\the\glslongtok},
4910     plural={\the\glslongpltok},%
4911     description={\protect\glslongemfont{\the\glslongtok}}%
4912   }%
4913   \renewcommand*\GlsXtrPostNewAbbreviation{%
4914     \glssetattribute{\the\glslabeltok}{regular}{true}}%
4915 }%
4916 {%

```

Mostly as long-noshort style:

```
4917 \GlsXtrUseAbbrStyleFmts{long-noshort}%
4918 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4919 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4920 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4921 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4922 }
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
4923 \newabbreviationstyle{long-noshort-em-desc}%
4924 {%
4925 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4926 }%
4927 {%
```

Mostly as long style:

```
4928 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4929 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4930 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4931 }
```

long-desc-em Backward compatibility:

```
4932 \@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.

```
4933 \newabbreviationstyle{long-em-noshort-em-desc}%
4934 {%
4935 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
4936 }%
4937 {%
```

Mostly as long style:

```
4938 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
4939 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4940 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4941 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
4942 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
4943 }
```

ort-em-footnote

```
4944 \newabbreviationstyle{short-em-footnote}%
4945 {%
4946 \GlsXtrUseAbbrStyleSetup{short-footnote}%
4947 }%
4948 {%
```

Mostly as long style:

```
4949 \GlsXtrUseAbbrStyleFmts{short-footnote}%
4950 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4951 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4952 }
```

footnote-em Backward compatibility:

```
4953 \@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

```
4954 \newabbreviationstyle{short-em-postfootnote}%
4955 {%
4956 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
4957 }%
4958 {%
```

Mostly as long style:

```
4959 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
4960 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
4961 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
4962 }
```

postfootnote-em Backward compatibility:

```
4963 \@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.

```
4964 \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.

```
4965 \ifdef\glscurrentfieldvalue
4966 {
4967 \newcommand*\glsxtruserparen[2]{%
4968 \glsxtrfullsep{#2}%
4969 (#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{})%
4970 }
4971 }
4972 {
4973 \newcommand*\glsxtruserparen[2]{%
4974 \glsxtrfullsep{#2}%
4975 (#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{})%
4976 }
4977 }
```

Font used for short form:

lsabbrvuserfont

```
4978 \newcommand*{\glsabbrvuserfont}[1]{#1}
```

Font used for short form on first use:

stabbrvuserfont

```
4979 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
4980 \newcommand*{\glslonguserfont}[1]{#1}
```

Font used for long form on first use:

rstlonguserfont

```
4981 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
4982 \newcommand*{\glsxtrusersuffix}{\glspluralsuffix}
```

long-short-user

```
4983 \newabbreviationstyle{long-short-user}%
```

```
4984 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
4985 \renewcommand*{\CustomAbbreviationFields}{%
```

```
4986   name={\protect\glsabbrvfont{\the\glsshorttok}},
```

```
4987   sort={\the\glsshorttok},
```

```
4988   first={\protect\glsfirstlongfont{\the\glslongtok}}%
```

```
4989   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
```

```
4990   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
```

```
4991   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}},
```

```
4992   plural={\protect\glsabbbfont{\the\glsshortpltok}},%
```

```
4993   description={\protect\glslonguserfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
4994 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
4995   \glshasattribute{\the\glslabeltok}{regular}}%
```

```
4996   {%
```

```
4997     \glssetattribute{\the\glslabeltok}{regular}{false}}%
```

```
4998   }%
```

```
4999   {}%
```

```
5000 }%
```

```
5001 }%
```

```
5002 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
5003 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
5004 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
5005 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
5006 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
5007 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
5008 \renewcommand*{\glxtrfullformat}[2]{%
5009   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
5010   \ifglxtrininsertinside\else##2\fi
5011   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
5012 }%
5013 \renewcommand*{\glxtrfullplformat}[2]{%
5014   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
5015   \ifglxtrininsertinside\else##2\fi
5016   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
5017 }%
5018 \renewcommand*{\Glsxtrfullformat}[2]{%
5019   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
5020   \ifglxtrininsertinside\else##2\fi
5021   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
5022 }%
5023 \renewcommand*{\Glsxtrfullplformat}[2]{%
5024   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
5025   \ifglxtrininsertinside\else##2\fi
5026   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
5027 }%
5028 }
```

short-user-desc

```
5029 \newabbreviationstyle{long-short-user-desc}%
5030 {%
5031   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5032 }%
5033 {%
5034   \GlsXtrUseAbbrStyleFmts{long-short-user}%
5035 }
```

short-long-user

```
5036 \newabbreviationstyle{short-long-user}%
5037 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
5038 \renewcommand*{\CustomAbbreviationFields}{%
5039   name={\protect\glsabbrvfont{\the\glsshorttok}},
5040   sort={\the\glsshorttok},
5041   description={\protect\glslonguserfont{\the\glslongtok}},%
5042   first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
5043   \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}},%
```

```

5044     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5045     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
5046     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

5047 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5048   \glsattribute{\the\glslabeltok}{regular}%
5049   {%
5050     \glssetattribute{\the\glslabeltok}{regular}{false}%
5051   }%
5052   {}%
5053 }%
5054}%
5055{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5056 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
5057 \renewcommand*{\glsabbrvfont[1]}{\glsabbrvuserfont{##1}}%
5058 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
5059 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
5060 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5061 \renewcommand*{\glsxtrfullformat}[2]{%
5062   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
5063   \ifglsxtrinsertinside\else##2\fi
5064   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
5065 }%
5066 \renewcommand*{\glsxtrfullplformat}[2]{%
5067   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
5068   \ifglsxtrinsertinside\else##2\fi
5069   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
5070 }%
5071 \renewcommand*{\Glsxtrfullformat}[2]{%
5072   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
5073   \ifglsxtrinsertinside\else##2\fi
5074   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslong{##1}}}{##1}%
5075 }%
5076 \renewcommand*{\Glsxtrfullplformat}[2]{%
5077   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
5078   \ifglsxtrinsertinside\else##2\fi
5079   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}{##1}%
5080 }%
5081}

```

-long-user-desc

```

5082 \newabbreviationstyle{short-long-user-desc}%
5083 {%
5084   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5085 }%

```

```

5086 {%
5087   \GlsXtrUseAbbrStyleFmts{short-long-user}%
5088 }

```

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:

```

5089 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```

5090 \renewcommand*{\markright}[1]{%
5091   \glsxtrmarkhook
5092   \@glsxtr@org@markright{#1}%
5093   \glsxtrrestoremarkhook
5094 }

```

`\markboth` Save original definition:

```

5095 \let\@glsxtr@org@markboth\markboth

```

Redefine (grouping not added in case it interferes with the original code):

```

5096 \renewcommand*{\markboth}[2]{%
5097   \glsxtrmarkhook
5098   \@glsxtr@org@markboth{#1}{#2}%
5099   \glsxtrrestoremarkhook
5100 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```
5101 \newcommand*{\glxstrRevertMarks}{%
5102   \let\markright\@glxstr@org@markright
5103   \let\markboth\@glxstr@org@markboth
5104 }
```

\glxstrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
5105 \newcommand*{\glxstrmarkhook}{%
```

Save current definitions:

```
5106   \let\@glxstr@org@MakeUppercase\MakeUppercase
5107   \let\@glxstr@org@glxstrtitleshort\glxstrtitleshort
5108   \let\@glxstr@org@glxstrtitleshortpl\glxstrtitleshortpl
5109   \let\@glxstr@org@Glsxtrtitleshort\Glsxtrtitleshort
5110   \let\@glxstr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
5111   \let\@glxstr@org@glxstrtitletext\glxstrtitletext
5112   \let\@glxstr@org@Glsxtrtitletext\Glsxtrtitletext
5113   \let\@glxstr@org@glxstrtitleplural\glxstrtitleplural
5114   \let\@glxstr@org@Glsxtrtitleplural\Glsxtrtitleplural
5115   \let\@glxstr@org@glxstrtitlefirst\glxstrtitlefirst
5116   \let\@glxstr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
5117   \let\@glxstr@org@glxstrtitlefirstplural\glxstrtitlefirstplural
5118   \let\@glxstr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
5119   \let\@glxstr@org@glxstrtitlelong\glxstrtitlelong
5120   \let\@glxstr@org@glxstrtitlelongpl\glxstrtitlelongpl
5121   \let\@glxstr@org@Glsxtrtitlelong\Glsxtrtitlelong
5122   \let\@glxstr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
5123   \let\@glxstr@org@glxstrtitlefull\glxstrtitlefull
5124   \let\@glxstr@org@glxstrtitlefullpl\glxstrtitlefullpl
5125   \let\@glxstr@org@Glsxtrtitlefull\Glsxtrtitlefull
5126   \let\@glxstr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
5127   \let\MakeUppercase\MakeTextUppercase
5128   \let\glxstrtitleshort\glxstrheadshort
5129   \let\glxstrtitleshortpl\glxstrheadshortpl
5130   \let\Glsxtrtitleshort\Glsxtrheadshort
5131   \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
5132   \let\glxstrtitletext\glxstrheadtext
5133   \let\Glsxtrtitletext\Glsxtrheadtext
5134   \let\glxstrtitleplural\glxstrheadplural
5135   \let\Glsxtrtitleplural\Glsxtrheadplural
5136   \let\glxstrtitlefirst\glxstrheadfirst
5137   \let\Glsxtrtitlefirst\Glsxtrheadfirst
5138   \let\glxstrtitlefirstplural\glxstrheadfirstplural
5139   \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
5140   \let\glxstrtitlelong\glxstrheadlong
```



```

5141 \let\glsxtrtitlelongpl\glsxtrheadlongpl
5142 \let\Glsxtrtitlelong\Glsxtrheadlong
5143 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
5144 \let\glsxtrtitlefull\glsxtrheadfull
5145 \let\glsxtrtitlefullpl\glsxtrheadfullpl
5146 \let\Glsxtrtitlefull\Glsxtrheadfull
5147 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
5148 }

```

`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.)

```

5149 \newcommand*{\glsxtrrestoremarkhook}{%
5150 \let\MakeUppercase\@glsxtr@org@MakeUppercase
5151 \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
5152 \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
5153 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
5154 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
5155 \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
5156 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
5157 \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
5158 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
5159 \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
5160 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
5161 \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
5162 \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
5163 \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
5164 \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
5165 \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
5166 \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
5167 \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
5168 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
5169 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
5170 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
5171 }

```

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.

```

5172 \newcommand*{\glsxtrheadshort}[1]{%
5173 \protect\NoCaseChange
5174 {%
5175 \glsifattribute{#1}{headuc}{true}%
5176 {%
5177 \GLSxtrshort[noindex,hyper=false]{#1}[]%
5178 }%
5179 {%

```

```

5180     \glsxtrshort[noindex,hyper=false]{#1}[]%
5181 }%
5182 }%
5183 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

5184 \newrobustcmd*{\glsxtrtitleshort}[1]{%
5185   \glsxtrshort[noindex,hyper=false]{#1}[]%
5186 }

```

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.

```

5187 \newcommand*{\glsxtrheadshortpl}[1]{%
5188   \protect\NoCaseChange
5189   {%
5190     \glsifattribute{#1}{headuc}{true}%
5191     {%
5192       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
5193     }%
5194     {%
5195       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
5196     }%
5197   }%
5198 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

5199 \newrobustcmd*{\glsxtrtitleshortpl}[1]{%
5200   \glsxtrshortpl[noindex,hyper=false]{#1}[]%
5201 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

5202 \newcommand*{\Glsxtrheadshort}[1]{%
5203   \protect\NoCaseChange
5204   {%
5205     \glsifattribute{#1}{headuc}{true}%
5206     {%
5207       \GLSxtrshort[noindex,hyper=false]{#1}[]%
5208     }%
5209     {%
5210       \Glsxtrshort[noindex,hyper=false]{#1}[]%
5211     }%
5212   }%
5213 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5214 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
5215   \Glsxtrshort[noindex,hyper=false]{#1}[]%
5216 }

```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```

5217 \newcommand*{\Glsxtrheadshortpl}[1]{%
5218   \protect\NoCaseChange
5219   {%
5220     \glsifattribute{#1}{headuc}{true}%
5221     {%
5222       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5223     }%
5224     {%
5225       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5226     }%
5227   }%
5228 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5229 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
5230   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5231 }

```

\glsxtrheadtext As above but for the text value.

```

5232 \newcommand*{\glsxtrheadtext}[1]{%
5233   \protect\NoCaseChange
5234   {%
5235     \glsifattribute{#1}{headuc}{true}%
5236     {%
5237       \GLStext[noindex,hyper=false]{#1}[]%
5238     }%
5239     {%
5240       \glstext[noindex,hyper=false]{#1}[]%
5241     }%
5242   }%
5243 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

5244 \newrobustcmd*{\glsxtrtitletext}[1]{%
5245   \glstext[noindex,hyper=false]{#1}[]%
5246 }

```

\Glsxtrheadtext First letter converted to upper case

```

5247 \newcommand*{\Glsxtrheadtext}[1]{%
5248   \protect\NoCaseChange
5249   {%

```

```

5250 \glsifattribute{#1}{headuc}{true}%
5251 {%
5252   \GLStext[noindex,hyper=false]{#1}[]%
5253 }%
5254 {%
5255   \GLStext[noindex,hyper=false]{#1}[]%
5256 }%
5257 }%
5258 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

5259 \newrobustcmd*{\Glsxtrtitletext}[1]{%
5260   \GLStext[noindex,hyper=false]{#1}[]%
5261 }

```

lsxtrheadplural As above but for the plural value.

```

5262 \newcommand*{\lsxtrheadplural}[1]{%
5263   \protect\NoCaseChange
5264   {%
5265     \glsifattribute{#1}{headuc}{true}%
5266     {%
5267       \GLSplural[noindex,hyper=false]{#1}[]%
5268     }%
5269     {%
5270       \Glsplural[noindex,hyper=false]{#1}[]%
5271     }%
5272   }%
5273 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

5274 \newrobustcmd*{\gsxtrtitleplural}[1]{%
5275   \Glsplural[noindex,hyper=false]{#1}[]%
5276 }

```

lsxtrheadplural Convert first letter to upper case.

```

5277 \newcommand*{\Glsxtrheadplural}[1]{%
5278   \protect\NoCaseChange
5279   {%
5280     \glsifattribute{#1}{headuc}{true}%
5281     {%
5282       \GLSplural[noindex,hyper=false]{#1}[]%
5283     }%
5284     {%
5285       \Glsplural[noindex,hyper=false]{#1}[]%
5286     }%
5287   }%
5288 }

```

`lsxtrtitleplural` Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
5289 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
5290   \Glsplural[noindex,hyper=false]{#1}[]%
5291 }
```

`glsxtrheadfirst` As above but for the first value.

```
5292 \newcommand*{\glsxtrheadfirst}[1]{%
5293   \protect\NoCaseChange
5294   {%
5295     \glsifattribute{#1}{headuc}{true}%
5296     {%
5297       \GLSfirst[noindex,hyper=false]{#1}[]%
5298     }%
5299     {%
5300       \glsfirst[noindex,hyper=false]{#1}[]%
5301     }%
5302   }%
5303 }
```

`lsxtrtitlefirst` Command to display first value in section title and table of contents.

```
5304 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
5305   \glsfirst[noindex,hyper=false]{#1}[]%
5306 }
```

`Glsxtrheadfirst` First letter converted to upper case

```
5307 \newcommand*{\Glsxtrheadfirst}[1]{%
5308   \protect\NoCaseChange
5309   {%
5310     \glsifattribute{#1}{headuc}{true}%
5311     {%
5312       \GLSfirst[noindex,hyper=false]{#1}[]%
5313     }%
5314     {%
5315       \Glsfirst[noindex,hyper=false]{#1}[]%
5316     }%
5317   }%
5318 }
```

`lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
5319 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
5320   \Glsfirst[noindex,hyper=false]{#1}[]%
5321 }
```

`headfirstplural` As above but for the firstplural value.

