

glossaries-extra.sty v1.10: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2016-12-17

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 | 15 |
| 1.3 | Modifications to Commands Provided by glossaries | 15 |
| 1.3.1 | Existence Checks | 15 |
| 1.3.2 | Document Definitions | 18 |
| 1.3.3 | Existing Glossary Style Modifications | 23 |
| 1.3.4 | Entry Formatting, Hyperlinks and Indexing | 27 |
| 1.3.5 | Entry Counting | 56 |
| 1.3.6 | Acronym Modifications | 69 |
| 1.3.7 | Indexing and Displaying Glossaries | 72 |
| 1.4 | Integration with glossaries-accsupp | 83 |
| 1.5 | Categories | 94 |
| 1.6 | Abbreviations | 117 |
| 1.6.1 | Abbreviation Styles Setup | 133 |
| 1.6.2 | Predefined Styles (Default Font) | 136 |
| 1.6.3 | Predefined Styles (Small Capitals) | 149 |
| 1.6.4 | Predefined Styles (Fake Small Capitals) | 153 |
| 1.6.5 | Predefined Styles (Emphasized) | 157 |
| 1.6.6 | Predefined Styles (User Parentheses Hook) | 163 |
| 1.7 | Using Entries in Headings | 166 |
| 1.8 | Multi-Lingual Support | 184 |
| 2 | Style Adjustments (glossaries-extra-stylemods.sty) | 185 |
| 2.1 | Package Initialisation | 185 |
| 2.2 | List-Like Styles | 186 |
| 2.3 | Longtable Styles | 186 |
| 2.4 | Long Ragged Styles | 187 |
| 2.5 | Supertabular Styles | 189 |
| 2.6 | Super Ragged Styles | 190 |
| 2.7 | Inline Style | 191 |
| 2.8 | Tree Styles | 191 |
| | Glossary | 203 |
| | Change History | 204 |
| | Index | 213 |

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2016/12/17 v1.10 (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.

`\warn@undefaction` This is how `\glxtrundefaction` should behave if `undefaction=warn` is set.

```

35 \newcommand*{\@glxtr@warn@undefaction}[2]{%
36   \@glxtrundeftag\GlossariesExtraWarning{#1}%
37 }

```

`\err@undefaction` This is how `\glxtrundefaction` should behave if `undefaction=error` is set.

```

38 \newcommand*{\@glxtr@err@undefaction}[2]{%
39   \@glxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
40 }

```

`\warn@onexistsordo` This is how `\glxtr@warnonexistsordo` should behave if `undefaction=warn` is set.

```

41 \newcommand*{\@glxtr@warn@onexistsordo}[1]{%
42   \GlossariesExtraWarning{glossaries-extra}{%
43     \string#1\space hasn't been defined, so
44     some errors won't be converted to warnings.
45     (This most likely means your version of
46     glossaries.sty is below version 4.19.))%
47 }

48 \define@choicekey{glossaries-extra.sty}{undefaction}[\val\nr]{%
49   {warn,error}%
50   {%
51     \ifcase\nr\relax
52       \let\glxtrundefaction\@glxtr@warn@undefaction
53       \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
54     \or
55       \let\glxtrundefaction\@glxtr@err@undefaction
56       \let\glxtr@warnonexistsordo\@gobble
57     \fi
58   }

```

In the event that someone wants to develop a post-processor that needs to know what entries have been used in the document, v1.08 introduces the record option, which will write information to the aux file whenever an entry needs to be indexed.

`\@glxtr@record` Does nothing by default.

```
59 \newcommand*{\@glxtr@record}[2]{}
```

`@@glxtr@record` This is the actual code that does the recording The first argument is the option list (as passed in the first optional argument to commands like `\gls`). This allows the `noindex` setting to be picked up.

```
60 \newcommand*{\@@glxtr@record}[2]{%
61   \begingroup
62   \def\@glsnumberformat{glsnumberformat}%
63   \ifcsdef{glo@#2@counter}%
64     {%
65       \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
66     }%
67     {%
```

Entry hasn't been defined, so we'll have to assume the page number by default.

```
68     \def\@gls@counter{page}%
69   }%
70   \setkeys{glslink}{#1}%
71   \ifKV@glslink@noindex
72   \else
73     \glswriteentry{#2}%
74     {%
```

Save the entry counter.

```
75     \glxtr@saveentrycounter
```

Temporarily redefine `\@@do@@wrglossary` so we can use `\glxtr@@do@wrglossary`.

```
76     \let\@@do@@wrglossary\@glxtr@dorecord
77     \glxtr@@do@wrglossary{#2}%
78   }%
79   \fi
80 \endgroup
81 }
```

`glxtr@dorecord`

```
82 \newcommand*\@glxtr@dorecord{%
83   \protected@write\@auxout{}\string\glxtr@record
84     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
85     {\@glslocref}}%
86 }
```

`\glxtr@record`

```
87 \newcommand*{\glxtr@record}[5]{}
```

tr@setup@record Initialise.

```
88 \newcommand*{\glxtr@setup@record}{}
```

saveentrycounter Only store the entry counter information if the indexing is on.

```
89 \newcommand*{\glxtr@indexonly@saveentrycounter}{%
90   \ifKV@glslink@noindex
91   \else
92     \glxtr@saveentrycounter
93   \fi
94 }
```

addloclistfield

```
95 \newcommand*{\glxtr@addloclistfield}{%
96   \key@ifundefined{glossentry}{loclist}{%
97     {%
98       \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
99       \appto\@gls@keymap{,{loclist}{loclist}}%
100      \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
101      \appto\@newglossaryentryposthook{%
102        \gls@assign@field{}{\@glo@label}{loclist}{\@glo@loclist}%
103      }%
104    }%
105  }%
106 }
```

Now define the record package option.

```
107 \define@choicekey{glossaries-extra.sty}{record}[\val\nr]%
108 {off,only,alsoindex}%
109 [only]%
110 {%
111   \ifcase\nr\relax
```

Don't record.

```
112   \def\glxtr@setup@record{%
113     \renewcommand*{\@glxtr@record}[2]{}%
114     \let\@do@wrglossary\glxtr@do@wrglossary
115     \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter
116     \let\glxtrundefaction\glxtr@err@undefaction
117     \let\glxtr@warnonexistsordo\@gobble
118   }%
119   \or
```

Only record (don't index).

```
120   \def\glxtr@setup@record{%
121     \let\@glxtr@record\@glxtr@record
122     \let\@do@wrglossary\@gobble
123     \let\@gls@saveentrycounter\relax
124     \let\glxtrundefaction\glxtr@warn@undefaction
125     \let\glxtr@warnonexistsordo\glxtr@warn@onexistsordo
```

```

126     \glsxtr@addloclistfield
127 }%
128 \or

```

Record and index.

```

129 \def\glsxtr@setup@record{%
130   \let\@glsxtr@record\@glsxtr@record
131   \let\@do@wrglossary\glsxtr@do@wrglossary
132   \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
133   \let\glsxtrundefaction\@glsxtr@warn@undefaction
134   \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
135   \glsxtr@addloclistfield
136 }%
137 \fi
138 }

```

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

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

```

139 \newcount\@glsxtr@docdefval

```

Need to provide conditional commands that are backward compatible:

`\ifglsxtrdocdef`

```

140 \newcommand*{\ifglsxtrdocdef}{\ifnum\@glsxtr@docdefval>0 }

```

`\glsxtrdocdeftrue`

```

141 \newcommand*{\@glsxtrdocdeftrue}{\@glsxtr@docdefval=1 }

```

`\glsxtrdocdeffalse`

```

142 \newcommand*{\@glsxtrdocdeffalse}{\@glsxtr@docdefval=0 }

```

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

```

143 \define@choicekey{glossaries-extra.sty}{docdef}[\val\nr]%
144 {false,true,restricted}[true]%
145 {%
146   \@glsxtr@docdefval=\nr\relax
147   \ifnum\@glsxtr@docdefval=2\relax
148     \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexists}%
149   \fi
150 }

```

`\glsxtrdocdefrestricted`

```

151 \newcommand*{\ifglsxtrdocdefrestricted}{\ifnum\@glsxtr@docdefval=2 }

```


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

```
152 \newcommand*{\@glsdoifexistsorwarn}{\glsdoifexistsorwarn}
```

indexcrossrefs Automatically index cross references at the end of the document

```
153 \define@boolkey{glossaries-extra.sty}{@glsxtr}{indexcrossrefs}[true]{%
154   \if@glsxtrindexcrossrefs
155   \else
156   \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
157   \fi
158 }
```

Switch off since this can increase the build time.

```
159 \@glsxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

164 \@glsxtr@declareoption{nowarn}{%
165   \let\GlossariesExtraWarning\@gobble
166   \let\GlossariesExtraWarningNoLine\@gobble
167   \glsxtr@doption{nowarn}%
168 }
```

glsxtrabbrvtype Glossary type for abbreviations.

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

bbreviationsdef Set by abbreviations option.

```
170 \newcommand*{\@glsxtr@abbreviationsdef}{}

bbreviationsdef
171 \newcommand*{\@glsxtr@doabbreviationsdef}{%
172   \@ifpackageloaded{babel}%
173   {\providecommand{\abbreviationsname}{\acronymname}}%
174   {\providecommand{\abbreviationsname}{Abbreviations}}%
175   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
176   \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
177   \newcommand*{\printabbreviations}[1][1]{%
178     \printglossary[type=\glsxtrabbrvtype,##1]%
179   }
```

```

179 }%
180 \disable@keys{glossaries-extra.sty}{abbreviations}%
    If the acronym option hasn't been used, change \acronymtype to \glstrabbrvtype.
181 \ifglssacronym
182 \else
183 \renewcommand*{\acronymtype}{\glstrabbrvtype}%
184 \fi
185 }%

```

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

```

186 \@glstr@declareoption{abbreviations}{%
187 \let\@glstr@abbreviationsdef\@glstr@doabbreviationsdef
188 }

```

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

```

189 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
190 \newcommand*{\ab}{\cglS}%
191 \newcommand*{\abp}{\cglSpl}%
192 \newcommand*{\as}{\glstrshort}%
193 \newcommand*{\asp}{\glstrshortpl}%
194 \newcommand*{\al}{\glstrlong}%
195 \newcommand*{\alp}{\glstrlongpl}%
196 \newcommand*{\af}{\glstrfull}%
197 \newcommand*{\afp}{\glstrfullpl}%
198 \newcommand*{\Ab}{\cGlS}%
199 \newcommand*{\Abp}{\cGlSpl}%
200 \newcommand*{\As}{\GlsXtrshort}%
201 \newcommand*{\Asp}{\GlsXtrshortpl}%
202 \newcommand*{\Al}{\GlsXtrlong}%
203 \newcommand*{\Alp}{\GlsXtrlongpl}%
204 \newcommand*{\Af}{\GlsXtrfull}%
205 \newcommand*{\Afp}{\GlsXtrfullpl}%
206 \newcommand*{\AB}{\cGLS}%
207 \newcommand*{\ABP}{\cGLSpl}%
208 \newcommand*{\AS}{\GLSxtrshort}%
209 \newcommand*{\ASP}{\GLSxtrshortpl}%
210 \newcommand*{\AL}{\GLSxtrlong}%
211 \newcommand*{\ALP}{\GLSxtrlongpl}%
212 \newcommand*{\AF}{\GLSxtrfull}%
213 \newcommand*{\AFP}{\GLSxtrfullpl}%
214 \newcommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

215 \let\GlsXtrDefineAbbreviationShortcuts\relax
216 }

```

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

```

217 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
218   \newcommand*{\newentry}{\newglossaryentry}%
219   \ifdef\printsymbols
220   {%
221     \newcommand*{\newsym}{\glstrnewsymbol}%
222   }{}%
223   \ifdef\printnumbers
224   {%
225     \newcommand*{\newnum}{\glstrnewnumber}%
226   }{}%
227   \let\GlsXtrDefineOtherShortcuts\relax
228 }

```

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.

```

229 \newcommand*{\@glstr@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.