```
5322 \newcommand*{\glsxtrheadfirstplural}[1]{%
5323   \protect\NoCaseChange
```

```

5324 {%
5325   \glsifattribute{#1}{headuc}{true}%
5326   {%
5327     \GLSfirstplural[noindex,hyper=false]{#1}[]%
5328   }%
5329   {%
5330     \glsfirstplural[noindex,hyper=false]{#1}[]%
5331   }%
5332 }%
5333 }

```

`\titlefirstplural` Command to display firstplural value in section title and table of contents.

```

5334 \newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
5335   \glsfirstplural[noindex,hyper=false]{#1}[]%
5336 }

```

`\headfirstplural` First letter converted to upper case

```

5337 \newcommand*{\Glsxtrheadfirstplural}[1]{%
5338   \protect\NoCaseChange
5339   {%
5340     \glsifattribute{#1}{headuc}{true}%
5341     {%
5342       \GLSfirstplural[noindex,hyper=false]{#1}[]%
5343     }%
5344     {%
5345       \Glsfirstplural[noindex,hyper=false]{#1}[]%
5346     }%
5347   }%
5348 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

5349 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
5350   \Glsfirstplural[noindex,hyper=false]{#1}[]%
5351 }

```

`\glsxtrheadlong` Command used to display long form in the page header.

```

5352 \newcommand*{\glsxtrheadlong}[1]{%
5353   \protect\NoCaseChange
5354   {%
5355     \glsifattribute{#1}{headuc}{true}%
5356     {%
5357       \GLSxtrlong[noindex,hyper=false]{#1}[]%
5358     }%
5359     {%
5360       \glsxtrlong[noindex,hyper=false]{#1}[]%
5361     }%
5362   }%
5363 }

```

`glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
5364 \newrobustcmd*{\glsxtrtitlelong}[1]{%
5365   \glsxtrlong[noindex,hyper=false]{#1}[]%
5366 }
```

`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.

```
5367 \newcommand*{\glsxtrheadlongpl}[1]{%
5368   \protect\NoCaseChange
5369   {%
5370     \glsifattribute{#1}{headuc}{true}%
5371     {%
5372       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5373     }%
5374     {%
5375       \glsxtrlongpl[noindex,hyper=false]{#1}[]%
5376     }%
5377   }%
5378 }
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
5379 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%
5380   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
5381 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
5382 \newcommand*{\Glsxtrheadlong}[1]{%
5383   \protect\NoCaseChange
5384   {%
5385     \glsifattribute{#1}{headuc}{true}%
5386     {%
5387       \GLSxtrlong[noindex,hyper=false]{#1}[]%
5388     }%
5389     {%
5390       \Glsxtrlong[noindex,hyper=false]{#1}[]%
5391     }%
5392   }%
5393 }
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5394 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
5395   \Glsxtrlong[noindex,hyper=false]{#1}[]%
5396 }
```

`\lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
5397 \newcommand*{\Glsxtrheadlongpl}[1]{%
5398   \protect\NoCaseChange
5399   {%
5400     \glsifattribute{#1}{headuc}{true}%
5401     {%
5402       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5403     }%
5404     {%
5405       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
5406     }%
5407   }%
5408 }
```

`\sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5409 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
5410   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
5411 }
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
5412 \newcommand*{\glsxtrheadfull}[1]{%
5413   \protect\NoCaseChange
5414   {%
5415     \glsifattribute{#1}{headuc}{true}%
5416     {%
5417       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5418     }%
5419     {%
5420       \glsxtrfull[noindex,hyper=false]{#1}[]%
5421     }%
5422   }%
5423 }
```

`\glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
5424 \newrobustcmd*{\glsxtrtitlefull}[1]{%
5425   \glsxtrfull[noindex,hyper=false]{#1}[]%
5426 }
```

`\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.

```
5427 \newcommand*{\glsxtrheadfullpl}[1]{%
5428   \protect\NoCaseChange
5429   {%
5430     \glsifattribute{#1}{headuc}{true}%
5431     {%
```



```

5432      \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
5433    }%
5434    {%
5435      \glsxtrfullpl[noindex,hyper=false]{#1}[]%
5436    }%
5437  }%
5438 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

5439 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
5440   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
5441 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

5442 \newcommand*{\Glsxtrheadfull}[1]{%
5443   \protect\NoCaseChange
5444   {%
5445     \glsifattribute{#1}{headuc}{true}%
5446     {%
5447       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5448     }%
5449     {%
5450       \Glsxtrfull[noindex,hyper=false]{#1}[]%
5451     }%
5452   }%
5453 }

```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

5454 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
5455   \Glsxtrfull[noindex,hyper=false]{#1}[]%
5456 }

```

lxsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

5457 \newcommand*{\Glsxtrheadfullpl}[1]{%
5458   \protect\NoCaseChange
5459   {%
5460     \glsifattribute{#1}{headuc}{true}%
5461     {%
5462       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
5463     }%
5464     {%
5465       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
5466     }%
5467   }%
5468 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5469 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
5470   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
5471 }
```

`\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.

```
5472 \ifdef\texorpdfstring
5473 {
5474   \newcommand*{\glsfmtshort}[1]{%
5475     \texorpdfstring
5476       {\Glsxtrtitleshort{#1}}%
5477       {\Glsentryshort{#1}}%
5478   }
5479 }
5480 {
5481   \newcommand*{\glsfmtshort}[1]{%
5482     \Glsxtrtitleshort{#1}}
5483 }
```

Similarly for the plural version.

`\glsfmtshortpl`

```
5484 \ifdef\texorpdfstring
5485 {
5486   \newcommand*{\glsfmtshortpl}[1]{%
5487     \texorpdfstring
5488       {\Glsxtrtitleshortpl{#1}}%
5489       {\Glsentryshortpl{#1}}%
5490   }
5491 }
5492 {
5493   \newcommand*{\glsfmtshortpl}[1]{%
5494     \Glsxtrtitleshortpl{#1}}
5495 }
```

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).

```
5496 \ifdef\texorpdfstring
5497 {
5498   \newcommand*{\Glsfmtshort}[1]{%
5499     \texorpdfstring
5500       {\Glsxtrtitleshort{#1}}%
5501       {\Glsentryshort{#1}}%
5502   }
5503 }
```

```

5504 {
5505   \newcommand*{\Glsfmtshort}[1]{%
5506     \Glsxtrtitleshort{#1}}
5507 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

5508 \ifdef\teorpdfstring
5509 {
5510   \newcommand*{\Glsfmtshortpl}[1]{%
5511     \teorpdfstring
5512       {\Glsxtrtitleshortpl{#1}}%
5513       {\glstryshortpl{#1}}%
5514   }
5515 }
5516 {
5517   \newcommand*{\Glsfmtshortpl}[1]{%
5518     \Glsxtrtitleshortpl{#1}}
5519 }

```

`\glsfmttext` As above but for the text value.

```

5520 \ifdef\teorpdfstring
5521 {
5522   \newcommand*{\glsfmttext}[1]{%
5523     \teorpdfstring
5524       {\glxtrtitletext{#1}}%
5525       {\glstrytext{#1}}%
5526   }
5527 }
5528 {
5529   \newcommand*{\glsfmttext}[1]{%
5530     \glxtrtitletext{#1}}
5531 }

```

`\Glsfmttext` First letter converted to upper case.

```

5532 \ifdef\teorpdfstring
5533 {
5534   \newcommand*{\Glsfmttext}[1]{%
5535     \teorpdfstring
5536       {\Glsxtrtitletext{#1}}%
5537       {\glstrytext{#1}}%
5538   }
5539 }
5540 {
5541   \newcommand*{\Glsfmttext}[1]{%
5542     \Glsxtrtitletext{#1}}
5543 }

```

`\glsfmtplural` As above but for the plural value.

```

5544 \ifdef\teorpdfstring

```

```

5545 {
5546   \newcommand*{\glsfmtplural}[1]{%
5547     \texorpdfstring
5548     {\glsxtrtitleplural{#1}}%
5549     {\glsentryplural{#1}}%
5550   }
5551 }
5552 {
5553   \newcommand*{\glsfmtplural}[1]{%
5554     \glsxtrtitleplural{#1}}
5555 }

```

`\Glsfmtplural` First letter converted to upper case.

```

5556 \ifdef\texorpdfstring
5557 {
5558   \newcommand*{\Glsfmtplural}[1]{%
5559     \texorpdfstring
5560     {\Glsxtrtitleplural{#1}}%
5561     {\glsentryplural{#1}}%
5562   }
5563 }
5564 {
5565   \newcommand*{\Glsfmtplural}[1]{%
5566     \Glsxtrtitleplural{#1}}
5567 }

```

`\glsfmtfirst` As above but for the first value.

```

5568 \ifdef\texorpdfstring
5569 {
5570   \newcommand*{\glsfmtfirst}[1]{%
5571     \texorpdfstring
5572     {\glsxtrtitlefirst{#1}}%
5573     {\glsentryfirst{#1}}%
5574   }
5575 }
5576 {
5577   \newcommand*{\glsfmtfirst}[1]{%
5578     \glsxtrtitlefirst{#1}}
5579 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

5580 \ifdef\texorpdfstring
5581 {
5582   \newcommand*{\Glsfmtfirst}[1]{%
5583     \texorpdfstring
5584     {\Glsxtrtitlefirst{#1}}%
5585     {\glsentryfirst{#1}}%
5586   }
5587 }

```

```

5588 {
5589   \newcommand*{\Glsfmtfirst}[1]{%
5590     \Glsxtrtitlefirst{#1}}
5591 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

5592 \ifdef\teorpdfstring
5593 {
5594   \newcommand*{\glsfmtfirstpl}[1]{%
5595     \teorpdfstring
5596       {\Glsxtrtitlefirstplural{#1}}%
5597       {\Glsentryfirstplural{#1}}%
5598   }
5599 }
5600 {
5601   \newcommand*{\glsfmtfirstpl}[1]{%
5602     \Glsxtrtitlefirstplural{#1}}
5603 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

5604 \ifdef\teorpdfstring
5605 {
5606   \newcommand*{\Glsfmtfirstpl}[1]{%
5607     \teorpdfstring
5608       {\Glsxtrtitlefirstplural{#1}}%
5609       {\Glsentryfirstplural{#1}}%
5610   }
5611 }
5612 {
5613   \newcommand*{\Glsfmtfirstpl}[1]{%
5614     \Glsxtrtitlefirstplural{#1}}
5615 }

```

`\glsfmtlong` As above but for the long value.

```

5616 \ifdef\teorpdfstring
5617 {
5618   \newcommand*{\glsfmtlong}[1]{%
5619     \teorpdfstring
5620       {\Glsxtrtitlelong{#1}}%
5621       {\Glsentrylong{#1}}%
5622   }
5623 }
5624 {
5625   \newcommand*{\glsfmtlong}[1]{%
5626     \Glsxtrtitlelong{#1}}
5627 }

```

`\Glsfmtlong` First letter converted to upper case.

```

5628 \ifdef\teorpdfstring

```

```

5629 {
5630   \newcommand*{\Glsfmtlong}[1]{%
5631     \texorpdfstring
5632     {\Glsxtrtitlelong{#1}}%
5633     {\glsentrylong{#1}}%
5634   }
5635 }
5636 {
5637   \newcommand*{\Glsfmtlong}[1]{%
5638     \Glsxtrtitlelong{#1}}
5639 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

5640 \ifdef\texorpdfstring
5641 {
5642   \newcommand*{\glsfmtlongpl}[1]{%
5643     \texorpdfstring
5644     {\glsxtrtitlelongpl{#1}}%
5645     {\glsentrylongpl{#1}}%
5646   }
5647 }
5648 {
5649   \newcommand*{\glsfmtlongpl}[1]{%
5650     \glsxtrtitlelongpl{#1}}
5651 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

5652 \ifdef\texorpdfstring
5653 {
5654   \newcommand*{\Glsfmtlongpl}[1]{%
5655     \texorpdfstring
5656     {\Glsxtrtitlelongpl{#1}}%
5657     {\glsentrylongpl{#1}}%
5658   }
5659 }
5660 {
5661   \newcommand*{\Glsfmtlongpl}[1]{%
5662     \Glsxtrtitlelongpl{#1}}
5663 }

```

`\glsfmtfull` In-line full format.

```

5664 \ifdef\texorpdfstring
5665 {
5666   \newcommand*{\glsfmtfull}[1]{%
5667     \texorpdfstring
5668     {\glsxtrtitlefull{#1}}%
5669     {\glsxtrinlinefullformat{#1}{}}%
5670   }
5671 }

```

```

5672 {
5673   \newcommand*{\glsfmtfull}[1]{%
5674     \glsxtrtitlefull{#1}}
5675 }

```

\Glsfmtfull First letter converted to upper case.

```

5676 \ifdef\teorpdfstring
5677 {
5678   \newcommand*{\Glsfmtfull}[1]{%
5679     \teorpdfstring
5680     {\Glsxtrtitlefull{#1}}%
5681     {\Glsxtrinlinefullformat{#1}-{}}%
5682   }
5683 }
5684 {
5685   \newcommand*{\Glsfmtfull}[1]{%
5686     \Glsxtrtitlefull{#1}}
5687 }

```

\glsfmtfullpl In-line full plural format.

```

5688 \ifdef\teorpdfstring
5689 {
5690   \newcommand*{\glsfmtfullpl}[1]{%
5691     \teorpdfstring
5692     {\glsxtrtitlefullpl{#1}}%
5693     {\glsxtrinlinefullplformat{#1}-{}}%
5694   }
5695 }
5696 {
5697   \newcommand*{\glsfmtfullpl}[1]{%
5698     \glsxtrtitlefullpl{#1}}
5699 }

```

\Glsfmtfullpl First letter converted to upper case.

```

5700 \ifdef\teorpdfstring
5701 {
5702   \newcommand*{\Glsfmtfullpl}[1]{%
5703     \teorpdfstring
5704     {\Glsxtrtitlefullpl{#1}}%
5705     {\Glsxtrinlinefullplformat{#1}-{}}%
5706   }
5707 }
5708 {
5709   \newcommand*{\Glsfmtfullpl}[1]{%
5710     \Glsxtrtitlefullpl{#1}}
5711 }

```

1.8 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

sariesExtraLang

```
5712 \newcommand*{\RequireGlossariesExtraLang}[1]{%
5713   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
5714 }
```

sariesExtraLang

```
5715 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
5716   \ProvidesFile{glossariesxtr-#1.ldf}%
5717 }
```

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.)

```
5718 \@ifpackageloaded{tracklang}
5719 {%
5720   \AnyTrackedLanguages
5721   {%
5722     \ForEachTrackedDialect{\this@dialect}{%
5723       \IfTrackedLanguageFileExists{\this@dialect}%
5724       {glossariesxtr-}% prefix
5725       {.ldf}%
5726       {%
5727         \RequireGlossariesExtraLang{\CurrentTrackedTag}%
5728       }%
5729     }%
5730   }%
5731 }%
5732 }%
5733 {}%
5734 }
5735 {}
```

Load `glossaries-extra-stylemods` if required.

```
5736 \@glsxtr@redefstyles
```

and set the style:

```
5737 \@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:

```
5738 \NeedsTeXFormat{LaTeX2e}
5739 \ProvidesPackage{glossaries-extra-stylemods}[2016/06/10 v1.05 (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`

```
5740 \newcommand*{\@glxtr@loadstyles}{%
5741 \DeclareOption*{%
5742   \IfFileExists{glossary-\CurrentOption.sty}%
5743   {\eappto\@glxtr@loadstyles{%
5744     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
5745   {\PackageError{glossaries-extra-styles}%
5746     {Unknown option '\CurrentOption'}{}}
5747 }
```

Process the package options:

```
5748 \ProcessOptions
```

Load the required packages:

```
5749 \@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`

```
5750 \providecommand{\renewglossarystyle}[2]{%
5751   \ifcsundef{@glstyle@#1}%
5752   {%
5753     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%
```

```

5754 }%
5755 {%
5756   \csdef{@glsstyle@#1}{#2}%
5757 }%
5758 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```

5759 \ifcsdef{@glsstyle@listdotted}
5760 {%
5761   \renewglossarystyle{listdotted}{%
5762     \setglossarystyle{list}%
5763     \renewcommand*{\glossentry}[2]{%
5764       \item[]\makebox[\glslistdottedwidth][l]{%
5765         \glstryitem{##1}%
5766         \glstarget{##1}{\glossentryname{##1}}%
5767         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
5768         \glossentrydesc{##1}\glspostdescription}%
5769     \renewcommand*{\subglossentry}[3]{%
5770       \item[]\makebox[\glslistdottedwidth][l]{%
5771         \glssubentryitem{##2}%
5772         \glstarget{##2}{\glossentryname{##2}}%
5773         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
5774         \glossentrydesc{##2}\glspostdescription}%
5775   }
5776 }
5777 {}

```

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.

```

5778 \ifcsdef{@glsstyle@long3col}
5779 {%
5780   \renewglossarystyle{long3col}{%
5781     \renewenvironment{theglossary}%
5782       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
5783       {\end{longtable}}%
5784     \renewcommand*{\glossaryheader}{}%
5785     \renewcommand*{\glsgroupheading}[1]{}%
5786     \renewcommand{\glossentry}[2]{%
5787       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5788       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline

```

```

5789 }%
5790 \renewcommand{\subglossentry}[3]{%
5791     &
5792     \glssubentryitem{##2}%
5793     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5794     ##3\tabularnewline
5795 }%
5796 \renewcommand*{\glsgroupskip}{%
5797     \ifglsgnogroupskip\else & &\tabularnewline\fi}%
5798 }
5799 }
5800 {}

```

Four column style:

```

5801 \ifcsdef{@glsstyle@long4col}
5802 {%
5803     \renewglossarystyle{long4col}{%
5804         \renewenvironment{theglossary}%
5805             {\begin{longtable}{llll}}%
5806             {\end{longtable}}%
5807         \renewcommand*{\glossaryheader}{}%
5808         \renewcommand*{\glsgroupheading}[1]{}%
5809         \renewcommand{\glossentry}[2]{%
5810             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5811             \glossentrydesc{##1}\glspostdescription &
5812             \glossentrysymbol{##1} &
5813             ##2\tabularnewline
5814         }%
5815         \renewcommand{\subglossentry}[3]{%
5816             &
5817             \glssubentryitem{##2}%
5818             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5819             \glossentrysymbol{##2} & ##3\tabularnewline
5820         }%
5821         \renewcommand*{\glsgroupskip}{%
5822             \ifglsgnogroupskip\else & &\tabularnewline\fi}%
5823     }
5824 }
5825 {}

```

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.

```

5826 \ifcsdef{@glsstyle@longragged3col}
5827 {%
5828     \renewglossarystyle{longragged3col}{%

```

```

5829 \renewenvironment{theglossary}%
5830     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
5831       >{\raggedright}p{\glspagelistwidth}}}%
5832     {\end{longtable}}}%
5833 \renewcommand*{\glossaryheader}{}%
5834 \renewcommand*{\glsgroupheading}[1]{}%
5835 \renewcommand{\glossentry}[2]{%
5836     \glssentryitem{##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     \ifglsgnোগroupskip\else & &\tabularnewline\fi}%
5847 }
5848 }
5849 {}

```

Four column style:

```

5850 \ifcsdef{@glstyle@altlongragged4col}
5851 {%
5852 \renewglossarystyle{altlongragged4col}{%
5853 \renewenvironment{theglossary}%
5854     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
5855       >{\raggedright}p{\glspagelistwidth}}}%
5856     {\end{longtable}}}%
5857 \renewcommand*{\glossaryheader}{}%
5858 \renewcommand*{\glsgroupheading}[1]{}%
5859 \renewcommand{\glossentry}[2]{%
5860     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5861     \glossentrydesc{##1}\glspostdescription & \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     \ifglsgnোগroupskip\else & & &\tabularnewline\fi}%
5872 }
5873 }
5874 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```

5875 \ifcsdef{@glsstyle@super3col}
5876 {%
5877   \renewglossarystyle{super3col}{%
5878     \renewenvironment{theglossary}%
5879       {\tablehead{}\tabletail{}}%
5880       \begin{supertabular}{\lp{\glsdescwidth}p{\glspagelistwidth}}}%
5881       {\end{supertabular}}}%
5882   \renewcommand*{\glossaryheader}{}%
5883   \renewcommand*{\glsgroupheading}[1]{}%
5884   \renewcommand{\glossentry}[2]{%
5885     \glstryitem{##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     \ifglsgnpgroupskip\else & &\tabularnewline\fi}%
5896 }
5897 }
5898 {}

```

Four column styles:

```

5899 \ifcsdef{@glsstyle@super4col}
5900 {%
5901   \renewglossarystyle{super4col}{%
5902     \renewenvironment{theglossary}%
5903       {\tablehead{}\tabletail{}}%
5904       \begin{supertabular}{\llll}}}%
5905       \end{supertabular}}}%
5906   \renewcommand*{\glossaryheader}{}%
5907   \renewcommand*{\glsgroupheading}[1]{}%
5908   \renewcommand{\glossentry}[2]{%
5909     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5910     \glossentrydesc{##1}\glspostdescription &
5911     \glossentrysymbol{##1} & ##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      \ifglsnogroupskip\else & & \tabularnewline\fi}%
5921    }
5922  }
5923 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

5924 \ifcsdef{@glsstyle@superragged3col}
5925 {%
5926   \renewglossarystyle{superragged3col}{%
5927     \renewenvironment{theglossary}%
5928       {\tablehead{}\tabletail}%
5929       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
5930         >{\raggedright}p{\glspagelistwidth}}}%
5931       {\end{supertabular}}}%
5932   \renewcommand*{\glossaryheader}{}%
5933   \renewcommand*{\glsgroupheading}[1]{}%
5934   \renewcommand{\glossentry}[2]{%
5935     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5936     \glossentrydesc{##1}\glspostdescription &
5937     ##2\tabularnewline
5938   }%
5939   \renewcommand{\subglossentry}[3]{%
5940     &
5941     \glssubentryitem{##2}%
5942     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5943     ##3\tabularnewline
5944   }%
5945   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
5946     &\tabularnewline\fi}%
5947 }
5948 }
5949 {}

```

Four columns:

```

5950 \ifcsdef{@glsstyle@altsuperragged4col}
5951 {%
5952   \renewglossarystyle{altsuperragged4col}{%
5953     \renewenvironment{theglossary}%
5954       {\tablehead{}\tabletail}%
5955       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
5956         >{\raggedright}p{\glspagelistwidth}}}%
5957       {\end{supertabular}}}%
5958   \renewcommand*{\glossaryheader}{}%
5959   \renewcommand{\glossentry}[2]{%
5960     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
5961     \glossentrydesc{##1}\glspostdescription &
5962     \glossentrysymbol{##1} & ##2\tabularnewline

```

```

5963 }%
5964 \renewcommand{\subglossentry}[3]{%
5965     &
5966     \glssubentryitem{##2}%
5967     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
5968     \glossentrysymbol{##2} & ##3\tabularnewline
5969 }%
5970 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & &
5971     &\tabularnewline\fi}%
5972 }
5973 }
5974 {}

```

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`.

```

5975 \ifdef{\@glsstyle@inline}
5976 {%
5977     \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
5978     Just use \glxtrpostdescription instead of \glspostdescription.
5979     \renewcommand*{\glsinlinedescformat}[3]{%
5980         \space#1\glxtrpostdescription}
5981     \renewcommand*{\glsinlinesubdescformat}[3]{%
5982         #1\glxtrpostdescription}
5983 }

```

2.8 Tree Styles

The `alttree` style is redefined to make it easier to made minor adjustments.

```

5984 \ifdef{\@glsstyle@alttree}
5985 {%

```

Only redefine this style if it's already been defined.

mbolDescLocation

```
\glxtraltrtreeSymbolDescLocation{\langle label \rangle}{\langle location list \rangle}
```

Layout the symbol, description and location for top-level entries.

```

5986 \newcommand{\glxtraltrtreeSymbolDescLocation}[2]{%
5987     {%
5988         \let\par\glxtrAltTreePar

```

```

5989      \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
5990      \glossentrydesc{#1}\glspostdescription \space #2\par
5991  }%
5992  }

```

`\trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

5993 \newlength\glxstrAltTreeIndent

```

`\lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

5994 \newcommand{\glxstrAltTreePar}{%
5995   \@@par
5996   \glxstrAltTreeSetHangIndent
5997   \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
5998 }

```

`\symbolDescLocation` `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

5999 \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
6000   \glxstralttreeSymbolDescLocation{#2}{#3}%
6001 }

```

`\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.

```

6002 \newlength\glxstrtreeindent

```

`\glxstralttreeInit` User-level initialisation for the alttree style.

```

6003 \newcommand*{\glxstralttreeInit}{%
6004   \settowidth{\glxstrtreeindent}{\glstreenamfmt{\glsgetwidestname\space}}%
6005   \glxstrAltTreeIndent=\parindent
6006 }

```

`\eglssetwidest` The original `\glsgsetwidest` only uses `\def`. This uses `\protected@csedef`.

```

6007 \newcommand*{\eglssetwidest}[2][0]{%
6008   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
6009 }

```

`\xglsgsetwidest` Like the above but uses `\protected@csxdef`.

```

6010 \newcommand*{\xglsgsetwidest}[2][0]{%
6011   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
6012 }

```

`\glsggetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

6013 \newcommand*{\glsggetwidestname}{\@glswidestname}

```


`etwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```
6014 \newcommand*\glsgetwidestsubname}[1]{%
6015   \ifcsundef{@glswidestname\romannumeral#1}%
6016     {\@glswidestname}%
6017     {\csuse{@glswidestname\romannumeral#1}}%
6018 }
```

`estTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```
6019 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

`sedTopLevelName` 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.

```
6020 \newrobustcmd*\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
6021   \dimen@=0pt\relax
6022   \gls@tmplen=0pt\relax
6023   \foralllglossaries[#1]{\@gls@type}%
6024   {%
6025     \forglsentries[\@gls@type]{\@glo@label}%
6026     {%
6027       \ifglsused{\@glo@label}%
6028       {%
6029         \ifglshasparent{\@glo@label}%
6030         {}%
6031         {%
6032           \settowidth{\dimen@}%
6033             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6034           \ifdim\dimen@>\gls@tmplen
6035             \gls@tmplen=\dimen@
6036             \eglssetwidest{\glsentryname{\@glo@label}}%
6037           \fi
6038         }%
6039       }%
6040     }%
6041   }%
6042 }%
6043 }
```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
6044 \newrobustcmd*\glsFindWidestUsedAnyName}[1][\@glo@types]{%
6045   \dimen@=0pt\relax
6046   \gls@tmplen=0pt\relax
6047   \foralllglossaries[#1]{\@gls@type}%
6048   {%
6049     \forglsentries[\@gls@type]{\@glo@label}%
6050     {%
```

```

6051     \ifglsused{\@glo@label}%
6052     {%
6053         \settowidth{\dimen@}%
6054         {\glstreenamefmt{\gl Sentryname{\@glo@label}}}%
6055         \ifdim\dimen@>\gls@tmplen
6056             \gls@tmplen=\dimen@
6057             \eglssetwidest{\gl Sentryname{\@glo@label}}%
6058         \fi
6059     }%
6060     {%
6061     }%
6062     }%
6063 }

```

FindWidestAnyName Like the above but doesn't check if the entry has been used.

```

6064 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
6065     \dimen@=0pt\relax
6066     \gls@tmplen=0pt\relax
6067     \forallglossaries[#1]{\@gls@type}%
6068     {%
6069         \forallglsentries[\@gls@type]{\@glo@label}%
6070         {%
6071             \settowidth{\dimen@}%
6072             {\glstreenamefmt{\gl Sentryname{\@glo@label}}}%
6073             \ifdim\dimen@>\gls@tmplen
6074                 \gls@tmplen=\dimen@
6075                 \eglssetwidest{\gl Sentryname{\@glo@label}}%
6076             \fi
6077         }%
6078     }%
6079 }

```

FindUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

6080 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
6081     \dimen@=0pt\relax
6082     \dimen@i=0pt\relax
6083     \dimen@ii=0pt\relax
6084     \forallglossaries[#1]{\@gls@type}%
6085     {%
6086         \forallglsentries[\@gls@type]{\@glo@label}%
6087         {%
6088             \ifglsused{\@glo@label}%
6089             {%
6090                 \ifglshasparent{\@glo@label}%
6091                 {%
6092                     \edef\@glo@parent{\csuse{glo@gl sdetoklabel}{\@glo@label}@parent}}%
6093                     \ifglshasparent{\@glo@parent}%
6094                 }%

```

```

6095         \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}}%
6096         \ifglshasparent{\@glo@parent}%
6097         {%
6098         {%
6099             \settowidth{\gls@tmplen}%
6100             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6101             \ifdim\gls@tmplen>\dimen@ii
6102             \dimen@ii=\gls@tmplen
6103             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6104             \fi
6105         }%
6106     }%
6107     {%
6108         \settowidth{\gls@tmplen}%
6109         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6110         \ifdim\gls@tmplen>\dimen@i
6111         \dimen@i=\gls@tmplen
6112         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6113         \fi
6114     }%
6115 }%
6116 {%
6117     \settowidth{\gls@tmplen}%
6118     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6119     \ifdim\gls@tmplen>\dimen@
6120     \dimen@=\gls@tmplen
6121     \eglssetwidest{\glsentryname{\@glo@label}}%
6122     \fi
6123 }%
6124 }%
6125 {}%
6126 }%
6127 }%
6128 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

6129 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
6130     \dimen@=0pt\relax
6131     \dimen@i=0pt\relax
6132     \dimen@ii=0pt\relax
6133     \foralllglossaries[#1]{\@gls@type}%
6134     {%
6135         \forglsentries[\@gls@type]{\@glo@label}%
6136         {%
6137             \ifglshasparent{\@glo@label}%
6138             {%
6139                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
6140                 \ifglshasparent{\@glo@parent}%
6141                 {%

```

```

6142      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
6143      \ifglshasparent{\@glo@parent}%
6144      {}%
6145      {%
6146        \settowidth{\gls@tmplen}%
6147          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6148        \ifdim\gls@tmplen>\dimen@ii
6149          \dimen@ii=\gls@tmplen
6150          \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6151        \fi
6152      }%
6153    }%
6154    {%
6155      \settowidth{\gls@tmplen}%
6156        {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6157      \ifdim\gls@tmplen>\dimen@i
6158        \dimen@i=\gls@tmplen
6159        \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6160      \fi
6161    }%
6162  }%
6163  {%
6164    \settowidth{\gls@tmplen}%
6165      {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6166    \ifdim\gls@tmplen>\dimen@
6167      \dimen@=\gls@tmplen
6168      \eglssetwidest{\glsentryname{\@glo@label}}%
6169    \fi
6170  }%
6171  }%
6172  }%
6173  }

```

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.

```

6174  \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
6175    \dimen@=0pt\relax
6176    \gls@tmplen=0pt\relax
6177    #2=0pt\relax
6178    \forallglossaries[#1]{\@gls@type}%
6179    {%
6180      \forglsentries[\@gls@type]{\@glo@label}%
6181      {%
6182        \ifglused{\@glo@label}%
6183        {%
6184          \settowidth{\dimen@}%
6185            {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6186          \ifdim\dimen@>\gls@tmplen
6187            \gls@tmplen=\dimen@

```

```

6188         \eglssetwidest{\glsentryname{\@glo@label}}}%
6189         \fi
6190         \settowidth{\dimen@}%
6191         {\glsentrysymbol{\@glo@label}}}%
6192         \ifdim\dimen@>#2\relax
6193             #2=\dimen@
6194         \fi
6195     }%
6196     {}%
6197 }%
6198 }%
6199 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

6200 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
6201     \dimen@=0pt\relax
6202     \gls@tmplen=0pt\relax
6203     #2=0pt\relax
6204     \forallglossaries[#1]{\@gls@type}%
6205     {%
6206         \forglsentries[\@gls@type]{\@glo@label}%
6207         {%
6208             \settowidth{\dimen@}%
6209             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6210             \ifdim\dimen@>\gls@tmplen
6211                 \gls@tmplen=\dimen@
6212                 \eglssetwidest{\glsentryname{\@glo@label}}}%
6213             \fi
6214             \settowidth{\dimen@}%
6215             {\glsentrysymbol{\@glo@label}}}%
6216             \ifdim\dimen@>#2\relax
6217                 #2=\dimen@
6218             \fi
6219         }%
6220     }%
6221 }

```

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.