```

230 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
231 {acronyms,acro,abbreviations,abbr,other,all,true,false}[true]{%
232   \ifcase\nr\relax % acronyms
233     \renewcommand*{\@glstr@setupshortcuts}{%
234       \glstrshortcutstrue
235       \DefineAcronymSynonyms
236     }%
237   \or % acro
238     \renewcommand*{\@glstr@setupshortcuts}{%
239       \glstrshortcutstrue
240       \DefineAcronymSynonyms
241     }%
242   \or % abbreviations
243     \renewcommand*{\@glstr@setupshortcuts}{%
244       \GlsXtrDefineAbbreviationShortcuts
245     }%
246   \or % abbr
247     \renewcommand*{\@glstr@setupshortcuts}{%
248       \GlsXtrDefineAbbreviationShortcuts
249     }%
250   \or % other
251     \renewcommand*{\@glstr@setupshortcuts}{%
252       \GlsXtrDefineOtherShortcuts
253     }%
254   \or % all
255     \renewcommand*{\@glstr@setupshortcuts}{%

```

```

256     \glsacrshortcutstrue
257     \DefineAcronymSynonyms
258     \GlsXtrDefineAbbreviationShortcuts
259     \GlsXtrDefineOtherShortcuts
260   }%
261   \or % true
262     \renewcommand*{\@glxtr@setupshortcuts}{%
263       \glsacrshortcutstrue
264       \DefineAcronymSynonyms
265       \GlsXtrDefineAbbreviationShortcuts
266       \GlsXtrDefineOtherShortcuts
267     }%
268   \else % none, false
269     \renewcommand*{\@glxtr@setupshortcuts}{}%
270   \fi
271 }

```

glxtr@doaccsupp

```

272 \newcommand*{\@glxtr@doaccsupp}{}

```

accsupp If accsupp, load glossaries-accsupp package.

```

273 \@glxtr@declareoption{accsupp}{%
274   \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

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

```

275 \newcommand{\glxtrNoGlossaryWarning}[1]{%
276   \@glxtr@defaultnoglossarywarning{#1}%
277 }

```

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

```

278 \define@choicekey{glossaries-extra.sty}{nomissingglstext}[\val\nr]%
279 {true,false}[true]{%
280   \ifcase\nr\relax % true
281     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
282       \null
283     }%
284   \else % false
285     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
286       \@glxtr@defaultnoglossarywarning{#1}%
287     }%
288   \fi
289 }

```

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

glxtr@redefstyles

```

290 \newcommand*{\@glxtr@redefstyles}{}

```

stylemods

```

291 \define@key{glossaries-extra.sty}{stylemods}{%
292   \ifblank{#1}%
293   {%
294     \renewcommand*{\@glsxtr@redefstyles}{%
295       \RequirePackage{glossaries-extra-stylemods}}%
296   }%
297   {%
298     \renewcommand*{\@glsxtr@redefstyles}{%
299       \@for\@glsxtr@tmp:=#1\do{%
300         \IfFileExists{glossary-\@glsxtr@tmp.sty}%
301         {%
302           \eappto\@glsxtr@redefstyles{%
303             \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
304         }%
305         {%
306           \PackageError{glossaries-extra}%
307             {Glossaries style package ‘glossary-\@glsxtr@tmp.sty’
308              doesn’t exist (did you mean to use the ‘style’ key?)}%
309             {The list of values (#1) in the ‘stylemods’ key should
310              match the glossary-xxx.sty files provided with
311              glossaries.sty}%
312         }%
313       }%
314     \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
315   }%
316 }

```

glsxtr@do@style

```

317 \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.

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

321   \setglossarystyle{#1}%
322 }%
323 }

```

Pass all other options to glossaries.

```

324 \DeclareOptionX*{%
325   \expandafter\glsxtr@doooption\expandafter{\CurrentOption}}

```

Process options.

```

326 \ProcessOptionsX

```

Load glossaries if not already loaded.

```
327 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
328 \@glsxtr@doaccsupp
```

Define abbreviations glossaries if required.

```
329 \@glsxtr@abbreviationsdef
330 \let\@glsxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
331 \@glsxtr@setupshortcuts
```

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

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

Now define the user command:

```
333 \newcommand*{\glossariesextrasetup}[1]{%
334   \let\glsxtr@setup@record\relax
335   \let\@glsxtr@setupshortcuts\relax
336   \setkeys{glossaries-extra.sty}{#1}%
337   \@glsxtr@abbreviationsdef
338   \let\@glsxtr@abbreviationsdef\relax
339   \@glsxtr@setupshortcuts
340   \glsxtr@setup@record
341 }
```

@@do@wrglossary Save original definition of @@do@wrglossary.

```
342 \let\glsxtr@@do@wrglossary\@do@wrglossary
```

aveentrycounter Save original definition of \@gls@saveentrycounter.

```
343 \let\glsxtr@saveentrycounter\@gls@saveentrycounter
```

aveentrycounter Change \@gls@saveentrycounter so that it only stores the entry counter information if the indexing is on.

```
344 \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

Set up record option if required.

```
345 \glsxtr@setup@record
```

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

```
346 \AtBeginDocument{%
347   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
348   \def\@glsxtrundeftag{\glsxtrundeftag}%
349 }
```

1.2 Extra Utilities

`\ifemptyglossary`

```
\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.)

```
350 \newcommand{\glstrifemptyglossary}[3]{%
351   \ifglossaryexists{#1}%
352   {%
353     \ifcsstring{glolist@#1}{,}{#2}{#3}%
354   }%
355   {%
356     \glstrundefaction{Glossary type '#1' doesn't exist}{}%
357     #2%
358   }%
359 }
```

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

`\glstoifexists`

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

```
360 \renewcommand{\glstoifexists}[2]{%
361   \ifglstentryexists{#1}{#2}%
362   {%
```

Define `\glslabel` in case it's needed after this command (for example in the post-link hook).

```
363   \edef\glslabel{\glstetoklabel{#1}}%
364   \glstrundefaction{Glossary entry '\glslabel'
365     has not been defined}{You need to define a glossary entry before
366     you can reference it.}%
367   }%
368 }
```

`\glstoifnoexists`

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

```
369 \renewcommand{\glstoifnoexists}[2]{%
370   \ifglstentryexists{#1}{%
```

```

371 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
372 has already been defined}{#2}%
373 }

```

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

```

374 \ifdef\glsoifexistsordo
375 {%
376 \renewcommand{\glsoifexistsordo}[3]{%
377 \ifglstryexists{#1}{#2}%
378 {%
379 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
380 has not been defined}{You need to define a glossary entry
381 before you can use it.}%
382 #3%
383 }%
384 }%
385 }
386 {%
387 \glstr@warnonexistsordo\glsoifexistsordo
388 \newcommand{\glsoifexistsordo}[3]{%
389 \ifglstryexists{#1}{#2}%
390 {%
391 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
392 has not been defined}{You need to define a glossary entry
393 before you can use it.}%
394 #3%
395 }%
396 }%
397 }

```

`\arynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```

398 \ifdef\doifglossarynoexistsordo
399 {%
400 \renewcommand{\doifglossarynoexistsordo}[3]{%
401 \ifglossaryexists{#1}%
402 {%
403 \glstrundefaction{Glossary type '#1' already exists}{#2}%
404 #3%
405 }%
406 {#2}%
407 }%
408 }
409 {%
410 \glstr@warnonexistsordo\doifglossarynoexistsordo
411 \newcommand{\doifglossarynoexistsordo}[3]{%
412 \ifglossaryexists{#1}%
413 {%
414 \glstrundefaction{Glossary type '#1' already exists}{#2}%

```



```

415      #3%
416    }%
417    {#2}%
418  }%
419 }
420

```

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

```

421 \appto\@newglossaryentryposthook{%
422   \ifdefvoid\@glo@see
423     {\csxdef{glo@\@glo@label @see}{}}%
424     {%
425       \csxdef{glo@\@glo@label @see}{\@glo@see}%
426       \@glxtr@autoindexcrossrefs
427     }%
428 }
429 \appto\@gls@keymap{,{see}{see}}

```

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

```

430 \newcommand*{\glxtrusesee}[1]{%
431   \glsdoifexists{#1}%
432   {%
433     \letcs{\@glo@see}{glo\@glsdetoklabel{#1}@see}%
434     \ifdefempty\@glo@see
435       {}%
436       {%
437         \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
438       }%
439     }%
440 }

```

`\glxtr@usesee`

```

441 \newcommand*{\glxtr@usesee}[1][\seename]{%
442   \@glxtr@usesee[#1]%
443 }

```

`\@glxtr@usesee`

```

444 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
445   \glxtruseseeformat{#1}{#2}%
446 }

```

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

```

447 \newcommand*{\glxtruseseeformat}[2]{%
448   \glsseeformat{#1}{#2}%
449 }

```

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

```

450 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}

```

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

```

451 \newcommand*\glstraddallcrossrefs{%
452   \forallglossaries{\@glo@type}%
453   {%
454     \forallsentries[\@glo@type]{\@glo@label}%
455     {%
456       \ifglused{\@glo@label}{\@glstr@addunusedxrefs{\@glo@label}}{}%
457     }%
458   }%
459 }

```

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

```

460 \newcommand*\@glstr@addunusedxrefs[1]{%
461   \letcs{\@glo@see}{glo@glsdetoklabel{#1}@see}%
462   \ifdefvoid\@glo@see
463   {%
464   {%
465     \expandafter\glstr@addunused\@glo@see\@end@glstr@addunused
466   }%
467 }

```

`\glstr@addunused` Adds all the entries if they haven't been used.

```

468 \newcommand*\glstr@addunused[1][]{%
469   \@glstr@addunused
470 }

```

`\glstr@addunused` Adds all the entries if they haven't been used.

```

471 \def\@glstr@addunused#1\@end@glstr@addunused{%
472   \@for\@glstr@label:=#1\do
473   {%
474     \ifglused{\@glstr@label}{}%
475     {%
476       \glstradd[format=glstrunusedformat]{\@glstr@label}%
477       \glstrunset{\@glstr@label}%
478       \@glstr@addunusedxrefs{\@glstr@label}%
479     }%
480   }%
481 }

```

`\glstrunusedformat`

```

482 \newcommand*\glstrunusedformat[1]{\unskip}

```

1.3.2 Document Definitions

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

```

483 \let\glstr@orgmakenoidxglossaries\makenoidxglossaries

```

```

484 \renewcommand{\makenoidxglossaries}{%
485   \glstr@orgmakenoidxglossaries
486   \ifglstrdocdefrestricted

```

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

```

487   \renewcommand*{\@gls@reference}[3]{%
488     \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
489     \ifinlistcs{##2}{@glsref@##1}%
490     {}%
491     {\listcsgadd{@glsref@##1}{##2}}%
492     \ifcsundef{glo@glstetoklabel{##2}@loclist}%
493     {\csgdef{glo@glstetoklabel{##2}@loclist}{}}%
494     {}%
495     \listcsgadd{glo@glstetoklabel{##2}@loclist}{##3}%
496   }%
497 \else

```

Disable document definitions.

```

498   \@glstrdocdeffalse
499 \fi
500 \disable@keys{glossaries-extra.sty}{docdef}%
501 }

```

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

```

502 \renewcommand*{\@gls@defdocnewglossaryentry}{%
503   \ifcase \@glstr@docdefval
504     \renewcommand*{\newglossaryentry}[2]{%
505       \PackageError{glossaries-extra}{Glossary entries must
506         be \MessageBreak defined in the preamble with \MessageBreak
507         package option 'docdef=false'\MessageBreak(consider using
508         'docdef=restricted')}{Move your glossary definitions to
509         the preamble. You can also put them in a \MessageBreak separate file
510         and load them with \string\loadglsentries.}%
511     }%
512   \or

```

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

```

513     \let\gls@checkseeallowed\relax
514     \let\newglossaryentry\new@glossaryentry
515   \or

```

Restricted mode just needs to allow the see value.

```

516     \let\gls@checkseeallowed\relax
517   \fi
518 }%

```

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`

```
519 \newcommand*{\GlsXtrEnableOnTheFly}{%
520   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
521 }
```

`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.)

```
522 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
523   \renewcommand*{\glsdetoklabel}[1]{%
524     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
525     {%
526       \expandafter\detokenize\expandafter{##1}%
527     }%
528     {\detokenize{##1}}}%
529   }%
530   \@GlsXtrEnableOnTheFly
531 }
532 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
533   \expandafter\if\glsbackslash#1%
534     #3%
535   \else
536     #4%
537   \fi
538 }
```

`sxtrstarflywarn`

```
539 \newcommand*{\glsxtrstarflywarn}{%
540   \GlossariesExtraWarning{Experimental starred version of
541     \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
542     read the warnings in the glossaries-extra user manual)}}%
543 }
```

`rEnableOnTheFly`

```
544 \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`

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

```

\glsxtr
546 \newcommand*\glsxtr[1] [] {%
547 \def\glsxtr@keylist{##1}%
548 \@glsxtr
549 }

\@glsxtr
550 \newcommand*\@glsxtr[2] [] {%
551 \ifglsentryexists{##2}%
552 {%
553 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
554 }%
555 {%
556 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
557 description={\nopostdesc},##1}%
558 }%
559 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
560 }

\Glsxtr
561 \newcommand*\Glsxtr[1] [] {%
562 \def\glsxtr@keylist{##1}%
563 \@Glsxtr
564 }

\@Glsxtr
565 \newcommand*\@Glsxtr[2] [] {%
566 \ifglsentryexists{##2}%
567 {%
568 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
569 }%
570 {%
571 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
572 description={\nopostdesc},##1}%
573 }%
574 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
575 }

\glsxtrpl
576 \newcommand*\glsxtrpl[1] [] {%
577 \def\glsxtr@keylist{##1}%
578 \@glsxtrpl
579 }

\@glsxtrpl
580 \newcommand*\@glsxtrpl[2] [] {%
581 \ifglsentryexists{##2}%
582 {%

```

```

583     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
584 }%
585 {%
586     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
587         description={\nopostdesc},##1}%
588 }%
589 \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
590 }

```

\Glsxtrpl

```

591 \newcommand*\Glsxtrpl[1][1]{%
592     \def\glsxtr@keylist{##1}%
593     \@Glsxtrpl
594 }

```

\@Glsxtrpl

```

595 \newcommand*\@Glsxtrpl[2][1]{%
596     \ifglsentryexists{##2}
597     {%
598         \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
599     }%
600     {%
601         \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
602             description={\nopostdesc},##1}%
603     }%
604     \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
605 }

```

\GlsXtrWarning

```

606 \newcommand*\GlsXtrWarning[2]{%
607     \def\glsxtr@optlist{##1}%
608     \@onelevel@sanitize\glsxtr@optlist
609     \GlossariesExtraWarning{The options '\glsxtr@optlist' have
610         been ignored for entry '##2' as it has already been defined}%
611 }

```

Disable commands after the glossary:

```

612 \renewcommand\@printglossary[2]{%
613     \def\glsxtr@printglossopts{##1}%
614     \@glsxtr@orgprintglossary{##1}{##2}%
615     \def\glsxtr{\glsxtr@disabledflycommand\glsxtr}%
616     \def\glsxtrpl{\glsxtr@disabledflycommand\glsxtrpl}%
617     \def\Glsxtr{\glsxtr@disabledflycommand\Glsxtr}%
618     \def\Glsxtrpl{\glsxtr@disabledflycommand\Glsxtrpl}%
619 }

```

\glsxtr@disabledflycommand

```

620 \newcommand*\glsxtr@disabledflycommand[1]{%
621     \PackageError{glossaries-extra}%

```

```

622   {\string##1\space can't be used after any of the \MessageBreak
623     glossaries have been displayed}%
624   {The on-the-fly commands enabled by
625     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
626     before the glossaries. If you want to use any entries \MessageBreak
627     after any of the glossaries, you must use the standard \MessageBreak
628     method of first defining the entry and then using the \MessageBreak
629     entry with commands like \string\gls}%
630     @@glxtr@disabledflycommand
631   }%
632   \newcommand*{@@glxtr@disabledflycommand}[2][\{##2}

```

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

```

633   \let\GlsXtrEnableOnTheFly\relax
634 }
635 \@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.

```

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

```

Modify \setglossarystyle to set the above.

etglossarystyle

```

637 \renewcommand*{\setglossarystyle}[1]{%
638   \ifcsundef{@glsstyle@#1}%
639   {%
640     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
641   }%
642   {%
643     \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

644     \protected@edef\@glxtr@current@style{#1}%
645   }%
646   \ifx\@glossary@default@style\relax
647     \protected@edef\@glossary@default@style{#1}%
648   \fi
649 }

```

In case we have an old version of glossaries:

```

650 \ifdef\@glossary@default@style
651 {}
652 {%
653   \let\@glossary@default@style\relax
654 }

```

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

```
655 \ifdef\glslistdottedwidth
656 {%
657   \ifdim\glslistdottedwidth=.5\hsize
658     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
659   \AtBeginDocument{%
660     \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
661       \setlength{\glslistdottedwidth}{.5\columnwidth}%
662     \fi
663   }%
664 \fi
665 }
666 {}%
```

Similarly for \glsdescwidth:

\glsdescwidth

```
667 \ifdef\glsdescwidth
668 {%
669   \ifdim\glsdescwidth=.6\hsize
670     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
671   \AtBeginDocument{%
672     \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
673       \setlength{\glsdescwidth}{.6\columnwidth}%
674     \fi
675   }%
676 \fi
677 }
678 {}%
```

and for \glspagelistwidth:

lspagelistwidth

```
679 \ifdef\glspagelistwidth
680 {%
681   \ifdim\glspagelistwidth=.1\hsize
682     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
683   \AtBeginDocument{%
684     \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
685       \setlength{\glspagelistwidth}{.1\columnwidth}%
686     \fi
687   }%
688 \fi
689 }
690 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```
691 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
```



```

692 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
693   \glsnonnumberlistfalse
694   \renewcommand*{\glossaryentrynumbers}[1]{%
695     \ifglsentryexists{\glscurrententrylabel}%
696     {%
697       \@glsxtrpreloctag
698       \GlsXtrFormatLocationList{#1}%
699       \@glsxtrpostloctag
700       \gls@save@numberlist{#1}%
701     }{}%
702   }%
703 \else
704   \glsnonnumberlisttrue
705   \renewcommand*{\glossaryentrynumbers}[1]{%
706     \ifglsentryexists{\glscurrententrylabel}%
707     {%
708       \gls@save@numberlist{#1}%
709     }{}%
710   }%
711 \fi

```

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

```

712 \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`

```

713 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
714   \let\@glsxtrpreloctag\@glsxtrpreloctag
715   \let\@glsxtrpostloctag\@glsxtrpostloctag
716   \renewcommand*{\@glsxtr@pagetag}{#1}%
717   \renewcommand*{\@glsxtr@pagestag}{#2}%
718   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
719     \csgdef{\@glsxtr@preloctag@##1}{##2}%
720   }%
721   \renewcommand*{\@glsxtr@doloctag}{%
722     \ifcsundef{\@glsxtr@preloctag@\glscurrententrylabel}%
723     {%
724       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
725         Rerun required}%
726     }%
727     {%
728       \csuse{\@glsxtr@preloctag@\glscurrententrylabel}%
729     }%

```

```

730 }%
731 }
732 \@onlypreamble\GlsXtrEnablePreLocationTag

glxstrpreloctag
733 \newcommand*{\@@glxstrpreloctag}{%
734   \let\@glxstr@org@delimN\delimN
735   \let\@glxstr@org@delimR\delimR
736   \let\@glxstr@org@glsgignore\glsgignore
   \gdef is required as the delimiters may occur inside a scope.
737   \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
738   \renewcommand*{\delimN}{%
739     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
740     \@glxstr@org@delimN}%
741   \renewcommand*{\delimR}{%
742     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
743     \@glxstr@org@delimR}%
744   \renewcommand*{\glsgignore}[1]{%
745     \gdef\@glxstr@thisloctag{\relax}%
746     \@glxstr@org@glsgignore{##1}}%
747   \@glxstr@doloctag
748 }

glxstrpreloctag
749 \newcommand*{\@glxstrpreloctag}{}

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

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

lsxtrpostloctag
752 \newcommand*{\@@glxstrpostloctag}{%
753   \let\delimN\@glxstr@org@delimN
754   \let\delimR\@glxstr@org@delimR
755   \let\glsgignore\@glxstr@org@glsgignore
756   \protected@write\@auxout{%
757     {\string\@glxstr@savepreloctag{\glscurrententrylabel}\@glxstr@thisloctag}}%
758 }

lsxtrpostloctag
759 \newcommand*{\@glxstrpostloctag}{}

lsxtr@preloctag
760 \newcommand*{\@glxstr@savepreloctag}[2]{}
761 \protected@write\@auxout{%
762   \string\providecommand\string\@glxstr@savepreloctag[2]{}

```

glsxtr@doloctag

```
763 \newcommand*{\@glsxtr@doloctag}{}
```

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

```
764 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
765   \XKV@plfalse
766   \XKV@sttrue
767   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
768   {%
769     \csname glsnonumberlist\XKV@resa\endcsname
770     \ifglsnonumberlist
771       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
772     \else
773       \def\glossaryentrynumbers##1{%
774         \@glsxtr@preloctag
775         \GlsXtrFormatLocationList{##1}%
776         \@glsxtr@postloctag
777         \gls@save@numberlist{##1}}%
778     \fi
779   }%
780 }
```

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.

```
781 \renewcommand*{\glsentryfmt}{%
782   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}{}}%
783   \glsifregular{\glslabel}%
784   {\glsxtrregularfont{\glsgenentryfmt}}%
785   {%
786     \ifglshasshort{\glslabel}%
787     {\glsxtrgenabbrvfmt}%
788     {\glsxtrregularfont{\glsgenentryfmt}}%
789   }%
790 }
```

sxtrregularfont Font used for regular entries.

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

Commands like \glsifplural are only used by the \gls-like commands in the glossaries package, but it might be useful for the postlink hook to know if the user has used, say,

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

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

```
792 \renewcommand{\@gls@field@link}[4] [] {%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the enter exists.

```
793 \@glsxtr@record{#2}{#3}%
794 \glsdoifexists{#3}%
795 {%
```

Save and restore the hyper setting (`\@gls@link` also does this, but that's too late if the optional argument of `\@gls@field@link` modifies it).

```
796 \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
797 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
798 \def\glscustomtext{#4}%
799 \@glsxtr@field@linkdefs
800 #1%
801 \@gls@link[#2]{#3}{#4}%
802 \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
803 }%
804 \glspostlinkhook
805 }
```

The commands `\gls`, `\Gls` etc don't use `\@gls@field@link`, so they need modifying as well to use `\@glsxtr@record`.

`\@gls@` Save the original definition and redefine.

```
806 \let\@glsxtr@org@gls@\@gls@
807 \def\@gls@#1#2{%
808 \@glsxtr@record{#1}{#2}%
809 \@glsxtr@org@gls@{#1}{#2}%
810 }%
```

`\@glspl@` Save the original definition and redefine.

```
811 \let\@glsxtr@org@glspl@\@glspl@
812 \def\@glspl@#1#2{%
813 \@glsxtr@record{#1}{#2}%
814 \@glsxtr@org@glspl@{#1}{#2}%
815 }%
```

`\@Gls@` Save the original definition and redefine.