```

6222 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
6223     \dimen@=0pt\relax
6224     \gls@tmplen=0pt\relax
6225     #2=0pt\relax
6226     #3=0pt\relax
6227     \forallglossaries[#1]{\@gls@type}%
6228     {%
6229         \forglsentries[\@gls@type]{\@glo@label}%

```

```

6230    {%
6231        \ifglsused{\@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        \settowidth{\dimen@}%
6245            {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6246        \ifdim\dimen@>#3\relax
6247            #3=\dimen@
6248        \fi
6249    }%
6250    {%
6251    }%
6252    }%
6253 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

6254 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
6255     \dimen@=0pt\relax
6256     \gls@tmplen=0pt\relax
6257     #2=0pt\relax
6258     #3=0pt\relax
6259     \foralllglossaries[#1]{\@gls@type}%
6260     {%
6261         \forglsentries[\@gls@type]{\@glo@label}%
6262         {%
6263             \settowidth{\dimen@}%
6264                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6265             \ifdim\dimen@>\gls@tmplen
6266                 \gls@tmplen=\dimen@
6267                 \eglssetwidest{\glsentryname{\@glo@label}}%
6268             \fi
6269             \settowidth{\dimen@}%
6270                 {\glsentrysymbol{\@glo@label}}%
6271             \ifdim\dimen@>#2\relax
6272                 #2=\dimen@
6273             \fi
6274             \settowidth{\dimen@}%
6275                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6276             \ifdim\dimen@>#3\relax

```

```

6277         #3=\dimen@
6278         \fi
6279     }%
6280 }%
6281 }

```

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.

```

6282 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
6283     \dimen@=0pt\relax
6284     \gls@tmplen=0pt\relax
6285     #2=0pt\relax
6286     \forallglossaries[#1]{\@gls@type}%
6287     {%
6288         \forglstentries[\@gls@type]{\@glo@label}%
6289         {%
6290             \ifglstused{\@glo@label}%
6291             {%
6292                 \settowidth{\dimen@}%
6293                 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6294                 \ifdim\dimen@>\gls@tmplen
6295                     \gls@tmplen=\dimen@
6296                     \eglissetwidest{\glstentryname{\@glo@label}}%
6297                 \fi
6298                 \settowidth{\dimen@}%
6299                 {\GlsXtrFormatLocationList{\glstentrynumberlist{\@glo@label}}}%
6300                 \ifdim\dimen@>#2\relax
6301                     #2=\dimen@
6302                 \fi
6303             }%
6304         }%
6305     }%
6306 }%
6307 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

6308 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
6309     \dimen@=0pt\relax
6310     \gls@tmplen=0pt\relax
6311     #2=0pt\relax
6312     \forallglossaries[#1]{\@gls@type}%
6313     {%
6314         \forglstentries[\@gls@type]{\@glo@label}%
6315         {%
6316             \settowidth{\dimen@}%
6317             {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6318             \ifdim\dimen@>\gls@tmplen
6319                 \gls@tmplen=\dimen@

```

```

6320      \eglssetwidest{\glsentryname{\@glo@label}}}%
6321      \fi
6322      \settowidth{\dimen@}%
6323      {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6324      \ifdim\dimen@>\#2\relax
6325          #2=\dimen@
6326      \fi
6327  }%
6328 }%
6329 }

```

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`.

```

6330 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
6331     \glstreeindent=\glsxtrtreetopindent\relax
6332 }

```

teTreeSubIndent

```

6333 %\cs{\glsxtrComputeTreeSubIndent}\marg{level}\marg{label}\marg{register}
6334 %\end{macrocode}
6335 % Compute the indent for the sub-entries. The first argument is the
6336 % level, the second argument is the entry label and the third
6337 % argument is the length register used to store the computed indent.
6338 % \begin{macrocode}
6339 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
6340     \ifcsundef{@glswidestname\romannumeral#1}%
6341     {%
6342         \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
6343     }%
6344     {%
6345         \settowidth{#3}{\glstreenamefmt{%
6346             \csname @glswidestname\romannumeral#1\endcsname\space}}%
6347     }%
6348 }

```

eeSetHangIndent Set `\hangindent` for top-level entries:

```

6349 \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

etSubHangIndent Set `\hangindent` for sub-entries:

```

6350 \newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `alttree`:

```

6351 \renewglossarystyle{alttree}{%
6352     \renewenvironment{theglossary}%
6353     {%
6354         \glsxtralttreeInit
6355         \def\@gls@prevlevel{-1}%

```



```

6356     \mbox{}\par}%
6357     {\par}%
6358 \renewcommand*\glossaryheader{}%
6359 \renewcommand*\glsgroupheading[1]{}%
6360 \renewcommand\glossentry[2]{%
6361     \ifnum\@gls@prevlevel=0\relax
6362     \else
6363         \glxtrComputeTreeIndent{##1}%
6364     \fi
6365     \parindent\glstreeindent
6366     \glxtrAltTreeSetHangIndent
6367     \makebox[Opt][r]%
6368     {%
6369         \glstreenamebox{\glstreeindent}%
6370         {%
6371             \glssentryitem{##1}%
6372             \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
6373         }%
6374     }%
6375     \glxtralttreeSymbolDescLocation{##1}{##2}%
6376     \def\@gls@prevlevel{0}%
6377 }
6378 \renewcommand\subglossentry[3]{%
6379     \ifnum##1=1\relax
6380         \glssubentryitem{##2}%
6381     \fi
6382     \ifnum\@gls@prevlevel=##1\relax
6383     \else
6384         \glxtrComputeTreeSubIndent{##1}{##2}{\@gls@tmplen}%
6385         \ifnum\@gls@prevlevel<##1\relax
6386             \setlength\glstreeindent\@gls@tmplen
6387             \addtolength\glstreeindent\parindent
6388             \parindent\glstreeindent
6389         \else
6390             \ifnum\@gls@prevlevel=0\relax
6391                 \glxtrComputeTreeIndent{##2}%
6392             \else
6393                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
6394             \fi
6395             \addtolength\parindent{-\glstreeindent}%
6396             \setlength\glstreeindent\parindent
6397         \fi
6398     \fi
6399     \glxtrAltTreeSetSubHangIndent{##1}%
6400     \makebox[Opt][r]{\glstreenamebox{\@gls@tmplen}{%
6401         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
6402     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
6403     \def\@gls@prevlevel{##1}%
6404 }%

```

```

6405 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
6406 }
6407 }%
6408 {%

```

Assume the style isn't required if it hasn't already been defined.

```

6409 }

```

Reset the default style

```

6410 \ifx\@glossary@default@style\relax
6411 \else
6412 \setglossarystyle{\@glxtr@current@style}
6413 \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)		\@Glsymbol@: added redefinition	28
General: Initial experimental release	4	\@Glsymbolplural@: added redefini-	
0.2 (2015-11-30)		tion	28
\Glsfmtshort: new	162	\@Glstext@: added redefinition	24
\glsfmtshort: new	162	\@Glsuseri@: added redefinition	28
\Glsfmtshortpl: new	163	\@Glsuserii@: added redefinition	29
\glsfmtshortpl: new	162	\@Glsuseriii@: added redefinition	29
short: switched inline full form to short		\@Glsuseriv@: added redefinition	29
(long)	129	\@Glsuserv@: added redefinition	30
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	30
\@ACRlong: added redefinition	33	\@acrlong: added redefinition	32
\@ACRlongpl: added redefinition	34	\@acrlongpl: added redefinition	33
\@ACRshort: added redefinition	31	\@acrshort: added redefinition	30
\@ACRshortpl: added redefinition	32	\@acrshortpl: added redefinition	31
\@Acrlong: added redefinition	33	\@gls@field@link: added optional ar-	
\@Acrlongpl: added redefinition	34	gument	23
\@Acrshort: added redefinition	31	\@glsdescplural@: added redefinition .	27
\@Acrshortpl: added redefinition	32	\@glsfirst@: added redefinition	24
\@GLSdesc@: added redefinition	27	\@glsfirstplural@: added redefinition	25
\@GLSdescplural@: added redefinition .	27	\@glsplural@: added redefinition	25
\@GLSfirst@: added redefinition	25	\@glssymbolplural@: added redefini-	
\@GLSfirstplural@: added redefinition	26	tion	28
\@GLSname@: added redefinition	26	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	25	new	67
\@GLSsymbol@: added redefinition	28	\@glsxtr@field@linkdefs: new	23
\@GLSsymbolplural@: added redefini-		\@glsxtr@insertdots: new	102
tion	28	\@print@glossary: added redefinition .	64
\@GLStext@: added redefinition	24	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	29	\abbrvdefaultfont	106
\@GLSuserii@: added redefinition	29	\glsaccessdesc: new	72
\@GLSuseriii@: added redefinition	29	\glsaccessdescplural: new	73
\@GLSuseriv@: added redefinition	30	\glsaccessfirst: new	70
\@GLSuserv@: added redefinition	30	\glsaccessfirstplural: new	70
\@GLSuservi@: added redefinition	30	\Glsaccesslong: new	75
\@Glsdesc@: added redefinition	27	\glsaccesslong: new	75
\@Glsdescplural@: added redefinition .	27	\glsaccessname: new	68
\@Glsfirst@: added redefinition	24	\glsaccessplural: new	69
\@Glsfirstplural@: added redefinition	25	\Glsaccessshort: new	74
\@Glsname@: added redefinition	26	\glsaccessshort: new	73
\@Glsplural@: added redefinition	25	\Glsaccessshortpl: new	74

\glsaccesssshortpl: new	74	\cGLSpl: new	50
\glsaccessssymbol: new	71	\cGLSpl@: new	50
\glsaccessssymbolplural: new	72	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	69	new	45
\glstentryfmt: added check for short ..	22	\cGLS: new	49
\glslongpltok: new	102	\cGLSformat: new	50
\glsshortpltok: new	102	\cGLSpl: new	50
\glsxtrdiscardperiod: added check		\cGLSplformat: new	50
for plural	99	\GlossariesExtraWarningNoLine:	
\GLSxtrlongpl: new	115	new	6
\Glsxtrlongpl: new	115	\glsenableentrycount: new	45
\glsxtrlongpl: new	114	\glsfirstabbrvdefaultfont: new ..	106
\glsxtrNoGlossaryWarning: new	9	\glsfirstlongdefaultfont: new ...	106
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirst: new	164
new	99	\glsfmtfirst: new	164
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtfirstpl: new	165
new	99	\glsfmtfirstpl: new	165
\glsxtrpostlinkendsentence: new ..	98	\glsfmtplural: new	164
\GLSxtrshortpl: new	114	\glsfmtplural: new	163
\Glsxtrshortpl: new	113	\Glsfmtshort: changed to use	
\glsxtrshortpl: new	113	\Glsxtrtitleshort	162
short-long-desc: fixed name to use		renamed from \Glstentryfmtshort ..	162
\glslabeltok	124	\glsfmtshort: changed to use	
\newabbreviation: fixed family name in		\glsxtrtitleshort	162
\setkeys	103	renamed from \glstentryfmtshort ..	162
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	123	\Glsxtrtitleshortpl	163
0.4 (2015-12-03)		renamed from \Glstentryfmtshortpl ..	163
\@glsxtr@doabbreviationsdef: added		\glsfmtshortpl: changed to use	
redefinition of \acronymtype	6	\glsxtrtitleshortpl	162
\Glsfmtshort: changed to use		renamed from \glstentryfmtshortpl ..	162
\Glsxtrshort	162	\Glsfmttext: new	163
\glsfmtshort: changed to use		\glsfmttext: new	163
\glsxtrshort	162	\glshasattribute: new	80
\Glsfmtshortpl: changed to use		\glshascategoryattribute: new	79
\glsxtrshortpl	163	\GlsXtrEnableEntryCounting: new ..	44
\glsfmtshortpl: changed to use		\glsxtrifcounttrigger: new	47
\glsxtrshortpl	162	\glsxtrscfont: new	133
\glsxtrifemptyglossary: new	11	\glsxtrscsuffix: new	134
\glsxtrnewnumber: added extra argu-		\glsxtrsmfont: new	137
ment	84	\glsxtrsmsuffix: new	138
\glsxtrnewsymbol: added extra argu-		short-em: new	144
ment	83	short-em-desc: new	145
\MakeAcronymsAbbreviations: set the		short-em-footnote: new	146
default type to \acronymtype	58	short-em-long: new	143
\newterm: fixed name argument	83	short-em-long-desc: new	143
0.5 (2015-12-07)		short-em-postfootnote: new	147
\cGLS: new	49	short-sc-footnote: new	137
\cGLS@: new	50	short-sc-postfootnote: new	137

short-sm: new	139	\glxtrheadtext: now uses headuc at-tribute	155
short-sm-desc: new	139	short-long: switch off regular attribute if set	123
short-sm-footnote: new	140	short-long-desc: switch off regular at-tribute if set	125
short-sm-long: new	138	long-short: switch off regular attribute if set	122
short-sm-long-desc: new	138	long-short-desc: switch off regular at-tribute if set	123
short-sm-postfootnote: new	140	footnote: switch off regular attribute if set	125
long-noshort-em: new	145	postfootnote: switch off regular at-tribute if set	127
long-noshort-em-desc: new	146		
long-noshort-sm: new	139	0.5.2 (2015-12-08)	
long-noshort-sm-desc: new	140	\@GLSdesc@: added accessibility support	27
long-short-em: new	141	\@GLSdescplural@: added accessibility support	27
long-short-em-desc: new	141	\@GLSfirst@: added accessibility sup-port	25
long-short-sm: new	138	\@GLSfirstplural@: added accessibility support	26
long-short-sm-desc: new	138	\@GLSname@: added accessibility support	26
0.5.1 (2015-12-02)		\@GLSplural@: added accessibility sup-port	25
\Glsaccesstext: new	69	\@GLSsymbol@: added accessibility sup-port	28
0.5.1 (2015-12-07)		\@GLSsymbolplural@: added accessibil-ity support	28
\@glxtr@doaccsupp: new	9	\@GLStext@: added accessibility support	24
General: removed \ifglxtruseuchead	153	\@GLSdesc@: added accessibility support	27
\Glsaccessdesc: new	72	\@GLSdescplural@: added accessibility support	27
\Glsaccessdescplural: new	73	\@Glsfirst@: added accessibility sup-port	24
\Glsaccessfirst: new	70	\@Glsfirstplural@: added accessibility support	25
\Glsaccessfirstplural: new	71	\@Glsname@: add accessibility support ..	26
\Glsaccessname: new	68	\@Glsplural@: added accessibility sup-port	25
\Glsaccessplural: new	70	\@Glsymbol@: added accessibility sup-port	28
\Glsaccesssymbol: new	71	\@Glsymbolplural@: added accessibil-ity support	28
\Glsaccesssymbolplural: new	72	\@Glstext@: added accessibility support	24
\Glsxtrheadfirst: now uses headuc at-tribute	157	\@glstdesc@: added accessibility support	26
\glxtrheadfirst: now uses headuc at-tribute	157	\@glstdescplural@: added accessibility support	27
\Glsxtrheadfirstplural: now uses headuc attribute	158		
\glxtrheadfirstplural: now uses headuc attribute	157		
\Glsxtrheadplural: now uses headuc attribute	156		
\glxtrheadplural: now uses headuc attribute	156		
\Glsxtrheadshort: now uses headuc at-tribute	154		
\glxtrheadshort: now uses headuc at-tribute	153		
\Glsxtrheadshortpl: now uses headuc attribute	155		
\glxtrheadshortpl: now uses headuc attribute	154		
\Glsxtrheadtext: now uses headuc at-tribute	155		