```
816 \let\@glsxtr@org@Gls@\@Gls@
817 \def\@Gls@#1#2{%
818 \@glsxtr@record{#1}{#2}%
819 \@glsxtr@org@Gls@{#1}{#2}%
820 }%
```

```

\@Glspl@ Save the original definition and redefine.
821 \let\@glxtr@org@Glspl@\@Glspl@
822 \def\@Glspl@#1#2{%
823   \@glxtr@record{#1}{#2}%
824   \@glxtr@org@Glspl@{#1}{#2}%
825 }%

\@GLS@ Save the original definition and redefine.
826 \let\@glxtr@org@GLS@\@GLS@
827 \def\@GLS@#1#2{%
828   \@glxtr@record{#1}{#2}%
829   \@glxtr@org@GLS@{#1}{#2}%
830 }%

\@GLSpl@ Save the original definition and redefine.
831 \let\@glxtr@org@GLSpl@\@GLSpl@
832 \def\@GLSpl@#1#2{%
833   \@glxtr@record{#1}{#2}%
834   \@glxtr@org@GLSpl@{#1}{#2}%
835 }%

\@glsdisp1 Save the original definition and redefine.
836 \let\@glxtr@org@glsdisp\@glsdisp
837 \renewcommand*{\@glsdisp}[3][]{%
838   \@glxtr@record{#1}{#2}%
839   \@glxtr@org@glsdisp[#1]{#2}{#3}%
840 }

\@gls@link@ Redefine to include \@glxtr@record
841 \renewcommand*{\@gls@link}[3][]{%
842   \@glxtr@record{#1}{#2}%
843   \glsdoifexistsordo{#2}%
844   {%
845     \let\do@gls@link@checkfirsthyper\relax
846     \@gls@link[#1]{#2}{#3}%
847   }%
848   {%
849     \glstextformat{#3}%
850   }%
851   \glspostlinkhook
852 }

\glsadd Redefine to include \@glxtr@record
853 \renewrobustcmd*{\glsadd}[2][]{%
854   \@gls@adjustmode
855   \@glxtr@record{#1}{#2}%
856   \glsdoifexists{#2}%
857   {%

```

```

858 \def\@glsnumberformat{glsnumberformat}%
859 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
860 \setkeys{glossadd}{#1}%
861 \@gls@saveentrycounter
862 \@@do@wrglossary{#2}%
863 }%
864 }

```

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

```

865 \newcommand*{\@glsxtr@field@linkdefs}{%
866 \let\glsxtrifwasfirstuse\@secondoftwo
867 \let\glsifplural\@secondoftwo
868 \let\gls@scapscase\@firstofthree
869 \let\glsinsert\@empty
870 }

```

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

`assignfieldfont`

```

871 \newcommand*{\glsxtrassignfieldfont}[1]{%
872 \ifglsentryexists{#1}%
873 {%
874 \ifgls@hasshort{#1}%
875 {%
876 \glssetabbrvfmt{\gls@category{#1}}%
877 \glsifregular{#1}%
878 {\let\@gls@field@font\glsxtrregularfont}%
879 {\let\@gls@field@font\@firstofone}%
880 }%
881 {%
882 \glsifnotregular{#1}%
883 {\let\@gls@field@font\@firstofone}%
884 {\let\@gls@field@font\glsxtrregularfont}%
885 }%
886 }%
887 {%
888 \let\@gls@field@font\@gobble
889 }%
890 }

```

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

```

891 \def\@glstext@#1#2[#3]{%
892 \glsxtrassignfieldfont{#2}%
893 \@gls@field@link{#1}{#2}{\@gls@field@font{\gls@accesstext{#2}#3}}%
894 }

```

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

```

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

```

```

896 \glstrassignfieldfont{#2}%
897 \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
898 {\@gls@field@font{\Glsaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
899 }

```

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

```

900 \def\@Glstext@#1#2[#3]{%
901 \glstrassignfieldfont{#2}%
902 \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
903 {\@gls@field@font{\Glsaccesstext{#2}#3}}}%
904 }

```

Version 1.07 ensures that `\glsfirst` etc honours the `nohyperfirst` attribute. Allow a convenient way for the user to revert to ignoring this attribute for these commands.

`ecknohyperfirst`

```

905 \newcommand*\glstrchecknohyperfirst}[1]{%
906 \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
907 }

```

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

```

908 \def\@glsfirst@#1#2[#3]{%
909 \glstrassignfieldfont{#2}%
910
911 \@gls@field@link
912 [\let\glstrifwasfirstuse\@firstoftwo
913 \glstrchecknohyperfirst{#2}%
914 ]{#1}{#2}%
915 {\@gls@field@font{\glsaccessfirst{#2}#3}}}%
916 }

```

Ensure that `\glsfirst` honours the `nohyperfirst` attribute.

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

```

916 \def\@Glsfirst@#1#2[#3]{%
917 \glstrassignfieldfont{#2}%
918
919 \@gls@field@link
920 [\let\glstrifwasfirstuse\@firstoftwo
921 \let\glscapscase\@secondofthree
922 \glstrchecknohyperfirst{#2}%
923 ]%
924 {\@gls@field@font{\Glsaccessfirst{#2}#3}}}%
925 }

```

Ensure that `\Glsfirst` honours the `nohyperfirst` attribute.

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

```

925 \def\@GLSfirst@#1#2[#3]{%
926 \glstrassignfieldfont{#2}%

```

Ensure that \GLSfirst honours the nohyperfirst attribute.

```
927 \@gls@field@link
928 [\let\glstrifwasfirstuse\@firstoftwo
929 \let\glscapscase\@thirdofthree
930 \glstrchecknohyperfirst{#2}%
931 ]%
932 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
933 }
```

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

```
934 \def\@glsplural@#1#2[#3]{%
935 \glstrassignfieldfont{#2}%
936 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
937 {\@gls@field@font{\glsaccessplural{#2}#3}}%
938 }
```

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

```
939 \def\@Glsplural@#1#2[#3]{%
940 \glstrassignfieldfont{#2}%
941 \@gls@field@link
942 [\let\glsifplural\@firstoftwo
943 \let\glscapscase\@secondofthree
944 ]%
945 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
946 }
```

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

```
947 \def\@GLSplural@#1#2[#3]{%
948 \glstrassignfieldfont{#2}%
949 \@gls@field@link
950 [\let\glsifplural\@firstoftwo
951 \let\glscapscase\@thirdofthree
952 ]%
953 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
954 }
```

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

```
955 \def\glsfirstplural@#1#2[#3]{%
956 \glstrassignfieldfont{#2}%
957 \@gls@field@link
958 [\let\glstrifwasfirstuse\@firstoftwo
959 \let\glsifplural\@firstoftwo
960 \glstrchecknohyperfirst{#2}%
961 ]%
962 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
963 }
```


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

```
964 \def\@Glsfirstplural@#1#2[#3]{%
965   \glstrassignfieldfont{#2}%
   Ensure that \glfirstplural honours the nohyperfirst attribute.
966   \@gls@field@link
967   [\let\glstrifwasfirstuse\@firstoftwo
968   \let\glstifplural\@firstoftwo
969   \let\glscapscase\@secondofthree
970   \glstrchecknohyperfirst{#2}%
971   ]%
972   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
973 }
```

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

```
974 \def\@GLSfirstplural@#1#2[#3]{%
975   \glstrassignfieldfont{#2}%
   Ensure that \glfirstplural honours the nohyperfirst attribute.
976   \@gls@field@link
977   [\let\glstrifwasfirstuse\@firstoftwo
978   \let\glstifplural\@firstoftwo
979   \let\glscapscase\@thirdofthree
980   \glstrchecknohyperfirst{#2}%
981   ]%
982   {#1}{#2}%
983   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
984 }
```

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

```
985 \def\@glsname@#1#2[#3]{%
986   \glstrassignfieldfont{#2}%
987   \@gls@field@link{#1}{#2}{\@gls@field@font{\Glsaccessname{#2}#3}}%
988 }
```

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

```
989 \def\@Glsname@#1#2[#3]{%
990   \glstrassignfieldfont{#2}%
991   \@gls@field@link
992   [\let\glscapscase\@secondoftwo]{#1}{#2}%
993   {\@gls@field@font{\Glsaccessname{#2}#3}}%
994 }
```

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

```
995 \def\@GLSname@#1#2[#3]{%
996   \glstrassignfieldfont{#2}%
997   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
998   {#1}{#2}%
999   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
1000 }
```

```

\@Glsdesc@
1001 \def\@Glsdesc@#1#2[#3]{%
1002   \glstrassignfieldfont{#2}%
1003   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
1004 }

\@GLSdesc@   First letter uppercase version.
1005 \def\@GLSdesc@#1#2[#3]{%
1006   \glstrassignfieldfont{#2}%
1007   \@gls@field@link
1008   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1009   {\@gls@field@font{\@GLSaccessdesc{#2}#3}}%
1010 }

\@GLSdesc@   All uppercase version.
1011 \def\@GLSdesc@#1#2[#3]{%
1012   \glstrassignfieldfont{#2}%
1013   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1014   {#1}{#2}{\@gls@field@font{\@GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
1015 }

@Glsdescplural@   No case-changing version.
1016 \def\@Glsdescplural@#1#2[#3]{%
1017   \glstrassignfieldfont{#2}%
1018   \@gls@field@link
1019   [\let\glscapscase\@secondoftwo
1020    \let\glsifplural\@firstoftwo
1021    ]{#1}{#2}{\@gls@field@font{\@glsaccessdescplural{#2}#3}}%
1022 }

@Glsdescplural@   First letter uppercase version.
1023 \def\@Glsdescplural@#1#2[#3]{%
1024   \glstrassignfieldfont{#2}%
1025   \@gls@field@link
1026   [\let\glscapscase\@secondoftwo
1027    \let\glsifplural\@firstoftwo
1028    ]{#1}{#2}{\@gls@field@font{\@Glsaccessdescplural{#2}#3}}%
1029 }

@GLSdescplural@   All uppercase version.
1030 \def\@GLSdesc@#1#2[#3]{%
1031   \glstrassignfieldfont{#2}%
1032   \@gls@field@link
1033   [\let\glscapscase\@thirdoftwo
1034    \let\glsifplural\@firstoftwo
1035    ]%
1036    {#1}{#2}%
1037    {\@gls@field@font{\@GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
1038 }

```

```

\@glssymbol@
1039 \def\@glssymbol@#1#2[#3]{%
1040   \glstrassignfieldfont{#2}%
1041   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
1042 }

\@Glsymbol@   First letter uppercase version.
1043 \def\@Glsymbol@#1#2[#3]{%
1044   \glstrassignfieldfont{#2}%
1045   \@gls@field@link
1046   [\let\glscapscase\@secondoftwo]%
1047   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
1048 }

\@GLSsymbol@   All uppercase version.
1049 \def\@GLSsymbol@#1#2[#3]{%
1050   \glstrassignfieldfont{#2}%
1051   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1052   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
1053 }

lssymbolplural@   No case-changing version.
1054 \def\@lssymbolplural@#1#2[#3]{%
1055   \glstrassignfieldfont{#2}%
1056   \@gls@field@link
1057   [\let\glscapscase\@secondoftwo
1058   \let\glsifplural\@firstoftwo
1059   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
1060 }

lssymbolplural@   First letter uppercase version.
1061 \def\@lssymbolplural@#1#2[#3]{%
1062   \glstrassignfieldfont{#2}%
1063   \@gls@field@link
1064   [\let\glscapscase\@secondoftwo
1065   \let\glsifplural\@firstoftwo
1066   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
1067 }

LSsymbolplural@   All uppercase version.
1068 \def\@LSsymbol@#1#2[#3]{%
1069   \glstrassignfieldfont{#2}%
1070   \@gls@field@link
1071   [\let\glscapscase\@thirdoftwo
1072   \let\glsifplural\@firstoftwo
1073   ]%
1074   {#1}{#2}%
1075   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
1076 }

```

\@Glsuseri@ First letter uppercase version.

```
1077 \def\@Glsuseri@#1#2[#3]{%
1078   \glstrassignfieldfont{#2}%
1079   \@gls@field@link
1080   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1081   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
1082 }
```

\@GLSuseri@ All uppercase version.

```
1083 \def\@GLSuseri@#1#2[#3]{%
1084   \glstrassignfieldfont{#2}%
1085   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1086   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
1087 }
```

\@Glsuserii@ First letter uppercase version.

```
1088 \def\@Glsuserii@#1#2[#3]{%
1089   \glstrassignfieldfont{#2}%
1090   \@gls@field@link
1091   [\let\glscapscase\@secondoftwo]%
1092   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
1093 }
```

\@GLSuserii@ All uppercase version.

```
1094 \def\@GLSuserii@#1#2[#3]{%
1095   \glstrassignfieldfont{#2}%
1096   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1097   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
1098 }
```

\@Glsuseriii@ First letter uppercase version.

```
1099 \def\@Glsuseriii@#1#2[#3]{%
1100   \glstrassignfieldfont{#2}%
1101   \@gls@field@link
1102   [\let\glscapscase\@secondoftwo]%
1103   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
1104 }
```

\@GLSuseriii@ All uppercase version.

```
1105 \def\@GLSuseriii@#1#2[#3]{%
1106   \glstrassignfieldfont{#2}%
1107   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1108   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
1109 }
```

\@Glsuseriv@ First letter uppercase version.

```
1110 \def\@Glsuseriv@#1#2[#3]{%
1111   \glstrassignfieldfont{#2}%
1112 }
```

```

1112 \@gls@field@link
1113 [\let\glscapscase\@secondoftwo]%
1114 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
1115 }

```

\@GLSuseriv@ All uppercase version.

```

1116 \def\@GLSuseriv@#1#2[#3]{%
1117 \glstrassignfieldfont{#2}%
1118 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1119 {#1}{#2}%
1120 {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
1121 }

```

\@Glsuserv@ First letter uppercase version.

```

1122 \def\@Glsuserv@#1#2[#3]{%
1123 \glstrassignfieldfont{#2}%
1124 \@gls@field@link
1125 [\let\glscapscase\@secondoftwo]%
1126 {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
1127 }

```

\@GLSuserv@ All uppercase version.

```

1128 \def\@GLSuserv@#1#2[#3]{%
1129 \glstrassignfieldfont{#2}%
1130 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1131 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
1132 }

```

\@Glsuservi@ First letter uppercase version.

```

1133 \def\@Glsuservi@#1#2[#3]{%
1134 \glstrassignfieldfont{#2}%
1135 \@gls@field@link
1136 [\let\glscapscase\@secondoftwo]%
1137 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
1138 }

```

\@GLSuservi@ All uppercase version.

```

1139 \def\@GLSuservi@#1#2[#3]{%
1140 \glstrassignfieldfont{#2}%
1141 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1142 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
1143 }

```

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

\@acrshort No case change.

```

1144 \def\@acrshort#1#2[#3]{%

```

```

1145 \glsdoifexists{#2}%
1146 {%
1147   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1148   \let\glxtrifwasfirstuse\@secondoftwo
1149   \let\gl@ifplural\@secondoftwo
1150   \let\glscapscase\@firstofthree
1151   \let\glinsert\@empty
1152   \def\glscustomtext{%
1153     \acronymfont{\glaccessshort{#2}}#3%
1154   }%
1155   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1156 }%
1157 \glspostlinkhook
1158 }

```

\@Acrshort First letter uppercase.

```

1159 \def\@Acrshort#1#2[#3]{%
1160   \glsdoifexists{#2}%
1161   {%
1162     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1163     \let\glxtrifwasfirstuse\@secondoftwo
1164     \let\gl@ifplural\@secondoftwo
1165     \let\glscapscase\@secondofthree
1166     \let\glinsert\@empty
1167     \def\glscustomtext{%
1168       \acronymfont{\Glsaccessshort{#2}}#3%
1169     }%
1170     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1171   }%
1172   \glspostlinkhook
1173 }

```

\@ACRshort All uppercase.

```

1174 \def\@ACRshort#1#2[#3]{%
1175   \glsdoifexists{#2}%
1176   {%
1177     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1178     \let\glxtrifwasfirstuse\@secondoftwo
1179     \let\gl@ifplural\@secondoftwo
1180     \let\glscapscase\@thirdofthree
1181     \let\glinsert\@empty
1182     \def\glscustomtext{%
1183       \mfirstucMakeUppercase{\acronymfont{\glaccessshort{#2}}#3}%
1184     }%
1185     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1186   }%
1187   \glspostlinkhook
1188 }

```

\@acrshortpl No case change.

```
1189 \def\@acrshortpl#1#2[#3]{%
1190   \glsdoifexists{#2}%
1191   {%
1192     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1193     \let\glxtrifwasfirstuse\@secondoftwo
1194     \let\gl@sifplural\@firstoftwo
1195     \let\glscapscase\@firstofthree
1196     \let\gl$insert\@empty
1197     \def\glscustomtext{%
1198       \acronymfont{\gl@saccesssshortpl{#2}}#3%
1199     }%
1200     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1201   }%
1202   \glspostlinkhook
1203 }
```

\@Acrshortpl First letter uppercase.

```
1204 \def\@Acrshortpl#1#2[#3]{%
1205   \glsdoifexists{#2}%
1206   {%
1207     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1208     \let\glxtrifwasfirstuse\@secondoftwo
1209     \let\gl@sifplural\@firstoftwo
1210     \let\glscapscase\@secondofthree
1211     \let\gl$insert\@empty
1212     \def\glscustomtext{%
1213       \acronymfont{\Glsaccesssshortpl{#2}}#3%
1214     }%
1215     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1216   }%
1217   \glspostlinkhook
1218 }
```

\@ACRshortpl All uppercase.

```
1219 \def\@ACRshortpl#1#2[#3]{%
1220   \glsdoifexists{#2}%
1221   {%
1222     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1223     \let\glxtrifwasfirstuse\@secondoftwo
1224     \let\gl@sifplural\@firstoftwo
1225     \let\glscapscase\@thirdofthree
1226     \let\gl$insert\@empty
1227     \def\glscustomtext{%
1228       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshortpl{#2}}#3}%
1229     }%
1230     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1231   }%
1232   \glspostlinkhook
```

1233 }

\@acrlong No case change.

```
1234 \def\@acrlong#1#2[#3]{%
1235   \glsdoifexists{#2}%
1236   {%
1237     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1238     \let\glxtrifwasfirstuse\@secondoftwo
1239     \let\glsifplural\@secondoftwo
1240     \let\glscapscase\@firstofthree
1241     \let\glsinsert\@empty
1242     \def\glscustomtext{%
1243       \acronymfont{\glsaccesslong{#2}}#3%
1244     }%
1245     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1246   }%
1247   \glspostlinkhook
1248 }
```

\@Acrlong First letter uppercase.

```
1249 \def\@Acrlong#1#2[#3]{%
1250   \glsdoifexists{#2}%
1251   {%
1252     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1253     \let\glxtrifwasfirstuse\@secondoftwo
1254     \let\glsifplural\@secondoftwo
1255     \let\glscapscase\@secondofthree
1256     \let\glsinsert\@empty
1257     \def\glscustomtext{%
1258       \acronymfont{\Glsaccesslong{#2}}#3%
1259     }%
1260     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1261   }%
1262   \glspostlinkhook
1263 }
```

\@ACRlong All uppercase.

```
1264 \def\@ACRlong#1#2[#3]{%
1265   \glsdoifexists{#2}%
1266   {%
1267     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1268     \let\glxtrifwasfirstuse\@secondoftwo
1269     \let\glsifplural\@secondoftwo
1270     \let\glscapscase\@thirdofthree
1271     \let\glsinsert\@empty
1272     \def\glscustomtext{%
1273       \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
1274     }%
1275     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1276   }%
1277 }
```



```

1276 }%
1277 \glspostlinkhook
1278 }

```

\@acrlongpl No case change.

```

1279 \def\@acrlongpl#1#2[#3]{%
1280 \glsdoidexists{#2}%
1281 {%
1282 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1283 \let\glxtrifwasfirstuse\@secondoftwo
1284 \let\gl@sifplural\@firstoftwo
1285 \let\glscapscase\@firstofthree
1286 \let\gl$insert\@empty
1287 \def\glscustomtext{%
1288 \acronymfont{\gl@saccesslongpl{#2}}#3%
1289 }%
1290 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1291 }%
1292 \glspostlinkhook
1293 }

```

\@Acrlongpl First letter uppercase.

```

1294 \def\@Acrlongpl#1#2[#3]{%
1295 \glsdoidexists{#2}%
1296 {%
1297 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1298 \let\glxtrifwasfirstuse\@secondoftwo
1299 \let\gl@sifplural\@firstoftwo
1300 \let\glscapscase\@secondofthree
1301 \let\gl$insert\@empty
1302 \def\glscustomtext{%
1303 \acronymfont{\Glsaccesslongpl{#2}}#3%
1304 }%
1305 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1306 }%
1307 \glspostlinkhook
1308 }

```

\@ACRlongpl All uppercase.

```

1309 \def\@ACRlongpl#1#2[#3]{%
1310 \glsdoidexists{#2}%
1311 {%
1312 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1313 \let\glxtrifwasfirstuse\@secondoftwo
1314 \let\gl@sifplural\@firstoftwo
1315 \let\glscapscase\@thirdofthree
1316 \let\gl$insert\@empty
1317 \def\glscustomtext{%
1318 \mfirstucMakeUppercase{\acronymfont{\gl@saccesslongpl{#2}}#3}%

```

```

1319 }%
1320 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1321 }%
1322 \glspostlinkhook
1323 }

```

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

\@glsaddkey

```

1324 \renewcommand*{\@glsaddkey}[7]{%
1325   \key@ifundefined{glossentry}{#1}%
1326   {%
1327     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1328     \appto\@gls@keymap{,{#1}{#1}}%
1329     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
1330     \appto\@newglossaryentryposthook{%
1331       \letcs{\@glo@tmp}{@glo@#1}%
1332       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1333     }%
1334     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1335     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1336   \ifcsdef{@gls@user@#1@}%
1337   {%
1338     \PackageError{glossaries}%
1339     {Can't define '\string#5' as helper command
1340     '\expandafter\string\csname @gls@user@#1@\endcsname' already
1341     exists}%
1342   }%
1343 }%
1344 {%
1345   \expandafter\newcommand\expandafter*\expandafter
1346   {\csname @gls@user@#1\endcsname}[2][ ]{%
1347     \new@ifnextchar[%
1348       {\csuse{@gls@user@#1@}{##1}{##2}}%
1349       {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
1350   \csdef{@gls@user@#1@}##1##2[##3]{%
1351     \@gls@field@link{##1}{##2}{#3{##2}##3}%
1352   }%
1353   \newrobustcmd*{#5}{%
1354     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
1355 }%

```

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

```

1356   \ifcsdef{@Gls@user@#1@}%
1357   {%
1358     \PackageError{glossaries}%
1359     {Can't define '\string#6' as helper command

```

```

1360         '\expandafter\string\csname @Gls@user@#1@\endcsname' already
1361         exists}%
1362     {}%
1363 }%
1364 {%
1365     \expandafter\newcommand\expandafter*\expandafter
1366     {\csname @Gls@user@#1@\endcsname}[2][\%
1367         \new@ifnextchar[%
1368             {\csuse{@Gls@user@#1@}{##1}{##2}}}%
1369             {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
1370     \csdef{@Gls@user@#1@}##1##2[##3]{%
1371         \@gls@field@link[\let\glscapscase\@secondofthree]%
1372         {##1}{##2}{#4{##2}##3}%
1373     }%
1374     \newrobustcmd*{#6}{%
1375         \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
1376     }%

```

Finally the all caps version (modified):