\@glsfirst@: added accessibility support	24	\mfu@checkword@do: added	96
\@glsfirstplural@: added accessibility support	25	\setabbreviationstyle: added check for post-definition style switch	118
\@glsname@: added accessibility support	26	0.5.3 (2015-12-09)	
\@glsplural@: added accessibility support	25	\@glsxtr@autoindex@at: new	93
\@glsymbol@: added accessibility support	27	\@glsxtr@autoindex@encap: new	93
\@glsymbolplural@: added accessibility support	28	\@glsxtr@autoindex@esc: new	93
\@glstext@: added accessibility support	24	\@glsxtr@autoindex@level: new	93
\@glsxtr@activate@initialtagging: new	97	\@glsxtr@autoindex@setname: new ..	91
\@glsxtr@do@titlecaps@warn: new ..	97	\@glsxtr@doabbreviationsdef: new ..	6
\@glsxtr@tag: new	97	General: removed \GlsXtrNoGlsWarningNoAutoMakeMain	66
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	68	\glsdescwidth: added	19
removed \glsxtrabbrvfmt	116	\glspagelistwidth: added	20
\glossaryentrynumbers: added	20	\glsxtrdoautoindexname: new	91
\Glossentrydesc: added	95	\glsxtrpostnamehook: new	90
\Glossentryname: added	89	\if@glsxtr@format@override: new ..	90
\Glossentrysymbol: added	95	\ProvidesGlossariesExtraLang: new	168
\glossentrysymbol: added	95	\RequireGlossariesExtraLang: new	168
\GLSaccessdesc: new	73, 77	0.5.4 ()	
\GLSaccessdescplural: new	73, 78	\@glsxtr@setentryunitcountunsetattr: new	56
\GLSaccessfirst: new	70, 76	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new	71, 77	\@newglossaryentry@defunitcounters: new	51
\GLSaccesslong: new	75, 78	\@GLSxtr@p@acrlong@: new	43
\GLSaccesslongpl: new	75, 79	\@GLSxtr@p@acrlongpl@: new	43
\Glsaccesslongpl: new	75	\@GLSxtr@p@acrshort@: new	43
\glsaccesslongpl: new	75	\@GLSxtr@p@acrshortpl@: new	43
\GLSaccessname: new	69, 76	\@GLSxtr@p@long@: new	42
\GLSaccessplural: new	70, 76	\@GLSxtr@p@longpl@: new	43
\GLSaccesssshort: new	74, 78	\@GLSxtr@p@plural@: new	41
\GLSaccesssshortpl: new	74, 78	\@GLSxtr@p@short@: new	42
\GLSaccessssymbol: new	71, 77	\@GLSxtr@p@shortpl@: new	42
\GLSaccessssymbolplural: new ...	72, 77	\@GLSxtr@p@text@: new	41
\GLSaccessstext: new	69, 76	\@GlsXtrEnableOnTheFly: new	16
\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here	22	\@Glsxtr: new	16
\GlsXtrEnableInitialTagging: new ..	95	\@Glsxtr@p@acrlong@: new	43
\glsxtrfieldtitlecase: new	84	\@Glsxtr@p@acrlongpl@: new	43
\GlsXtrFormatLocationList: new ...	20	\@Glsxtr@p@acrshort@: new	43
\glsxtrnewabbrevpresetkeyhook: new	104	\@Glsxtr@p@acrshortpl@: new	43
\glsxtrtagfont: new	97	\@Glsxtr@p@long@: new	42
\KV@printgloss@nonumberlist: added	22	\@Glsxtr@p@longpl@: new	43
		\@Glsxtr@p@plural@: new	41
		\@Glsxtr@p@short@: new	42
		\@Glsxtr@p@shortpl@: new	42
		\@Glsxtr@p@text@: new	41
		\@Glsxtrpl: new	17
		\@alt@gls@hyp@opt: new	39

\@gls@alt@hyp@opt: new	39	\GlsXtrEnableEntryUnitCounting:	
\@gls@alt@hyp@opt@char: new	39	new	56
\@gls@alt@hyp@opt@keys: new	39	\GlsXtrEnableOnTheFly: new	15
\@gls@increment@currunitcount:		\Glsxtrpl: new	17
new	52	\glsxtrpl: new	17
\@gls@local@increment@currunitcount:		\glsxtrpostlocalreset: new	44
new	52	\glsxtrpostlocalunset: new	44
\@gls@setdefault@glslink@opts:		\glsxtrpostreset: new	44
new	37	\glsxtrpostunset: new	44
\@glsxtr: new	16	\glsxtrprotectlinks: new	40
\@glsxtr@addunitcounter: new	51	\GlsXtrSetAltModifier: new	39
\@glsxtr@currunitcount: new	53	\GlsXtrSetDefaultGlsOpts: new	38
\@glsxtr@ifunitcounter: new	51	\glsxtrstarflywarn: new	16
\@glsxtr@p@acrlong@: new	43	\GlsXtrWarning: new	17
\@glsxtr@p@acrlongpl@: new	43	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshort@: new	43	disables \setacronymstyle	58
\@glsxtr@p@acrshortpl@: new	43	1.0 (2016-01-24)	
\@glsxtr@p@long@: new	42	\@glsxtr@autoindexcrossrefs: new ..	6
\@glsxtr@p@longpl@: new	43	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@plural@: new	41	new	62
\@glsxtr@p@short@: new	41	\@glsxtr@idx@entrynumberlist: new	63
\@glsxtr@p@shortpl@: new	42	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@p@text@: new	41	new	62
\@glsxtr@prevunitcount: new	53	\@glsxtr@noidx@entrynumberlist:	
\@glsxtr@unitcountlist: new	51	new	63
\@glsxtrpl: new	17	\@glsxtr@noidx@numberlistloop:	
\@newglossaryentryposthook: added		new	63
empty see value if not set and added		\@glsxtr@reg@glosslist: new	59
‘see’ to field key map	13	\makeglossaries: new	60
\@sGlsXtrEnableOnTheFly: new	15	1.01 (2016-02-02)	
\cGlsformat: added	50	\glsxtrdiscardperiod: added check	
\cglformat: added	50	for first use	99
\cGlsplformat: added	51	short-desc: fixed typo in \glsxtrinlinefullformat	
\cglsplformat: added	50	and added missing second argument	130
\glsdisablehyper: added	40	1.02 (2016-04-25)	
\glsdohyperlink: added	40	\@glsxtr@current@style: new	18
\glsdonohyperlink: added	40	\Glsfmtfull: new	167
\glsenableentryunitcount: new	53	\glsfmtfull: new	166
\glsasattribute: added check for en-		\Glsfmtfullpl: new	167
try’s existence	80	\glsfmtfullpl: new	167
\glsifattribute: added check for en-		\Glsfmtlong: new	165
try’s existence	81	\glsfmtlong: new	165
\glspostlinkhook: added existence		\Glsfmtlongpl: new	166
check	98	\glsfmtlongpl: new	166
\Glsxtr: new	16	\Glsxtrheadfull: new	161
\glsxtr: new	16	\glsxtrheadfull: new	160
\glsxtrcat: new	16	\Glsxtrheadfullpl: new	161
\glsxtrdowrglossaryhook: new	39	\glsxtrheadfullpl: new	160
		\Glsxtrheadlong: new	159

\glxtrheadlong: new	158	\@GLStext@: set abbreviation and regular format	24
\Glsxtrheadlongpl: new	160	\@GLSuseri@: set regular format	29
\glxtrheadlongpl: new	159	\@GLSuserii@: set regular format	29
\glxtrtitlefull: new	161	\@GLSuseriii@: set regular format	29
\Glsxtrtitlefull: new	160	\@GLSuseriv@: set regular format	30
\glxtrtitlefullpl: new	162	\@GLSuseriv@: set regular format	30
\glxtrtitlefullpl: new	161	\@GLSuservi@: set regular format	30
\Glsxtrtitlelong: new	159	\@Glsdesc@: set abbreviation and regular format	27
\glxtrtitlelong: new	159	\@Glsdescplural@: set abbreviation and regular format	27
\Glsxtrtitlelongpl: new	160	\@Glsfirst@: set abbreviation and regu- lar format	24
\glxtrtitlelongpl: new	159	\@Glsfirstplural@: set abbreviation and regular format	25
\ifglxtrininsertinside: new	121	\@Glsname@: set abbreviation and regular format	26
postfootnote: added redef of \glxtrsetupfulldefs	127	\@Glsplural@: set abbreviation and reg- ular format	25
stylemods: new	9	\@Glsymbol@: set regular format	28
1.03 (2016-04-27)		\@Glsymbolplural@: set regular format	28
\@GLSfirstplural@: bug fix: misspelt cs name	26	\@Glstext@: set abbreviation and regular format	24
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	25	\@Glsuseri@: set regular format	28
\@Glsfirstplural@: bug fix: misspelt cs name	25	\@Glsuserii@: set regular format	29
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural	25	\@Glsuseriii@: set regular format	29
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	25	\@Glsuseriv@: set regular format	29
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	159	\@Glsuseriv@: set regular format	30
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	154	\@Glsuservi@: set regular format	30
1.04 (2015-04-30)		\@gls@preglossaryhook: added check for entry's existence	97
short-em-footnote: renamed from "footnote-em"	146	\@Glsdesc@: set abbreviation and regular format	26
1.04 (2016-05-02)		\@Glsdescplural@: set abbreviation and regular format	27
\@@glxtrpostloctag: new	22	\@glsfirst@: set abbreviation and regu- lar format	24
\@GLSdesc@: set abbreviation and regular format	27	\@glsfirstplural@: set abbreviation and regular format	25
\@GLSdescplural@: set abbreviation and regular format	27	\@Glsname@: set abbreviation and regular format	26
\@GLSfirst@: set abbreviation format ..	25	\@glsplural@: set abbreviation and reg- ular format	25
\@GLSfirstplural@: set abbreviation and regular format	26	\@glssymbol@: set regular format	27
\@GLSname@: set abbreviation and regular format	26	\@glssymbolplural@: set regular format	28
\@GLSplural@: set abbreviation and reg- ular format	25	\@glstext@: set abbreviation and regular format	24
\@GLSsymbol@: set regular format	28		
\@GLSsymbolplural@: set regular format	28		

\@glstr@deprecated@abbrstyle:		
new	120	
\@glstr@do@style: new	10	
\@glstr@doloctag: new	22	
\@glstr@idx@entrynumberlist:		
switched from \let to \newcommand	63	
\@glstr@pagetag: new	21	
\@glstr@pagetag: new	21	
\@glstr@preloctag: new	22	
\@glstr@postloctag: new	22	
\@glstr@preloctag: new	21	
\glossentrydesc: added glossdescfont		
attribute check	85	
\Glossentryname: added glossnamefont		
attribute check	89	
\glossentryname: added glossnamefont		
attribute check	86	
moved post name hook inside condi-		
tion	89	
\glsabbrvemfont: new	141	
\glsabbrvuserfont: new	148	
\glsfirstabbrvemfont: new	141	
\glsfirstabbrvuserfont: new	148	
\glsfirstlongemfont: new	141	
\glsfirstlonguserfont: new	148	
\glsifnotregularcategory: new	81	
\glslongdefaultfont: new	106	
\glslongemfont: new	141	
\glslongfont: new	106	
\glslonguserfont: new	148	
\glstrassignfieldfont: new	23	
\GlsXtrEnablePreLocationTag: new .	21	
\glstrfirstscfont: new	134	
\glstrfirstsmfont: new	137	
\glstrlongshortdescsort: new ...	123	
\glstrpostnamehook: added category		
check	90	
\glstrregularfont: new	23	
\glstruserfield: new	147	
\glstruserparen: new	147	
\glstrusersuffix: new	148	
\GlsXtrWarnDeprecatedAbbrStyle:		
new	120	
short-em-long-em: new	143	
short-em-long-em-desc: new	144	
short-em-nolong: new	144	
short-em-nolong-desc: new	145	
short-em-postfootnote: renamed from		
“postfootnote-em”	147	
short-footnote: new	127	
short-long-user: new	149	
short-long-user-desc: new	150	
short-nolong: new	130	
short-nolong-desc: new	131	
short-postfootnote: new	128	
short-sc-footnote: renamed from		
“footnote-sc”	137	
short-sc-nolong: new	135	
short-sc-nolong-desc: new	136	
short-sc-postfootnote: renamed from		
“postfootnote-sc”	137	
short-sm-footnote: renamed from		
“footnote-sm”	140	
short-sm-nolong: new	139	
short-sm-nolong-desc: new	139	
short-sm-postfootnote: renamed from		
“postfootnote-sm”	140	
\letabbreviationstyle: new	120	
\newabbreviationstyle: bug fix: cor-		
rected test for existence	119	
long-em-noshort-em: new	145	
long-em-noshort-em-desc: new	146	
long-em-short-em: new	142	
long-em-short-em-desc: new	142	
long-noshort: new	133	
long-noshort-desc: new	133	
long-noshort-em: renamed from “long-		
em”	145	
long-noshort-em-desc: renamed from		
“long-desc-em”	146	
long-noshort-sc: renamed from “long-		
sc”	136	
long-noshort-sc-desc: renamed from		
“long-desc-sc”	136	
long-noshort-sm: renamed from “long-		
sm”	139	
long-noshort-sm-desc: renamed from		
\long-desc-sm	140	
long-short-user: new	148	
long-short-user-desc: new	149	
\renewabbreviationstyle: new	120	
style: new	10	
1.05 (2016-06-10)		
\glsfirstlongfootnotefont: new ..	125	
\glslongfootnotefont: new	125	
short-em-long: fixed incorrect font used		
by long form	143	

1.05 (??)

\eglssetwidest:new	176	\glsFindWidestUsedTopLevelName:	
\glsFindWidestAnyName:new	178	new	177
\glsFindWidestAnyNameLocation:		\glsgetwidestname:new	176
new	183	\glsgetwidestsubname:new	177
\glsFindWidestAnyNameSymbol:new	181	\glxtrAltTreeIndent:new	176
\glsFindWidestAnyNameSymbolLocation:		\glxtralttreeInit:new	176
new	182	\glxtrAltTreePar:new	176
\glsFindWidestLevelTwo:new	179	\glxtrAltTreeSetHangIndent:new	184
\glsFindWidestUsedAnyName:new ..	177	\glxtrAltTreeSetSubHangIndent:	
\glsFindWidestUsedAnyNameLocation:		new	184
new	183	\glxtralttreeSubSymbolDescLocation:	
\glsFindWidestUsedAnyNameSymbol:		new	176
new	180	\glxtralttreeSymbolDescLocation:	
\glsFindWidestUsedAnyNameSymbolLocation:		new	175
new	181	\glxtrComputeTreeIndent:new ...	184
\glsFindWidestUsedLevelTwo:new .	178	\glxtrComputeTreeSubIndent:new	184
		\glxtrtreetopindent:new	176
		\xglsetwidest:new	176

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	
\.	98, 99, 175
\@@cGLS@	46, 54, 55
\@@cGLSpl@	46, 55
\@@cGLspl@	46, 54
\@@cglS@	46, 54
\@@cglSpl@	46, 48, 54
\@@glo@assign@sortkey	62
\@@glo@no@assign@sortkey	62
\@@glslocalreset	44
\@@glslocalunset	44
\@@glsreset	44
\@@glsunset	44
\@@glsxtr@autoindex@escspch	93, 94
\@@glsxtr@checkspch	92, 94, 95
\@@glsxtr@disabledflycommand	18
\@@glsxtr@postloctag	21
\@@glsxtr@preloctag	21
\@newglossaryentry@defcounters	45
\@newglossaryentry@defunitcounters	53
\@par	176
\@ACRlong	41
\@ACRlongpl	41
\@ACRshort	41
\@ACRshortpl	41
\@Acrlong	41
\@Acrlongpl	41
\@Acrshort	41
\@Acrshortpl	41
\@GLS@	40, 49, 50
\@GLSdesc@	27
\@GLSpl@	40, 49, 50
\@GLSplural@	41
\@GLSsymbol@	28
\@GLStext@	41
\@GLSxtr@full	108
\@GLSxtr@fullpl	109
\@GLSxtr@p@acrlong@	41
\@GLSxtr@p@acrlongpl@	41
\@GLSxtr@p@acrshort@	41
\@GLSxtr@p@acrshortpl@	41
\@GLSxtr@p@long@	40
\@GLSxtr@p@longpl@	41
\@GLSxtr@p@plural@	40
\@GLSxtr@p@short@	40
\@GLSxtr@p@shortpl@	41
\@GLSxtr@p@text@	40
\@GLSxtrlong	40, 112
\@GLSxtrlongpl	41, 115

<code>\@Glsxtrpl</code>	17, 18	<code>\@gls@checkedmkidx</code>	92, 94
<code>\@Glsxtrshort</code>	40, 110	<code>\@gls@checkmkidxchars</code>	91
<code>\@Glsxtrshortpl</code>	41, 113	<code>\@gls@codepage</code>	64
<code>\@acrlong</code>	41	<code>\@gls@declareoption</code>	4
<code>\@acrlongpl</code>	41	<code>\@gls@doautomake</code>	62
<code>\@acrshort</code>	41	<code>\@gls@encapchar</code>	92
<code>\@acrshortpl</code>	41	<code>\@gls@entry@count</code>	46, 47
<code>\@alt@gls@hyp@opt</code>	39	<code>\@gls@entry@field</code>	35, 46
<code>\@auxout</code>	22, 47, 55, 60, 64	<code>\@gls@entry@unitcount</code>	55
<code>\@cGLS</code>	49	<code>\@gls@field@font</code>	24–30
<code>\@cGLS@</code>	46, 49, 54, 55	<code>\@gls@field@link</code>	24–30, 35, 36
<code>\@cGLSpl</code>	50	<code>\@gls@hyp@opt</code>	35, 36, 39, 49, 50, 106–115
<code>\@cGLSpl@</code>	46, 50, 55	<code>\@gls@hyp@opt@cs</code>	39
<code>\@cGLspl@</code>	46, 54	<code>\@gls@increment@currcount</code>	46
<code>\@cgls@</code>	46, 49, 54	<code>\@gls@increment@currunitcount</code>	54
<code>\@cglspl@</code>	46, 54	<code>\@gls@keymap</code>	13, 35
<code>\@disable@onlypremakeg</code>	60	<code>\@gls@label</code>	38, 39, 118
<code>\@do@auxoutstuff</code>	64	<code>\@gls@levelchar</code>	92
<code>\@do@newglossaryentry</code>	57, 58, 104	<code>\@gls@link</code>	23, 31–34, 107–116
<code>\@empty</code>	23, 31–34, 92, 107–116	<code>\@gls@link@checkfirsthyper</code>	59
<code>\@end@glxstr@addunused</code>	14	<code>\@gls@link@nocheckfirsthyper</code>	23, 30–34, 107–116
<code>\@endfortrue</code>	119	<code>\@gls@local@increment@currcount</code>	46
<code>\@firstofone</code>	24, 85, 86, 91, 96	<code>\@gls@local@increment@currunitcount</code>	54
<code>\@firstofthree</code>	23, 31–34, 39, 107, 108, 110, 111, 113, 115	<code>\@gls@loclist</code>	62, 63
<code>\@firstoftwo</code>	24–28, 32, 34, 37, 39, 100, 101, 107–109, 113–116	<code>\@gls@longpl</code>	102–104
<code>\@for</code>	10, 14, 45, 56, 60, 62, 84, 96	<code>\@gls@noidx@nosanitizesort</code>	61
<code>\@glo@assign@sortkey</code>	61	<code>\@gls@noidx@sanitizesort</code>	61
<code>\@glo@category</code>	51	<code>\@gls@noidx@loclist@finalsep</code>	62
<code>\@glo@countunit</code>	51	<code>\@gls@noidx@loclist@prev</code>	62
<code>\@glo@default@sorttype</code>	62	<code>\@gls@noidx@loclist@sep</code>	62
<code>\@glo@label</code>	13, 14, 35, 177–184	<code>\@gls@org@glsnoidxdisplayloc</code>	63
<code>\@glo@name</code>	91	<code>\@gls@org@glsseeformat</code>	63
<code>\@glo@parent</code>	178–180	<code>\@gls@preglossaryhook</code>	96
<code>\@glo@see</code>	13, 14	<code>\@gls@prevlevel</code>	184, 185
<code>\@glo@sort</code>	91	<code>\@gls@quotechar</code>	92
<code>\@glo@sorttype</code>	62	<code>\@gls@reference</code>	60
<code>\@glo@thisvalue</code>	147	<code>\@gls@setdefault@glslink@opts</code>	38
<code>\@glo@tmp</code>	35	<code>\@gls@short</code>	103
<code>\@glo@type</code>	13, 57, 60, 61, 64, 67, 68	<code>\@gls@shortpl</code>	102–104
<code>\@glo@types</code>	82, 177–183	<code>\@gls@tmpb</code>	94
<code>\@glossary@default@style</code>	18, 19, 186	<code>\@gls@type</code>	62, 118, 177–183
<code>\@gls@</code>	40, 48, 49	<code>\@gls@write@entrycounts</code>	46
<code>\@gls@actualchar</code>	92	<code>\@gls@write@entryunitcounts</code>	55
<code>\@gls@alt@hyp@opt</code>	39	<code>\@gls@write@entryunitcounts@do</code>	56
<code>\@gls@alt@hyp@opt@char</code>	39	<code>\@gls@abbrv@current@abbreviation</code>	103, 116
<code>\@gls@alt@hyp@opt@keys</code>	39	<code>\@gls@acronymlists</code>	57
<code>\@gls@automake</code>	62	<code>\@gls@entry</code>	47, 55, 56
		<code>\@gls@link</code>	40

<code>\@glsnumberformat</code>	90, 91	<code>\@glsxtr@entrycount@org@localreset</code> ..	46
<code>\@glsorder</code>	60	<code>\@glsxtr@entrycount@org@localunset</code> ..	46
<code>\@glspl@</code>	40, 48, 49	<code>\@glsxtr@entrycount@org@reset</code>	46
<code>\@glsplural@</code>	41	<code>\@glsxtr@entrycount@org@unset</code>	46
<code>\@gls punc@token</code>	100	<code>\@glsxtr@entryunitcount@org@localreset</code>	
<code>\@glsstyle@alttree</code>	175	54
<code>\@glsstyle@inline</code>	175	<code>\@glsxtr@entryunitcount@org@localunset</code>	
<code>\@glsstyle@listdotted</code>	170	54
<code>\@gls target</code>	40	<code>\@glsxtr@entryunitcount@org@reset</code> ..	54
<code>\@gls text@</code>	41	<code>\@glsxtr@entryunitcount@org@unset</code> ..	54
<code>\@gls widestname</code>	176, 177, 184	<code>\@glsxtr@field@linkdefs</code>	23
<code>\@glsxtr</code>	16, 18	<code>\@glsxtr@format@overridefalse</code>	90
<code>\@glsxtr@abbreviationsdef</code>	7, 10, 11	<code>\@glsxtr@format@overridetrue</code>	91
<code>\@glsxtr@activate@initialtagging</code>	96, 97	<code>\@glsxtr@foundinlist</code>	101
<code>\@glsxtr@addunitcounter</code>	51	<code>\@glsxtr@full</code>	106
<code>\@glsxtr@addunusedxrefs</code>	14	<code>\@glsxtr@fullpl</code>	108
<code>\@glsxtr@attrval</code>	85–91	<code>\@glsxtr@glossdescfont</code>	85, 86
<code>\@glsxtr@autoindex@at</code>	91–93	<code>\@glsxtr@glossnamefont</code>	87–90
<code>\@glsxtr@autoindex@doextra@esc</code>	91	<code>\@glsxtr@gobbleto@endescspch</code>	94
<code>\@glsxtr@autoindex@encap</code>	91–93	<code>\@glsxtr@idx@displaynumberlist</code>	60
<code>\@glsxtr@autoindex@esc</code>	92, 94, 95	<code>\@glsxtr@idx@entrynumberlist</code>	61
<code>\@glsxtr@autoindex@escat</code>	92, 93	<code>\@glsxtr@ifcsstart</code>	15
<code>\@glsxtr@autoindex@escencap</code>	92, 93	<code>\@glsxtr@ifpunctoken</code>	101
<code>\@glsxtr@autoindex@esclevel</code>	92, 93	<code>\@glsxtr@ifunitcounter</code>	51
<code>\@glsxtr@autoindex@escquote</code>	92, 94	<code>\@glsxtr@insert@dots</code>	102
<code>\@glsxtr@autoindex@level</code>	92, 93	<code>\@glsxtr@insert@dots@next</code>	102, 103
<code>\@glsxtr@autoindex@setname</code>	91	<code>\@glsxtr@insertdots</code>	103
<code>\@glsxtr@autoindex@crossrefs</code>	5, 13	<code>\@glsxtr@label</code>	14, 84
<code>\@glsxtr@cat</code>	45, 56, 57, 96	<code>\@glsxtr@loadstyles</code>	169
<code>\@glsxtr@csname</code>	52, 54	<code>\@glsxtr@noidx@displaynumberlist</code> ...	61
<code>\@glsxtr@current@style</code>	19, 186	<code>\@glsxtr@noidx@entrynumberlist</code>	61
<code>\@glsxtr@currentunitcount</code>	52, 54	<code>\@glsxtr@noidx@numberlistloop</code>	61
<code>\@glsxtr@currunitcount</code>	53, 55	<code>\@glsxtr@notfoundinlist</code>	101
<code>\@glsxtr@declareoption</code>	4, 6, 7, 9	<code>\@glsxtr@optlist</code>	17, 18
<code>\@glsxtr@defaultnoglossarywarning</code> ...	9	<code>\@glsxtr@org@Glsxtrtitlefirst</code> ..	152, 153
<code>\@glsxtr@deprecated@abbrstyle</code>		<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>	
.....	136, 137, 140, 141, 145–147	152, 153
<code>\@glsxtr@disabledflycommand</code>	18	<code>\@glsxtr@org@Glsxtrtitlefull</code> ..	152, 153
<code>\@glsxtr@do@wrindex</code>	38	<code>\@glsxtr@org@Glsxtrtitlefullpl</code> ..	152, 153
<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	37	<code>\@glsxtr@org@Glsxtrtitlelong</code> ..	152, 153
<code>\@glsxtr@do@style</code>	10, 168	<code>\@glsxtr@org@Glsxtrtitlelongpl</code> ..	152, 153
<code>\@glsxtr@do@titlecaps@warn</code> ...	85–88, 96	<code>\@glsxtr@org@Glsxtrtitleplural</code> ..	152, 153
<code>\@glsxtr@do@abbreviationsdef</code>	7	<code>\@glsxtr@org@Glsxtrtitleshort</code> ..	152, 153
<code>\@glsxtr@do@accsupp</code>	9, 10	<code>\@glsxtr@org@Glsxtrtitleshortpl</code> ..	152, 153
<code>\@glsxtr@do@octag</code>	21	<code>\@glsxtr@org@Glsxtrtitletext</code> ..	152, 153
<code>\@glsxtr@do@stylewarn</code>	118, 119	<code>\@glsxtr@org@MakeUppercase</code>	152, 153
<code>\@glsxtr@enabletagging</code>	95	<code>\@glsxtr@org@checkfirsthyper</code>	36, 59
<code>\@glsxtr@end@</code>	15	<code>\@glsxtr@org@delimN</code>	21, 22
<code>\@glsxtr@endescspch</code>	92–95	<code>\@glsxtr@org@delimR</code>	21, 22

short	130
\abbreviationsname	6
\abbrvpluralsuffix	103, 104, 122, 124, 126, 128–130, 132, 134–141, 149, 150
\ABP	7
\Abp	7
\abp	7
\ACRfullfmt	58
\Acrfullfmt	58
\acrfullfmt	58
\ACRfullplfmt	58
\Acrfullplfmt	58
\acrfullplfmt	58
\acronymentry	57
\acronymfont	31–34, 43, 58, 59
\acronymname	6
\acronymsort	57
\acronymtype	6, 57, 58
\acrpluralsuffix	57
\actualchar	94
\addtolength	185
\advance	47, 56
\AF	7
\Af	7
\af	7
\AFP	7
\Afp	7
\afp	7
\AL	7
\Al	7
\al	7
\ALP	7
\Alp	7
\alp	7
\AnyTrackedLanguages	168
\appto	10, 13, 35, 38, 45, 53, 91, 100, 103
\AS	7
\As	7
\as	7
\ASP	7
\Asp	7
\asp	7
\AtBeginDocument	11, 19, 20
\AtEndDocument	13, 46, 55, 64

B

babel package	91, 93, 100
\begin	49, 66, 170–174, 184

C

category attributes:	
discardperiod	99
entrycount	44, 45, 47, 56
firsttuc	88
glossdesc	85
glossdescfont	85
glossname	86
glossnamefont	86, 89
headuc	153
indexname	91
indexonlyfirst	38
insertdots	103
nohyper	37
nohyperfirst	125
regular	22, 50, 121– 123, 125, 127, 130–132, 142, 144, 148, 150
\cGLS	7, 45, 56
\cGls	7, 45, 56
\cgls	7, 45, 56
\cGLSformat	49
\cGlsformat	48
\cglsformat	48, 50
\cGLSpl	7, 45, 56
\cGlspl	7, 45, 56
\cglspl	7, 45, 56
\cGLSplformat	49
\cGlsplformat	48
\cglsplformat	48, 50
\columnwidth	19, 20
\count@	47, 55, 56
\cs	49, 184
\csdef	35, 36, 46, 51, 52, 54, 79, 119, 120, 127, 170
\csedef	52
\csgdef	21, 46, 52, 54, 55
\csletcs	120
\csname	19, 22, 31–36, 52, 64, 67, 68, 84, 102, 107–116, 121, 184
\csuse	21, 35, 36, 51–55, 79, 90, 98, 119, 120, 177–180
\csxdef	13, 52, 55
\CurrentOption	10, 169
\CurrentTrackedTag	168
\CustomAbbreviationFields	104, 121, 123–125, 127, 129, 130, 132, 133, 142, 143, 145, 148, 149

D

\DeclareAcronymList	57
---------------------------	----

`\DeclareOption` 4, 169
`\DeclareOptionX` 4, 10
`\def` 11,
14–18, 20, 22–34, 39–43, 48–50, 57, 60,
62, 90–97, 100–104, 107–116, 118, 184, 185
`\define@boolkey` 5, 37
`\define@choicekey` 5, 8, 9
`\define@key` 9, 10, 35, 102
`\DefineAcronymSynonyms` 8
`\delimN` 21, 22
`\delimR` 21, 22
`\detokenize` 15
`\dimen@` 59, 177–184
`\dimen@i` 178–180
`\dimen@ii` 178–180
`\dimexpr` 19, 20, 176
`\disable@keys` 6, 11, 14
`\do` 10, 14, 45, 56, 60, 62, 84, 96
`\do@gl@link@checkfirsthyper`
..... 23, 30–34, 107–116
`\do@gl@disablehyperinlist` 38
doc package 94
`\DTLifinlist` 60, 61

E

`\eappto` 10, 91, 169
`\edef` 37, 51, 52, 54, 60,
61, 64, 85, 86, 88–90, 92, 94, 102, 178–180
`\eglssetwidest` 177–184
`\else` 5, 6, 9, 15, 20, 22, 38, 47, 59, 61, 65–67,
91, 92, 94, 101–103, 110–116, 122, 124,
126, 128–133, 149, 150, 171–175, 185, 186
`\emph` 141
`\encapchar` 94
`\end` 66, 170–174, 184
`\endcsname` 19, 22, 31–36,
52, 64, 67, 68, 84, 102, 107–116, 121, 184
entry categories:
 abbreviation 116
 general 79, 81
 index 83
`\epreto` 91
`\equal` 67
etoolbox package 4
`\expandafter` ... 10, 14–17, 35, 36, 39, 50,
51, 60, 61, 84, 87, 88, 91, 94, 101, 103, 104
`\expandonce` 57, 92

F

`\fi` 5, 6, 9, 13, 15, 19,
20, 22, 38, 47, 55, 56, 59, 61, 62, 64–68,
91, 92, 95, 101, 103, 110–116, 122, 124,
126, 128–133, 149, 150, 171–175, 177–186
first use 187
 flag 187
 text 187
`\firstacronymfont` 58, 59
`\footnote` 125–127
`\forallallglossaries` 13, 82, 84, 177–183
`\forallallglsentries` 47, 55
`\ForEachTrackedDialect` 168
`\forglentries` 13, 82, 84, 177–183
`\forlistcsloop` 56
`\forlistloop` 62, 63, 97
`\futurelet` 100