```

1377     \ifcsdef{@GLS@user@#1@}%
1378     {%
1379         \PackageError{glossaries}%
1380         {Can't define '\string#7' as helper command
1381         '\expandafter\string\csname @GLS@user@#1@\endcsname' already
1382         exists}%
1383     {}%
1384 }%
1385 {%
1386     \expandafter\newcommand\expandafter*\expandafter
1387     {\csname @GLS@user@#1@\endcsname}[2][\%
1388         \new@ifnextchar[%
1389             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
1390             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
1391     \csdef{@GLS@user@#1@}##1##2[##3]{%
1392         \@gls@field@link[\let\glscapscase\@thirdofthree]%
1393         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
1394     }%
1395     \newrobustcmd*{#7}{%
1396         \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
1397     }%
1398 }%
1399 {%
1400     \PackageError{glossaries-extra}{Key '#1' already exists}{}%
1401 }%
1402 }

```

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

```

1403 \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.

```
1404 \let\@glstr@org@checkfirsthyper\@gls@link@checkfirsthyper
1405 \renewcommand*{\@gls@link@checkfirsthyper}{%
```

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

```
1406 \ifglsused{\glslabel}%
1407 {\let\glstrifwasfirstuse\@secondoftwo}
1408 {\let\glstrifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
1409 \edef\glscategorylabel{\glscategory{\glslabel}}%
1410 \ifglsused{\glslabel}%
1411 {%
1412   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
1413   {\KV@glslink@hyperfalse}{}%
1414 }%
1415 {%
1416   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
1417   {\KV@glslink@hyperfalse}{}%
1418 }%
1419 \glslinkcheckfirsthyperhook
1420 }
```

ablehyperinlist 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.

```
1421 \ifdef\do@glstdisablehyperinlist
1422 {%
1423   \let\@glstr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
1424   \renewcommand*{\do@glstdisablehyperinlist}{%
1425     \@glstr@do@glstdisablehyperinlist
1426     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
1427   }
1428 }
1429 }
```

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

```
1430 \define@boolkey{glslink}{noindex}[true]{}
1431 \KV@glslink@noindexfalse
```

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

lt@glslink@opts

```
1432 \ifdef\@gls@setdefault@glslink@opts
1433 {
1434   \renewcommand*{\@gls@setdefault@glslink@opts}{%
1435     \KV@glslink@noindexfalse
```

```

1436 }
1437 }
1438 {
    Not defined so prepend it to \do@glssdisablehyperinlist to achieve the same effect.
1439 \newcommand*{\@gls@setdefault@glslink@opts}{%
1440 \KV@glslink@noindexfalse
1441 }
1442 \preto\do@glssdisablehyperinlist{\@gls@setdefault@glslink@opts}
1443 }

```

DefaultGlsOpts Set the default options for \glslink etc.

```

1444 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
1445 \renewcommand*{\@gls@setdefault@glslink@opts}{\setkeys{glslink}{#1}}%
1446 }

```

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

```

1447 \newcommand*{\glstrifindexing}[2]{%
1448 \ifKV@glslink@noindex #2\else #1\fi
1449 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

1450 \renewcommand*{\glswriteentry}[2]{%
1451 \glstrifindexing
1452 {%
1453 \ifglssindexonlyfirst
1454 \ifglssused{#1}
1455 {\glstrdoautoindexname{#1}{dualindex}}%
1456 {#2}%
1457 \else
1458 \glssifattribute{#1}{indexonlyfirst}{true}%
1459 {\ifglssused{#1}
1460 {\glstrdoautoindexname{#1}{dualindex}}%
1461 {#2}}%
1462 {#2}%
1463 \fi
1464 }%
1465 {}%
1466 }

```

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

```

1467 \appto\@do@@wrglossary{\@glstrdo@@wrindex
1468 \glstrdowrglossaryhook{\@gls@label}}%
1469 }

```

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

Similarly for the “noidx” version:

s@noidxglossary

```
1470 \appto\gls@noidxglossary{\@glstrdo@wrindex
1471 \glstrdowrglossaryhook{\@gls@label}}%
1472 }
```

xtrdo@wrindex

```
1473 \newcommand*{\@glstrdo@wrindex}{%
1474 \glstrdoautoindexname{\@gls@label}{dualindex}}%
1475 }
```

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.)

```
1476 \newcommand*{\glstrdowrglossaryhook}[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.

```
1477 \newcommand*{\@gls@alt@hyp@opt}[1]{%
1478 \let\glslinkvar\@firstofthree
1479 \let\@gls@hyp@opt@cs#1\relax
1480 \@ifstar{\s@gls@hyp@opt}%
1481 {\@ifnextchar+%
1482 {\@firstoftwo{\p@gls@hyp@opt}}%
1483 {%
1484 \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
1485 {\@firstoftwo{\@alt@gls@hyp@opt}}%
1486 {\#1}%
1487 }%
1488 }%
1489 }
```

alt@gls@hyp@opt User version

```
1490 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
1491 \let\glslinkvar\@firstofthree
1492 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
1493 \newcommand*{\@gls@alt@hyp@opt@char}{{}
```

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

```
1494 \newcommand*{\@gls@alt@hyp@opt@keys}{{}
```

rSetAltModifier

```
1495 \newcommand*{\GlsXtrSetAltModifier}[2]{%
1496 \let\@gls@hyp@opt\@gls@alt@hyp@opt
1497 \def\@gls@alt@hyp@opt@char{\#1}%
1498 \def\@gls@alt@hyp@opt@keys{\#2}%
1499 }
```

`\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 `\glentryshort` and, similarly, the long form commands like `\acrlong` and `\glxtrlong` to use `\glentrylong`.

```
1500 \renewcommand*{\glsdohyperlink}[2]{%
1501   \hyperlink{#1}{\glxtrprotectlinks#2}}
```

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

```
1502 \ifundef\glsdonohyperlink
1503 {%
1504   \renewcommand{\glsdisablehyper}{%
1505     \KV@glslink@hyperfalse
1506     \let\@glslink\glsdonohyperlink
1507     \let\@glstarget\@secondoftwo
1508   }
1509 }
1510 {}
```

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

```
1511 \def\glsdonohyperlink#1#2{{\glxtrprotectlinks #2}}
```

Reset `\@glslink` with patched versions:

```
1512 \ifcsundef{hyperlink}%
1513 {%
1514   \let\@glslink\glsdonohyperlink
1515 }%
1516 {%
1517   \let\@glslink\glsdohyperlink
1518 }
```

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

```
1519 \newcommand*{\glxtrprotectlinks}{%
1520   \KV@glslink@hyperfalse
1521   \KV@glslink@noindextrue
1522   \let\@gls@\@glxtrp@text@
1523   \let\@Gls@\@Glsxtrp@text@
1524   \let\@GLS@\@GLSxtrp@text@
1525   \let\@glspl@\@glxtrp@plural@
1526   \let\@Glspl@\@Glsxtrp@plural@
1527   \let\@GLSpl@\@GLSxtrp@plural@
1528   \let\@glxtrshort@\glxtrp@short@
```

```

1529 \let\@Glsxtrshort\@Glsxtrp@short@
1530 \let\@GLSxtrshort\@GLSxtrp@short@
1531 \let\@glsxtrlong\@glsxtrp@long@
1532 \let\@Glsxtrlong\@Glsxtrp@long@
1533 \let\@GLSxtrlong\@GLSxtrp@long@
1534 \let\@glsxtrshortpl\@glsxtrp@shortpl@
1535 \let\@Glsxtrshortpl\@Glsxtrp@shortpl@
1536 \let\@GLSxtrshortpl\@GLSxtrp@shortpl@
1537 \let\@glsxtrlongpl\@glsxtrp@longpl@
1538 \let\@Glsxtrlongpl\@Glsxtrp@longpl@
1539 \let\@GLSxtrlongpl\@GLSxtrp@longpl@
1540 \let\@acrshort\@glsxtrp@acrshort@
1541 \let\@Acrshort\@Glsxtrp@acrshort@
1542 \let\@ACRshort\@GLSxtrp@acrshort@
1543 \let\@acrshortpl\@glsxtrp@acrshortpl@
1544 \let\@Acrshortpl\@Glsxtrp@acrshortpl@
1545 \let\@ACRshortpl\@GLSxtrp@acrshortpl@
1546 \let\@acrlong\@glsxtrp@acrlong@
1547 \let\@Acrlong\@Glsxtrp@acrlong@
1548 \let\@ACRlong\@GLSxtrp@acrlong@
1549 \let\@acrlongpl\@glsxtrp@acrlongpl@
1550 \let\@Acrlongpl\@Glsxtrp@acrlongpl@
1551 \let\@ACRlongpl\@GLSxtrp@acrlongpl@
1552 }

```

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

@glsxtrp@text@

```
1553 \def\@glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@Glsxtrp@text@

```
1554 \def\@Glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtrp@text@

```
1555 \def\@GLSxtrp@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtrp@plural@

```
1556 \def\@glsxtrp@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lSxtrp@plural@

```
1557 \def\@Glsxtrp@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtrp@plural@

```
1558 \def\@GLSxtrp@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glsxtrp@short@

```
1559 \def\@glsxtrp@short@#1#2[#3]{%
```

```
1560 {%
```



```

1561 \glsetabbrvfmt{\glscategory{#2}}%
1562 \glabbrvfont{\glentryshort{#2}}#3%
1563 }%
1564 }

```

Glsxtr@p@short@

```

1565 \def\@Glsxtr@p@short@#1#2[#3]{%
1566 {%
1567 \glsetabbrvfmt{\glscategory{#2}}%
1568 \glabbrvfont{\glentryshort{#2}}#3%
1569 }%
1570 }

```

GLSxtr@p@short@

```

1571 \def\@GLSxtr@p@short@#1#2[#3]{%
1572 {%
1573 \glsetabbrvfmt{\glscategory{#2}}%
1574 \mfirstucMakeUppercase{\glabbrvfont{\glentryshort{#2}}#3}%
1575 }%
1576 }

```

sxtr@p@shortpl@

```

1577 \def\@glxtr@p@shortpl@#1#2[#3]{%
1578 {%
1579 \glsetabbrvfmt{\glscategory{#2}}%
1580 \glabbrvfont{\glentryshortpl{#2}}#3%
1581 }%
1582 }

```

sxtr@p@shortpl@

```

1583 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
1584 {%
1585 \glsetabbrvfmt{\glscategory{#2}}%
1586 \glabbrvfont{\glentryshortpl{#2}}#3%
1587 }%
1588 }

```

Sxtr@p@shortpl@

```

1589 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
1590 {%
1591 \glsetabbrvfmt{\glscategory{#2}}%
1592 \mfirstucMakeUppercase{\glabbrvfont{\glentryshortpl{#2}}#3}%
1593 }%
1594 }

```

@glxtr@p@long@

```

1595 \def\@glxtr@p@long@#1#2[#3]{\glentrylong{#2}#3}

```

```

@Glsxtr@p@long@
1596 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}}

@GLSxtr@p@long@
1597 \def\@GLSxtr@p@long@#1#2[#3]{%
1598   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylong{#2}}#3}}}

lsxtr@p@longpl@
1599 \def\@glxtr@p@longpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

lsxtr@p@longpl@
1600 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsentrylongpl{#2}}#3}}

LSxtr@p@longpl@
1601 \def\@GLSxtr@p@longpl@#1#2[#3]{%
1602   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
1603 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
1604 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
1605 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
1606   {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
1607 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1608 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1609 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
1610   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
1611 \def\@glxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
1612 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
1613 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
1614   {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}

```

tr@p@acrlongpl@