G

`\gdef` 21, 93, 94
`\Genacrfullformat` 58
`\genacrfullformat` 58
`\GenericAcronymFields` 57
`\Genplacrfullformat` 58
`\genplacrfullformat` 58
`\glo@name` 87, 88
glossaries package
..... 4, 5, 7, 8, 10, 12, 19, 23, 36, 37,
40, 44, 49, 57, 59, 88, 97, 101, 147, 151, 169
glossaries-accsupp package 9, 10, 68
glossaries-extra package 2, 68
glossaries-extra-stylemods package ... 9, 97, 168
`\GlossariesExtraWarning`
..... 5, 6, 16, 18, 58, 66, 85–90, 96, 120
`\GlossariesExtraWarningNoLine` . 6, 47, 56
`\GlossariesWarning` 21, 62, 63, 118
`\GlossariesWarningNoLine` 60, 64
glossary styles:
 alttree 175, 176, 184
 inline 175
 listdotted 170
 listdottedstyle 170
 sublistdotted 170
glossary-long package 171
glossary-longbooktabs package 171
glossary-mcols package 169
glossary-tree package 169
`\glossaryentrynumbers` 22
`\glossaryheader` 170–174, 185

<code>\glossarysection</code>	67	<code>\glsaccessplural</code>	25
<code>\glossarytitle</code>	67	<code>\Glsaccessshort</code>	
<code>\glossarytoctitle</code>	67 31, 110, 117, 124, 126, 128, 131, 150	
<code>\glossentry</code>	170–174, 185	<code>\glsaccessshort</code>	31, 105, 110,
<code>\glossentrydesc</code>	170–176	111, 117, 122, 124, 126, 128–132, 149, 150	
<code>\glossentryname</code>	170–174, 185	<code>\Glsaccessshortpl</code>	
<code>\glossentrysymbol</code>	171–176 32, 114, 116, 124, 126, 128, 131, 150	
<code>\GLS</code>	45, 56	<code>\glsaccessshortpl</code>	32, 105, 113, 114,
<code>\Gls</code>	17, 45, 56	116, 117, 122, 124, 126, 128–132, 149, 150	
<code>\gls</code>	16, 18, 45, 56, 65	<code>\GLSaccesssymbol</code>	28
<code>\gls@assign@field</code>	35	<code>\Glsaccesssymbol</code>	28, 95
<code>\gls@checkseeallowed</code>	15, 60	<code>\glsaccesssymbol</code>	27, 95, 99
<code>\gls@codepage</code>	64	<code>\GLSaccesssymbolplural</code>	28
<code>\gls@defdocnewglossaryentry</code>	45, 53	<code>\Glsaccesssymbolplural</code>	28
<code>\gls@defglossaryentry</code>	16, 17	<code>\glsaccesssymbolplural</code>	28
<code>\gls@save@numberlist</code>	20, 22	<code>\GLSaccessstext</code>	24
<code>\gls@tmplen</code>	177–183, 185	<code>\Glsaccessstext</code>	24
<code>\glsabbrvdefaultfont</code>		<code>\glsaccessstext</code>	24
..... 106, 122, 124, 126, 128–130, 132		<code>\glsacrshortcutstrue</code>	8
<code>\glsabbrvmfont</code>	141–147	<code>\glsacspacemax</code>	59
<code>\glsabbrvfont</code>	41, 42, 58, 106, 110,	<code>\glsadd</code>	14, 65
111, 113, 114, 116, 117, 121–130, 132–150		<code>\glsaddstoragekey</code>	79
<code>\glsabbrvuserfont</code>	148–150	<code>\glsbackslash</code>	15
<code>\glsabrvfont</code>		<code>\glsapscase</code>	23–34, 36, 107–118
..... 122, 123, 125, 127, 142, 143, 148, 150		<code>\glsacategory</code>	23, 24, 37, 41, 42, 80–
<code>\GLSaccessdesc</code>	27	82, 85, 86, 88–90, 95, 98, 107–111, 113, 114	
<code>\Glsaccessdesc</code>	27, 85, 95	<code>\glsacategorylabel</code>	37, 102–104, 127
<code>\glsaccessdesc</code>	26, 85, 99	<code>\glsaclosebrace</code>	66, 67
<code>\GLSaccessdescplural</code>	27	<code>\glsacurrententrylabel</code> ...	20–22, 90, 97, 98
<code>\Glsaccessdescplural</code>	27	<code>\glsacurrentfieldvalue</code>	147
<code>\glsaccessdescplural</code>	27	<code>\glsacustomtext</code>	23, 31–34, 107–116, 118
<code>\GLSaccessfirst</code>	25	<code>\glsadefaulttype</code>	6, 65
<code>\Glsaccessfirst</code>	24	<code>\glsadescriptionaccessdisplay</code> ..	72, 73, 85
<code>\glsaccessfirst</code>	24	<code>\glsadescriptionpluralaccessdisplay</code> .	73
<code>\GLSaccessfirstplural</code>	26	<code>\glsadescwidth</code>	170, 172–174
<code>\Glsaccessfirstplural</code>	26	<code>\glsadetoklabel</code>	12,
<code>\glsaccessfirstplural</code>	25	14, 15, 46, 51–56, 62, 63, 84, 87, 88, 178–180	
<code>\Glsaccesslong</code> 33, 105, 112, 122, 129, 132, 149		<code>\glsadisplaynumberlist</code>	60, 62
<code>\glsaccesslong</code>	33, 105, 111, 112,	<code>\glsadohyperlink</code>	40
122, 124, 126, 128, 129, 131–133, 149, 150		<code>\glsadoifexists</code> ...	23, 30–34, 62, 63, 107–116
<code>\Glsaccesslongpl</code>		<code>\glsadoifexistsorwarn</code>	85, 86, 88, 89, 95
..... 34, 105, 115, 122, 129, 132, 149		<code>\glsadonohyperlink</code>	40
<code>\glsaccesslongpl</code>	34, 105, 115, 116,	<code>\glsadosanitizesort</code>	61
122, 124, 126, 128, 129, 131–133, 149, 150		<code>\glsenableentrycount</code>	45, 47, 55
<code>\GLSaccessname</code>	26	<code>\glsenableentryunitcount</code>	46, 56
<code>\Glsaccessname</code>	26	<code>\glsentrycurrcount</code>	46, 47, 53
<code>\glsaccessname</code>	26	<code>\Glsentrydesc</code>	73, 77, 86
<code>\GLSaccessplural</code>	25	<code>\glsentrydesc</code>	72, 73, 77, 86
<code>\Glsaccessplural</code>	25	<code>\Glsentrydescplural</code>	73, 78

<code>\glsentrydescplural</code>	73, 77, 78	<code>\glsfirstabbrvdefaultfont</code>	
<code>\Glsentryfirst</code>	50, 70, 76	106, 122, 124, 126, 128–130, 132
<code>\glsentryfirst</code>	50, 70, 76, 164	<code>\glsfirstabbrvemfont</code>	142–147
<code>\Glsentryfirstplural</code>	51, 71, 77	<code>\glsfirstabbrvfont</code> 58, 105, 121–132, 134–150	
<code>\glsentryfirstplural</code> ...	50, 71, 76, 77, 165	<code>\glsfirstabbrvuserfont</code>	149, 150
<code>\Glsentryfull</code>	58	<code>\glsfirstaccessdisplay</code>	70
<code>\glsentryfull</code>	58	<code>\glsfirstlongdefaultfont</code>	
<code>\Glsentryfullpl</code>	58	122, 124, 129, 131, 132
<code>\glsentryfullpl</code>	58	<code>\glsfirstlongemfont</code>	142–144, 146
<code>\glsentryitem</code>	170–174, 185	<code>\glsfirstlongfont</code>	
<code>\Glsentrylong</code>	42, 43, 50, 75, 78	105, 121–129, 131–133, 142–146, 148–150
<code>\glsentrylong</code> 42, 43, 50, 75, 78, 127, 165, 166		<code>\glsfirstlongfootnotefont</code>	126–128
<code>\Glsentrylongpl</code>	43, 51, 75, 78	<code>\glsfirstlonguserfont</code>	149, 150
<code>\glsentrylongpl</code>	43, 50, 75, 78, 79, 166	<code>\GLSfirstplural</code>	158
<code>\Glsentryname</code>	68, 76, 87–90	<code>\Glsfirstplural</code>	158
<code>\glsentryname</code>	68, 69, 76, 91, 177–184	<code>\glsfirstplural</code>	158
<code>\glsentrynumberlist</code>	61, 63, 182–184	<code>\glsfirstpluralaccessdisplay</code>	71
<code>\Glsentryplural</code>	70, 76	<code>\glsforeachincategory</code>	118
<code>\glsentryplural</code>	69, 70, 76, 164	<code>\glsenentryfmt</code>	23
<code>\glsentryprevcount</code>	46, 47, 53	<code>\glsgetattribute</code> ...	47, 51–53, 85, 86, 88–91
<code>\glsentryprevmaxcount</code>	54	<code>\glsgetcategoryattribute</code>	80
<code>\glsentryprevtotalcount</code>	53	<code>\glsgetwidestname</code>	176
<code>\Glsentryshort</code>	42, 43, 74, 78	<code>\glsgroupheading</code>	170–174, 185
<code>\glsentryshort</code>	41–43, 59, 73, 74, 78, 162	<code>\glsgroupskip</code>	171–175, 186
<code>\Glsentryshortpl</code>	42, 43, 74, 78	<code>\glsattribute</code> 47, 52, 54, 56, 85, 86, 88,	
<code>\glsentryshortpl</code>	42, 43, 74, 78, 162, 163	89, 91, 122, 123, 125, 127, 142, 144, 148, 150	
<code>\Glsentrysymbol</code>	71, 77	<code>\glsattribute</code>	80
<code>\glsentrysymbol</code>	71, 72, 77, 181, 182	<code>\glsnumber</code>	91
<code>\Glsentrysymbolplural</code>	72, 77	<code>\glsifattribute</code>	
<code>\glsentrysymbolplural</code>	72, 77	...	37, 38, 82, 85–88, 97, 99, 100, 153–161
<code>\Glsentrytext</code>	69, 76	<code>\glsifcategory</code>	82
<code>\glsentrytext</code>	69, 76, 163	<code>\glsifcategoryattribute</code> ..	37, 81, 103, 104
<code>\Glsentryuseri</code>	29	<code>\glsifnotregular</code>	24
<code>\glsentryuseri</code>	29	<code>\glsifnotregularcategory</code>	82
<code>\Glsentryuserii</code>	29	<code>\glsifplural</code> 23, 25–28, 30–34, 99, 100, 107–117	
<code>\glsentryuserii</code>	29	<code>\glsifregular</code>	23, 24, 50, 51
<code>\Glsentryuseriii</code>	29	<code>\glsifregularcategory</code>	81
<code>\glsentryuseriii</code>	29	<code>\glsignore</code>	21, 22
<code>\Glsentryuseriv</code>	29	<code>\glsinlinedescformat</code>	175
<code>\glsentryuseriv</code>	30	<code>\glsinlinesubdescformat</code>	175
<code>\Glsentryuserv</code>	30	<code>\glsinsert</code>	23, 31–34, 107–118
<code>\glsentryuserv</code>	30	<code>\glskeylisttok</code>	57, 103, 104
<code>\Glsentryuservi</code>	30	<code>\glslabel</code>	23, 37, 98, 99, 116–118, 127
<code>\glsentryuservi</code>	30	<code>\glslabeltok</code>	57, 103, 104, 121–125,
<code>\glsfieldxdef</code>	84	127, 129, 130, 132, 133, 142–145, 148–150	
<code>\glsfindwidesttoplevelname</code>	177	<code>\glsletentryfield</code>	91
<code>\GLSfirst</code>	157	<code>\glslink</code>	58
<code>\Glsfirst</code>	157	<code>\glslink options</code>	
<code>\glsfirst</code>	157	format	90

noindex	37	\glissetattribute	122, 123, 125, 127, 129, 130, 132, 133, 142, 144, 145, 148, 150
\glslinkcheckfirsthyperhook	37	\glissetcategoryattribute	45, 57, 59, 80, 81, 83, 84, 96
\glslinkvar	39	\glsshortaccessdisplay	73, 74
\glslistdottedwidth	170	\glsshortpltok	104, 122–125, 127, 129, 130, 142, 143, 148, 150
\glslongaccessdisplay	75	\glsshortpluralaccessdisplay	74
\glslongdefaultfont	106, 122, 124, 125, 129, 131, 132	\glsshorttok .	57, 103, 104, 121, 123–125, 127, 129, 130, 133, 142, 143, 145, 148, 149
\glslongemfont	141–146	\glssubentryitem	170–175, 185
\glslongfont	42, 43, 106, 111, 112, 115, 116, 122, 124, 126, 128, 129, 131, 132, 142–144, 146, 149, 150	\glssymbolaccessdisplay	71, 72
\glslongfootnotefont	125, 126, 128	\glssymbolpluralaccessdisplay	72
\glslongpltok	104, 122– 125, 127, 132, 133, 142, 143, 145, 148, 150	\glstarget	170–175, 185
\glslongpluralaccessdisplay	75	\GLStext	155, 156
\glslongtok ..	57, 103, 104, 121–125, 127, 129, 130, 132, 133, 142, 143, 145, 148, 149	\Glstext	156
\glslonguserfont	148–150	\glstext	155
\glsnameaccessdisplay	68, 69, 87, 89	\glstextaccessdisplay	69
\glsnamefont	87–90	\glstextup	134
\glsnoidxdisplayloc	63	\glstreeindent	184, 185
\glsnoidxdisplayloclisthandler	62	\glstreenamebox	185
\glsnoidxloclist	63	\glstreenamefmt	176–185
\glsnoidxnumberlistloophandler	63	\glstype	31–34, 107–116
\glsnonumberlistfalse	20	\glset	14, 48, 49
\glsnonumberlisttrue	20	\glswrite	60
\glsnumberlistloop	61	\Glsxtr	18
\glsnumlistlastsep	62	\glxtr	18
\glsnumlistsep	62	\glxtr@addunused	14
\glsopenbrace	66, 67	\glxtr@applyabbrvfmt	116
\glsorder	60	\glxtr@applyabbrvstyle	102, 103, 119
\glspagelistwidth	170, 172–174	\glxtr@doption	4, 6, 10, 11
\GLSpl	45, 56	\glxtr@ifnextpunc	100
\Glspl	17, 45, 56	\glxtr@ifpunctoken	100
\glspl	17, 45, 56	\glxtr@keylist	16, 17
\GLSplural	156	\glxtr@next	101
\Glsplural	156, 157	\glxtr@orgmakenoidxglossaries	14
\glsplural	156	\glxtr@punclist	100, 101
\glspluralaccessdisplay	69, 70	\glxtr@warnonexistsordo	5, 12, 13
\glspluralsuffix	103, 106, 122, 124, 126, 128–130, 132, 134, 138, 148	\glxtr@abbrvtype	6, 104
\glspostdescription	97, 170–176	\glxtr@addallcrossrefs	13
\glspostinline	175	\glxtrAltTreeIndent	176
\glspostlinkhook	23, 31–34, 107–116	\glxtrAltTreeInit	184
\glsprestandardsort	61	\glxtrAltTreePar	175
\glseeformat	63	\glxtrAltTreeSetHangIndent ...	176, 185
\glsetabbrvfmt	23, 24, 41, 42, 85, 86, 88, 89, 95, 107–111, 113, 114	\glxtrAltTreeSetSubHangIndent	185
		\glxtraltrtreeSubSymbolDescLocation	185
		\glxtraltrtreeSymbolDescLocation ..	176, 185
		\glxtrassignfieldfont	24–30

<code>\glxstrcat</code>	16, 17	<code>\Glsxtrheadshortpl</code>	152
<code>\glxstrComputeTreeIndent</code>	185	<code>\glxstrheadshortpl</code>	152
<code>\glxstrComputeTreeSubIndent</code>	185	<code>\Glsxtrheadtext</code>	152
<code>\GlsXtrDefineAbbreviationShortcuts</code>	8, 9	<code>\glxstrheadtext</code>	152
<code>\GlsXtrDefineOtherShortcuts</code>	8, 9	<code>\glxstrtrifcounttrigger</code>	48, 49
<code>\glxstrdiscardperiod</code>	98	<code>\glxstrtrifemptyglossary</code>	67
<code>\glxstrdoautoindexname</code>	38, 39, 90	<code>\glxstrtrifindexing</code>	38
<code>\glxstrdopostpunc</code>	127	<code>\glxstrtrifnextpunc</code>	100, 101
<code>\glxstrdowrglossaryhook</code>	38	<code>\glxstrtrifperiod</code>	99, 100
<code>\GlsXtrEnableEntryCounting</code>	56	<code>\glxstrtrifwasfirstuse</code>	
<code>\GlsXtrEnableEntryUnitCounting</code>	45	.. 23–26, 30–34, 37, 99, 107, 110–116, 127	
<code>\GlsXtrEnableOnTheFly</code>	16, 18	<code>\Glsxtrinlinefullformat</code>	106,
<code>\glxstrfieldtitlecase</code>	85–88	107, 119, 120, 126, 128, 129, 131, 132, 167	
<code>\glxstrfirstscfont</code>	134–137	<code>\glxtrinlinefullformat</code>	
<code>\glxstrfirstsmfont</code>	138–141 106–108, 119, 120, 126, 128–132, 166	
<code>\GlsXtrFormatLocationList</code>	20, 22, 182–184	<code>\Glsxtrinlinefullplformat</code>	106,
<code>\GLSxtrfull</code>	7, 160, 161	109, 119, 120, 126, 128, 129, 131, 132, 167	
<code>\Glsxtrfull</code>	7, 161	<code>\glxtrinlinefullplformat</code>	105, 106, 108,
<code>\glxstrfull</code>	7, 160	109, 119, 120, 126, 128, 129, 131, 132, 167	
<code>\Glsxtrfullformat</code>	105, 118–120,	<code>\glxtrinsertinsidefalse</code>	121
122, 124, 126, 128, 130, 131, 133, 149, 150		<code>\GLSxtrlong</code>	7, 158, 159
<code>\glxstrfullformat</code>	105, 118–	<code>\Glsxtrlong</code>	7, 159
120, 122–124, 126, 128, 130–132, 149, 150		<code>\glxstrlong</code>	7, 158, 159
<code>\GLSxtrfullpl</code>	7, 161	<code>\GLSxtrlongpl</code>	7, 159, 160
<code>\Glsxtrfullpl</code>	7, 161, 162	<code>\Glsxtrlongpl</code>	7, 160
<code>\glxstrfullpl</code>	7, 161	<code>\glxstrlongpl</code>	7, 159
<code>\Glsxtrfullplformat</code>	105, 117, 119, 120,	<code>\glxstrlongshortdescsort</code>	123
122, 124, 126, 128, 130, 131, 133, 149, 150		<code>\glxstrmarkhook</code>	151
<code>\glxstrfullplformat</code>	117, 119, 120,	<code>\glxstrnewabbrevpresetkeyhook</code>	104
122, 124, 126, 128, 130, 131, 133, 149, 150		<code>\glxstrnewnumber</code>	7
<code>\glxstrfullsep</code>	105, 121–	<code>\glxstrnewsymbol</code>	7
124, 126, 128, 129, 131, 132, 142, 143, 147		<code>\glxstrNoGlossaryWarning</code>	9, 64
<code>\glxstrgenabbrvfmt</code>	23	<code>\GlsXtrNoGlsWarningAutoMake</code>	67
<code>\Glsxtrheadfirst</code>	152	<code>\GlsXtrNoGlsWarningBuildInfo</code>	68
<code>\glxtrheadfirst</code>	152	<code>\GlsXtrNoGlsWarningCheckFile</code>	67
<code>\Glsxtrheadfirstplural</code>	152	<code>\GlsXtrNoGlsWarningEmptyMain</code>	67
<code>\glxtrheadfirstplural</code>	152	<code>\GlsXtrNoGlsWarningEmptyNotMain</code>	67
<code>\Glsxtrheadfull</code>	153	<code>\GlsXtrNoGlsWarningEmptyStart</code>	67
<code>\glxtrheadfull</code>	153	<code>\GlsXtrNoGlsWarningHead</code>	67
<code>\Glsxtrheadfullpl</code>	153	<code>\GlsXtrNoGlsWarningMisMatch</code>	68
<code>\glxtrheadfullpl</code>	153	<code>\GlsXtrNoGlsWarningNoOut</code>	68
<code>\Glsxtrheadlong</code>	153	<code>\GlsXtrNoGlsWarningTail</code>	68
<code>\glxtrheadlong</code>	152	<code>\Glsxtrpl</code>	18
<code>\Glsxtrheadlongpl</code>	153	<code>\glxstrpl</code>	18
<code>\glxtrheadlongpl</code>	153	<code>\glxstrpostdescription</code>	83, 97, 175
<code>\Glsxtrheadplural</code>	152	<code>\glxstrpostlink</code>	98
<code>\glxtrheadplural</code>	152	<code>\glxstrpostlinkendsentence</code>	98
<code>\Glsxtrheadshort</code>	152	<code>\glxstrpostlinkhook</code>	98
<code>\glxtrheadshort</code>	152	<code>\glxstrpostlocalreset</code>	44, 46, 54

<code>\ifglsonlyfirst</code>	38	<code>\makeatletter</code>	64, 93
<code>\ifglsnogroupskip</code>	171–175, 186	<code>\makeatother</code>	93
<code>\ifglsonumberlist</code>	22	<code>\makebox</code>	170, 185
<code>\ifglssanitizesort</code>	61	<code>\makefirsttuc</code>	97
<code>\ifglssused</code>	14, 37, 38, 47, 56, 116, 177, 178, 180, 182, 183	<code>\makeglossaries</code>	59, 64, 66–68
<code>\ifglsxindy</code>	64–66	<code>\makeglossary</code>	60
<code>\ifglsxtrinsertinside</code>	110–116, 122, 124, 126, 128–133, 149, 150	<code>makeindex</code>	187
<code>\ifHy@hyperindex</code>	90	<code>makeindex</code>	59
<code>\ifKV@glslink@noindex</code>	38	<code>\makenoidxglossaries</code>	66
<code>\ifnum</code>	47, 55, 56, 185	<code>\MakeTextUppercase</code>	152
<code>\ifthenelse</code>	67	<code>\MakeUppercase</code>	152, 153
<code>\IfTrackedLanguageFileExists</code>	168	<code>\marg</code>	184
<code>\ifundef</code>	40, 60, 96	<code>\markboth</code>	152
<code>\ifx</code>	19, 20, 92, 94, 101, 102, 186	<code>\markright</code>	152
<code>\immediate</code>	47, 55, 64	<code>\maxdimen</code>	19, 20
<code>\index</code>	91	<code>\mbox</code>	185
<code>\indexspace</code>	186	<code>\medskip</code>	67
<code>\input</code>	168	<code>\MessageBreak</code>	15, 18, 47, 56, 61, 118, 119
<code>\istfilename</code>	60	<code>mfirsttuc package</code>	96
<code>\item</code>	66, 170	<code>\mfirsttucMakeUppercase</code>	24–34, 36, 42, 43, 50, 58, 69– 79, 87, 88, 108, 109, 111, 112, 114, 116–118
J			
<code>\jobname</code>	64–68	<code>\mfu@checkword@arg</code>	96, 97
		<code>\mfu@checkword@do</code>	97
K			
<code>\key@ifundefined</code>	35	N	
<code>\KV@glslink@hyperfalse</code>	37, 40	<code>\NeedsTeXFormat</code>	4, 169
<code>\KV@glslink@noindexfalse</code>	37, 38	<code>\new@glossaryentry</code>	15, 61
<code>\KV@glslink@noindextrue</code>	40	<code>\new@ifnextchar</code> .	35, 36, 49, 50, 100, 106–115
L			
<code>\LaTeX</code>	65, 66	<code>\newabbr</code>	7
<code>\leaders</code>	170	<code>\newabbreviation</code>	7, 58
<code>\let</code>	4, 6–8, 10, 11, 14, 15, 18, 19, 21–34, 36, 37, 39– 41, 45, 46, 54–60, 62, 63, 85–92, 96, 97, 100–103, 107–116, 127, 151–153, 175, 177	<code>\newabbreviationhook</code>	104
<code>\letabbreviationstyle</code>	127, 128, 130, 131, 133, 135, 136, 139, 144, 145	<code>\newabbreviationstyle</code>	121, 123–125, 127, 129, 130, 132–150
<code>\letcs</code>	14, 35, 62, 63, 85–90	<code>\newacronym</code>	57, 58
<code>\levelchar</code>	94	<code>\newacronymhook</code>	57
<code>\listadd</code>	51	<code>\newacronymstyle</code>	58, 59
<code>\listbreak</code>	96	<code>\newcommand</code>	4–18, 20–23, 35–40, 44–47, 49–56, 58, 59, 62, 63, 65–84, 90– 116, 118–121, 123, 125, 133, 134, 137, 138, 141, 147, 148, 152–169, 175–177, 184
<code>\listcseadd</code>	52	<code>\newentry</code>	7
<code>\listcsxadd</code>	52	<code>\newglossary</code>	6, 60
<code>\loadglsentries</code>	15, 65	<code>\newglossaryentry</code> 7, 15, 45, 53, 57, 83, 84, 104	
M			
<code>\MakeAcronymsAbbreviations</code>	59	<code>\newglossaryentry options</code>	
		<code>desc</code>	72, 73, 77
		<code>descplural</code>	73, 77, 78
		<code>first</code>	39, 70, 76, 121, 157, 158, 164, 187
		<code>firstplural</code> 70, 71, 76, 77, 121, 157, 158, 165, 187	

hyper	151		
long	75, 78, 165		
longplural	75, 79, 166		
name	68, 69, 76, 91		
noindex	151		
plural	69, 70, 76, 121, 156, 157, 163		
see	6, 15, 60		
short	74, 78, 102		
shortplural	74, 78, 102		
symbol	71, 77		
symbolplural	72, 77		
text	39, 69, 76, 121, 155, 156, 163		
\newif	90, 121		
\newlength	176		
\newnum	7		
\newrobustcmd	35, 36, 49, 50, 96, 97, 106–115, 154–162, 177–183		
\newsym	7		
\newterm	83		
\newtoks	102		
\newwrite	60		
\NoCaseChange	153–161		
\noexpand	10, 57, 64, 92, 104, 169		
\nofiles	67		
\noindent	67		
\nopostdesc	16, 17, 83		
\nr	5, 8, 9		
\ns@GLSxtrfull	108		
\ns@Glsxtrfull	107		
\ns@glxtrfull	106		
\ns@GLSxtrfullpl	109		
\ns@Glsxtrfullpl	108		
\ns@glxtrfullpl	108		
\ns@GLSxtrlong	112		
\ns@Glsxtrlong	112		
\ns@glxtrlong	111		
\ns@GLSxtrlongpl	115		
\ns@Glsxtrlongpl	115		
\ns@glxtrlongpl	114		
\ns@GLSxtrshort	111		
\ns@Glsxtrshort	110		
\ns@glxtrshort	109, 110		
\ns@GLSxtrshortpl	114		
\ns@Glsxtrshortpl	113		
\ns@glxtrshortpl	113		
\null	9		
\number	52–55		
\numexpr	52, 55		
		O	
		\or	5, 8
		\org@glossaryentrynumbers	20
		P	
		\p@gl@s@hyp@opt	39
		package options:	
		abbreviations	6, 7
		accsupp	9, 68
		acronym	6
		automake	62, 65
		docdef	14, 15, 45, 53
		nonumberlist	20
		numbers	7
		shortcuts	8
		all	8
		false	8
		none	8
		true	8
		style	10
		stylemods	10
		symbols	7, 83
		undefaction	12
		warn	5
		\PackageError	5, 10, 15, 18, 35, 36, 45, 46, 53, 55, 56, 58, 60, 61, 118–121, 169
		\PackageWarning	6
		\PackageWarningNoLine	6
		\par	67, 68, 175, 176, 185
		\parindent	176, 185
		\PassOptionsToPackage	4
		\preto	38
		\printabbreviations	6
		\printglossaries	60, 66
		\printglossary	6, 60, 66
		\printglossary options	
		nonumberlist	22
		\printnoidxglossaries	66
		\printnoidxglossary	66
		\printnumbers	7, 83, 84
		\printsymbols	7, 83
		\ProcessOptions	169
		\ProcessOptionsX	10
		\protect	76–79, 105, 121–127, 129, 130, 132–143, 145, 148–150, 153–161
		\protected@csedef	176
		\protected@csxdef	176
		\protected@edef	19, 57, 91, 104
		\protected@write	22, 60

<code>\providecommand</code>	6, 22, 36, 47, 55, 60, 64, 169	<code>\strut</code>	170–175
<code>\ProvidesFile</code>	168	<code>\subglossentry</code>	170–175, 185
<code>\ProvidesPackage</code>	4, 169		
Q		T	
<code>\quotechar</code>	94	<code>\tablehead</code>	173, 174
R		<code>\tabletail</code>	173, 174
<code>\raggedright</code>	172, 174	<code>\tabularnewline</code>	170–175
<code>\relax</code>	5, 7–11, 15, 18–21, 39, 46, 47, 55, 60, 63, 92, 94, 96, 98, 99, 102, 103, 177–186	<code>\TeX</code>	65
<code>relsize package</code>	137	<code>\texorpdfstring</code>	162–167
<code>\renewcommand</code>	5, 6, 8–15, 18, 20–23, 35–38, 40, 44–47, 50, 51, 53–62, 64, 83, 85, 86, 88, 89, 95–98, 106, 119–151, 170–175, 185, 186	<code>textcase package</code>	151
<code>\renewenvironment</code>	170–174, 184	<code>\textsc</code>	133
<code>\renewglossarystyle</code>	170–174, 184	<code>\textsmaller</code>	137
<code>\RequireGlossariesExtraLang</code>	168	<code>\texttt</code>	65–67
<code>\RequirePackage</code>	4, 9, 10, 169	<code>\the</code>	57, 94, 104, 121–125, 127, 129, 130, 132, 133, 142–145, 148–150
<code>\reserved@a</code>	100	<code>\theindex</code>	91
<code>\reserved@b</code>	100	<code>\this@dialect</code>	168
<code>\reserved@d</code>	101	<code>\toks@</code>	94
<code>\RestoreAcronyms</code>	58, 59	U	
<code>\romannumeral</code>	176, 177, 184	<code>\undef</code>	95
S		<code>\underline</code>	97
<code>\s@gl@s@hyp@opt</code>	39	<code>\unskip</code>	14, 170
<code>\s@gl@xtr@enabletagging</code>	95	<code>\usepackage</code>	66, 67
<code>\setabbreviationstyle</code>	58, 122, 130	V	
<code>\setacronymstyle</code>	58, 59	<code>\val</code>	5, 8, 9
<code>\SetGenericNewAcronym</code>	59	W	
<code>\setglossarystyle</code>	10, 170, 186	<code>\warn@nomakeglossaries</code>	60
<code>\setkeys</code>	10, 11, 38, 57, 103, 104	<code>\warn@noprintglossary</code>	60
<code>\setlength</code>	19, 20, 176, 185	<code>\write</code>	47, 55, 60, 64
<code>\settowidth</code>	59, 176–184	X	
<code>\setupglossaries</code>	4, 11	<code>\xcapitalisewords</code>	84
<code>\sfcode</code>	98, 99, 175	<code>\xifinlist</code>	51
<code>\space</code>	5, 16, 18, 45–47, 53, 55, 56, 58–61, 64, 67, 99, 105, 123, 175, 176, 184	<code>xindy</code>	187
<code>\spacefactor</code>	98, 99, 103, 175	<code>xindy</code>	59
<code>\string</code>	5, 15, 16, 18, 22, 35, 36, 45–47, 53, 55, 56, 58–61, 64–67, 87–91	<code>xkeyval package</code>	4
		<code>\XKV@checkchoice</code>	22
		<code>\XKV@plfalse</code>	22
		<code>\XKV@resa</code>	22
		<code>\XKV@sttrue</code>	22