```
1615 \def\@glxstrp@acrlongpl@#1#2[#3]{\@glstrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
1616 \def\@Glsxtrp@acrlongpl@#1#2[#3]{\@Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
1617 \def\@GLSxtrp@acrlongpl@#1#2[#3]{%
```

```
1618 {\mfirstucMakeUppercase{\@glstrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glxstrp@opt

```
1619 \newcommand*{\@glxstrp@opt}{hyper=false,noindex}
```

\glxstrsetpopts Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.

```
1620 \newcommand*{\glxstrsetpopts}[1]{%
```

```
1621 \renewcommand*{\@glxstrp@opt}{#1}%
```

```
1622 }
```

lossxtrsetpopts Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.

```
1623 \newcommand*{\lossxtrsetpopts}{%
```

```
1624 \glxstrsetpopts{noindex}%
```

```
1625 }
```

\@@glxstrp

```
1626 \newrobustcmd*{\@@glxstrp}[2]{%
```

Add scope.

```
1627 {%
```

```
1628 \let\glspostlinkhook\relax
```

```
1629 \csname#1\expandafter\endcsname\expandafter[\@glxstrp@opt]{#2}[]%
```

```
1630 }%
```

```
1631 }
```

\@glxstrp

```
1632 \newrobustcmd*{\@glxstrp}[2]{%
```

```
1633 \ifcsdef{gls#1}%
```

```
1634 {%
```

```
1635 \@glxstrp{gls#1}{#2}%
```

```
1636 }%
```

```
1637 {%
```

```
1638 \ifcsdef{glsxtr#1}%
```

```
1639 {%
```

```
1640 \@glxstrp{glsxtr#1}{#2}%
```

```
1641 }%
```

```
1642 {%
```

```
1643 \PackageError{glossaries-extra}{‘#1’ not recognised by
```

```
1644 \string\glxstrp}{%}
```

```

1645 }%
1646 }%
1647 }

```

\@Glsxtrp

```

1648 \newrobustcmd*{\@Glsxtrp}[2]{%
1649   \ifcsdef{Gls#1}%
1650   {%
1651     \@glsxtrp{Gls#1}{#2}%
1652   }%
1653   {%
1654     \ifcsdef{Glsxtr#1}%
1655     {%
1656       \@glsxtrp{Glsxtr#1}{#2}%
1657     }%
1658     {%
1659       \PackageError{glossaries-extra}{‘#1’ not recognised by
1660         \string\Glsxtrp}{}%
1661     }%
1662   }%
1663 }

```

\@GLSxtrp

```

1664 \newrobustcmd*{\@GLSxtrp}[2]{%
1665   \ifcsdef{GLS#1}%
1666   {%
1667     \@glsxtrp{GLS#1}{#2}%
1668   }%
1669   {%
1670     \ifcsdef{GLSxtr#1}%
1671     {%
1672       \@glsxtrp{GLSxtr#1}{#2}%
1673     }%
1674     {%
1675       \PackageError{glossaries-extra}{‘#1’ not recognised by
1676         \string\GLSxtrp}{}%
1677     }%
1678   }%
1679 }

```

\glsxtr@entry@p

```

1680 \newrobustcmd*{\glsxtr@entry@p}[2]{%
1681   \glsifattribute{#1}{headuc}{true}%
1682   {%
1683     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
1684   }%
1685   {%
1686     \@gls@entry@field{#1}{#2}%
1687   }%

```

1688 }

\glstrp Not robust as it needs to expand somewhat.

```
1689 \ifdef\teorpdfstring
1690 {
1691   \newcommand{\glstrp}[2]{%
1692     \protect\NoCaseChange
1693     {%
1694       \protect\teorpdfstring
1695       {%
1696         \protect\glstrifinmark
1697         {%
1698           \ifcsdef{glstrhead#1}%
1699           {%
1700             {\protect\csuse{glstrhead#1}{#2}}%
1701             }%
1702             {%
1703               \glstr@headentry@p{#2}{#1}%
1704               }%
1705             }%
1706             {%
1707               \@glstrp{#1}{#2}%
1708               }%
1709             }%
1710             {%
1711               \protect\@glstr@entry@field{#2}{#1}%
1712               }%
1713             }%
1714           }
1715 }
1716 {
1717   \newcommand{\glstrp}[2]{%
1718     \protect\NoCaseChange
1719     {%
1720       \protect\glstrifinmark
1721       {%
1722         \ifcsdef{glstrhead#1}%
1723         {%
1724           {\protect\csuse{glstrhead#1}}%
1725           }%
1726           {%
1727             \glstr@headentry@p{#2}{#1}%
1728             }%
1729           }%
1730           {%
1731             \@glstrp{#1}{#2}%
1732             }%
1733           }%
1734         }
```

1735 }

Provide short synonyms for the most common option.

`\glsps`

```
1736 \newcommand*{\glsps}{\glsxtrp{short}}
```

`\glspt`

```
1737 \newcommand*{\glspt}{\glsxtrp{text}}
```

`\Glsxtrp` As above but use first letter upper case (but not for the bookmarks, which can't process `\uppercase`).

```
1738 \ifdef\teorpdfstring
1739 {
1740   \newcommand{\Glsxtrp}[2]{%
1741     \protect\NoCaseChange
1742     {%
1743       \protect\teorpdfstring
1744       {%
1745         \protect\glsxtrifinmark
1746         {%
1747           \ifcsdef{Glsxtrhead#1}%
1748             {%
1749               {\protect\csuse{Glsxtrhead#1}{#2}}%
1750             }%
1751             {%
1752               \protect\@Gls@entry@field{#2}{#1}%
1753             }%
1754             }%
1755             {%
1756               \@Glsxtrp{#1}{#2}%
1757             }%
1758             }%
1759             {%
1760               \protect\@gls@entry@field{#2}{#1}%
1761             }%
1762             }%
1763         }
1764     }
1765 {
1766   \newcommand{\Glsxtrp}[2]{%
1767     \protect\NoCaseChange
1768     {%
1769       \protect\glsxtrifinmark
1770       {%
1771         \ifcsdef{Glsxtrhead#1}%
1772           {%
1773             {\protect\csuse{Glsxtrhead#1}}%
1774           }%

```

```

1775      {%
1776      \protect\@Gls@entry@field{#2}{#1}%
1777      }%
1778    }%
1779    {%
1780    \@Glsxtrp{#1}{#2}%
1781    }%
1782  }%
1783 }
1784 }

```

\GLSxtrp As above but all upper case (but not for the bookmarks, which can't process \uppercase).

```

1785 \ifdef\teorpdfstring
1786 {
1787   \newcommand{\GLSxtrp}[2]{%
1788     \protect\NoCaseChange
1789     {%
1790       \protect\teorpdfstring
1791       {%
1792         \protect\glsxtrifinmark
1793         {%
1794           \ifcsdef{GLSxtr#1}%
1795           {%
1796             {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
1797           }%
1798           {%
1799             \protect\mfirstucMakeUppercase
1800             {%
1801               \protect\@gls@entry@field{#2}{#1}%
1802             }%
1803           }%
1804         }%
1805       }%
1806       \@GLSxtrp{#1}{#2}%
1807     }%
1808   }%
1809   {%
1810     \protect\@gls@entry@field{#2}{#1}%
1811   }%
1812 }%
1813 }
1814 }
1815 {
1816   \newcommand{\GLSxtrp}[2]{%
1817     \protect\NoCaseChange
1818     {%
1819       \protect\glsxtrifinmark
1820       {%
1821         \ifcsdef{GLSxtr#1}%

```

```

1822      {%
1823      {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
1824      }%
1825      {%
1826      \protect\mfirstucMakeUppercase
1827      {%
1828      \protect\@gls@entry@field{#2}{#1}%
1829      }%
1830      }%
1831      }%
1832      {%
1833      \@GLSxtrp{#1}{#2}%
1834      }%
1835      }%
1836    }
1837 }

```

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.

```

1838 \renewcommand*{\@glsunset}[1]{%
1839   \@@glsunset{#1}%
1840   \glsxtrpostunset{#1}%
1841 }%

```

glsxtrpostunset

```

1842 \newcommand*{\glsxtrpostunset}[1]{%

```

\@glslocalunset Local unset.

```

1843   \renewcommand*{\@glslocalunset}[1]{%
1844     \@@glslocalunset{#1}%
1845     \glsxtrpostlocalunset{#1}%
1846   }%

```

rpostlocalunset

```

1847 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

\@glsreset Global reset.

```

1848 \renewcommand*{\@glsreset}[1]{%
1849   \@@glsreset{#1}%
1850   \glsxtrpostreset{#1}%
1851 }%

```


glsxtrpostreset

```
1852 \newcommand*{\glsxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
1853 \renewcommand*{\@glslocalreset}[1]{%
1854   \@glslocalreset{#1}%
1855   \glsxtrpostlocalreset{#1}%
1856 }%
```

rpostlocalreset

```
1857 \newcommand*{\glsxtrpostlocalreset}[1]{}
```

leEntryCounting The first argument is the list of categories and the second argument is the value of the entrycount attribute.

```
1858 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
1859   \glsenableentrycount
```

Redefine \gls etc:

```
1860   \renewcommand*{\gls}{\cglsl}%
1861   \renewcommand*{\Gls}{\cGls}%
1862   \renewcommand*{\glspl}{\cglspl}%
1863   \renewcommand*{\Glspl}{\cGlspl}%
1864   \renewcommand*{\GLS}{\cGLS}%
1865   \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
1866   \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
1867   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
1868   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
1869     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
1870       can't be used with \string\GlsXtrEnableEntryCounting}%
1871     {Use one or other but not both commands}}%
1872 }
```

ycountunsetattr

```
1873 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
1874   \@for\@glsxtr@cat:=#1\do
1875   {%
1876     \ifdefempty{\@glsxtr@cat}{}%
1877     {%
1878       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
1879     }%
1880   }%
1881 }
```

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

nableentrycount

```
1882 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
1883 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
1884 \renewcommand*{\gls@defdocnewglossaryentry}{%
1885 \renewcommand*\newglossaryentry[2]{%
1886 \PackageError{glossaries}{\string\newglossaryentry\space
1887 may only be used in the preamble when entry counting has
1888 been activated}{If you use \string\glsenableentrycount\space
1889 you must place all entry definitions in the preamble not in
1890 the document environment}%
1891 }%
1892 }%
```

New commands to access new fields:

```
1893 \newcommand*{\glsentrycurrcount}[1]{%
1894 \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
1895 {0}{\@gls@entry@field{##1}{currcount}}}%
1896 }%
1897 \newcommand*{\glsentryprevcount}[1]{%
1898 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
1899 {0}{\@gls@entry@field{##1}{prevcount}}}%
1900 }%
```

Adjust post unset and reset:

```
1901 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
1902 \renewcommand*{\glsxtrpostunset}[1]{%
1903 \@glsxtr@entrycount@org@unset{##1}%
1904 \@gls@increment@currcount{##1}%
1905 }%
1906 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
1907 \renewcommand*{\glsxtrpostlocalunset}[1]{%
1908 \@glsxtr@entrycount@org@localunset{##1}%
1909 \@gls@local@increment@currcount{##1}%
1910 }%
1911 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
1912 \renewcommand*{\glsxtrpostreset}[1]{%
1913 \@glsxtr@entrycount@org@reset{##1}%
1914 \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
1915 }%
1916 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
1917 \renewcommand*{\glsxtrpostlocalreset}[1]{%
1918 \@glsxtr@entrycount@org@localreset{##1}%
1919 \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
1920 }%
```

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

```

1921 \let\@cglS@\@cglS@
1922 \let\@cglSpl@\@cglSpl@
1923 \let\@cGLS@\@cGLS@
1924 \let\@cGlSpl@\@cGlSpl@
1925 \let\@cGLS@\@cGLS@
1926 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

1927 \AtEndDocument{\@glS@write@entrycounts}%
1928 \renewcommand*{\@glS@entry@count}[2]{%
1929   \csgdef{glo@\glSdetoklabel{##1}@prevcount}{##2}%
1930 }%
1931 \let\glSenableentrycount\relax
1932 \renewcommand*{\glSenableentryunitcount}{%
1933   \PackageError{glossaries-extra}{\string\glSenableentryunitcount\space
1934     can't be used with \string\glSenableentrycount}%
1935   {Use one or other but not both commands}%
1936 }%
1937 }

```

`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.

```

1938 \renewcommand*{\@glS@write@entrycounts}{%
1939   \immediate\write\@auxout
1940     {\string\providecommand*{\string\@glS@entry@count}[2]{}}%
1941   \count@=0\relax
1942   \forallglSentries{\@glSentry}{%
1943     \glShasattribute{\@glSentry}{entrycount}%
1944     {%
1945       \ifglSused{\@glSentry}%
1946       {%
1947         \immediate\write\@auxout
1948           {\string\@glS@entry@count{\@glSentry}{\glSentrycurrcount{\@glSentry}}}%
1949       }%
1950     }%
1951     \advance\count@ by \@ne
1952   }%
1953 }%
1954 }%
1955 \ifnum\count@=0
1956   \GlossariesExtraWarningNoLine{Entry counting has been enabled
1957     \MessageBreak with \string\glSenableentrycount\space but the
1958     \MessageBreak attribute 'entrycount' hasn't
1959     \MessageBreak been assigned to any of the defined
1960     \MessageBreak entries}%
1961 \fi
1962 }

```

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

```

1963 \newcommand*{\glxtrifcounttrigger}[3]{%
1964   \glshasattribute{#1}{entrycount}%
1965   {%
1966     \ifnum\gl Sentryprevcount{#1}>\gl sgetattribute{#1}{entrycount}\relax
1967     #3%
1968   \else
1969     #2%
1970   \fi
1971 }%
1972 {#3}%
1973 }

```

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

`\@@cgl`s@

```

1974 \def\@@cgl#1#2[#3]{%
1975   \glxtrifcounttrigger{#2}%
1976   {%
1977     \cgl sformat{#2}{#3}%
1978     \gl sunset{#2}%
1979   }%
1980   {%
1981     \@gls@{#1}{#2}[#3]%
1982   }%
1983 }%

```

`\@@cgl`s@

```

1984 \def\@@cgl spl#1#2[#3]{%
1985   \glxtrifcounttrigger{#2}%
1986   {%
1987     \cgl splformat{#2}{#3}%
1988     \gl sunset{#2}%
1989   }%
1990   {%
1991     \@glspl@{#1}{#2}[#3]%
1992   }%
1993 }%

```

`\@@cGls`@

```

1994 \def\@@cGls#1#2[#3]{%
1995   \glxtrifcounttrigger{#2}%
1996   {%
1997     \cGlsformat{#2}{#3}%
1998     \gl sunset{#2}%

```

```

1999 }%
2000 {%
2001 \cGls@{#1}{#2}[#3]%
2002 }%
2003 }%

```

\@@cGlspl@

```

2004 \def\@@cGlspl@#1#2[#3]{%
2005 \glxtrifcounttrigger{#2}%
2006 {%
2007 \cGlsplformat{#2}{#3}%
2008 \glunset{#2}%
2009 }%
2010 {%
2011 \cGlspl@{#1}{#2}[#3]%
2012 }%
2013 }%

```

\@@cGLS@

```

2014 \def\@@cGLS@#1#2[#3]{%
2015 \glxtrifcounttrigger{#2}%
2016 {%
2017 \cGLSformat{#2}{#3}%
2018 \glunset{#2}%
2019 }%
2020 {%
2021 \cGLS@{#1}{#2}[#3]%
2022 }%
2023 }%

```

\@@cGLSpl@

```

2024 \def\@@cGLSpl@#1#2[#3]{%
2025 \glxtrifcounttrigger{#2}%
2026 {%
2027 \cGLSplformat{#2}{#3}%
2028 \glunset{#2}%
2029 }%
2030 {%
2031 \cGLSpl@{#1}{#2}[#3]%
2032 }%
2033 }%
2034 %
2035 % Remove default warnings from \cs{cgl} etc so that it can be used
2036 % interchangeably with \cs{gl} etc.
2037 %\begin{macro}\@cgl@
2038 % \begin{macrocode}
2039 \def\@cgl@#1#2[#3]{\cgl@{#1}{#2}[#3]}

```

\cGls@

2040 \def\@cGLs@#1#2[#3]{\@GLs@{#1}{#2}[#3]}

\@cglsp1@

2041 \def\@cglsp1@#1#2[#3]{\@glsp1@{#1}{#2}[#3]}

\@cGLsp1@

2042 \def\@cGLsp1@#1#2[#3]{\@GLsp1@{#1}{#2}[#3]}

Add all upper case versions not provided by glossaries.

\cGLS

2043 \newrobustcmd*{\cGLS}{\@glshyp@opt\@cGLS}

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

2044 \newcommand*{\@cGLS}[2][\%

2045 \new@ifnextchar[\@cGLS@{#1}{#2}]{\@cGLS@{#1}{#2}[]}%

2046 }

\@cGLS@

2047 \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.

2048 \newcommand*{\cGLSformat}[2]{\%

2049 \expandafter\mfirstucMakeUppercase\expandafter{\cglspformat{#1}{#2}}%

2050 }

\cGLSp1

2051 \newrobustcmd*{\cGLSp1}{\@glshyp@opt\@cGLSp1}

\@cGLSp1 Defined the un-starred form. Need to determine if there is a final optional argument

2052 \newcommand*{\@cGLSp1}[2][\%

2053 \new@ifnextchar[\@cGLSp1@{#1}{#2}]{\@cGLSp1@{#1}{#2}[]}%

2054 }

\@cGLSp1@

2055 \def\@cGLSp1@#1#2[#3]{\@GLSp1@{#1}{#2}[#3]}

\cGLSp1format Format used by \cGLSp1 if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

2056 \newcommand*{\cGLSp1format}[2]{\%

2057 \expandafter\mfirstucMakeUppercase\expandafter{\cglsp1format{#1}{#2}}%

2058 }

Modify the trigger formats to check for the regular attribute.

\cglformat

```
2059 \renewcommand*\cglformat}[2]{%
2060   \glsifregular{#1}
2061   {\glsentryfirst{#1}}%
2062   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2063 }
```

\cGlsformat

```
2064 \renewcommand*\cGlsformat}[2]{%
2065   \glsifregular{#1}
2066   {\Glsentryfirst{#1}}%
2067   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2068 }
```

\cglsplformat

```
2069 \renewcommand*\cglsplformat}[2]{%
2070   \glsifregular{#1}
2071   {\glsentryfirstplural{#1}}%
2072   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
2073 }
```

\cGlsplformat

```
2074 \renewcommand*\cGlsplformat}[2]{%
2075   \glsifregular{#1}
2076   {\Glsentryfirstplural{#1}}%
2077   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2078 }
```

New code similar to above for unit counting.

defunitcounters

```
2079 \newcommand*\@@newglossaryentry@defunitcounters{%
2080   \edef\@glo@countunit{\csuse{@glxtr@categoryattr@@\@glo@category @unitcount}}%
2081   \ifdefvoid\@glo@countunit
2082   {}%
2083   {%
2084     \@glxtr@ifunitcounter{\@glo@countunit}%
2085     {}%
2086     {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
2087   }%
2088 }
```

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

```
2089 \newcommand*\@glxtr@unitcountlist{}
```

@addunitcounter

```
2090 \newcommand*\@glxtr@addunitcounter}[1]{%
2091   \listadd{\@glxtr@unitcountlist}{#1}%

```

```

2092 \ifcsundef{glxstr@theunit@#1}
2093 {%
2094   \ifcsdef{theH#1}%
2095   {\csdef{glxstr@theunit@#1}{\csuse{theH#1}}}%
2096   {\csdef{glxstr@theunit@#1}{\csuse{the#1}}}%
2097 }%
2098 {}%
2099 }

```

r@ifunitcounter

```

2100 \newcommand*{\@glxstr@ifunitcounter}[3]{%
2101   \xifinlist{#1}{\@glxstr@unitcountlist}{#2}{#3}%
2102 }

```

urrentunitcount

```

2103 \newcommand*\@glxstr@currentunitcount[1]{%
2104   glo@\glsdetoklabel{#1}@currunit@\glsggetattribute{#1}{unitcount}.%
2105   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
2106 }

```

eviousunitcount

```

2107 \newcommand*\@glxstr@previousunitcount[1]{%
2108   glo@\glsdetoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
2109   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
2110 }

```

t@currunitcount

```

2111 \newcommand*{\@glx@increment@currunitcount}[1]{%
2112   \glshasattribute{#1}{unitcount}%
2113   {%
2114     \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
2115     \ifcsundef{\@glxstr@csname}%
2116     {%
2117       \csgdef{\@glxstr@csname}{1}%
2118       \listcsxadd
2119       {glo@\glsdetoklabel{#1}@unitlist}%
2120       {\glsggetattribute{#1}{unitcount}.%
2121        \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
2122     }%
2123   }%
2124   {%
2125     \csxdef{\@glxstr@csname}%
2126     {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
2127   }%
2128 }%
2129 {}%
2130 }

```

t@currunitcount


```

2131 \newcommand*{\@gls@local@increment@currunitcount}[1]{%
2132   \gls@hasattribute{#1}{unitcount}%
2133   {%
2134     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
2135     \ifcsundef{\@glsxtr@csname}%
2136     {%
2137       \csdef{\@glsxtr@csname}{1}%
2138       \listcseadd
2139         {glo@\glsdetoklabel{#1}@unitlist}%
2140         {\glsgetattribute{#1}{unitcount}.%
2141          \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
2142       }%
2143     }%
2144     {%
2145       \csedef{\@glsxtr@csname}%
2146         {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
2147       }%
2148     }%
2149   }%
2150 }

```

r@currunitcount

```

2151 \newcommand*{\@glsxtr@currunitcount}[2]{%
2152   \ifcsundef
2153     {glo@\glsdetoklabel{#1}@currunit@#2}%
2154     {0}%
2155     {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
2156 }%

```

r@prevunitcount

```

2157 \newcommand*{\@glsxtr@prevunitcount}[2]{%
2158   \ifcsundef
2159     {glo@\glsdetoklabel{#1}@prevunit@#2}%
2160     {0}%
2161     {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
2162 }%

```

eentryunitcount

```

2163 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
2164   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
2165   \renewcommand*{\gls@defdocnewglossaryentry}{%
2166     \renewcommand*\newglossaryentry[2]{%
2167       \PackageError{glossaries}{\string\newglossaryentry\space
2168         may only be used in the preamble when entry counting has
2169         been activated}{If you use \string\glsenableentryunitcount\space
2170         you must place all entry definitions in the preamble not in

```

```

2171     the document environment}%
2172 }%
2173 }%

```

New commands to access new fields:

```

2174 \newcommand*{\glstentrycurrcount}[1]{%
2175   \@glstxtr@currunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
2176   \csuse{glstxtr@theunit@\glstgetattribute{##1}{unitcount}}}%
2177 }%
2178 \newcommand*{\glstentryprevcount}[1]{%
2179   \@glstxtr@prevunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
2180   \csuse{glstxtr@theunit@\glstgetattribute{##1}{unitcount}}}%
2181 }%

```

Access total count:

```

2182 \newcommand*{\glstentryprevtotalcount}[1]{%
2183   \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
2184     {0}%
2185     {%
2186       \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
2187     }%
2188 }%

```

Access max value:

```

2189 \newcommand*{\glstentryprevmaxcount}[1]{%
2190   \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
2191     {0}%
2192     {%
2193       \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
2194     }%
2195 }%

```

Adjust post unset and reset:

```

2196 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
2197 \renewcommand*{\glstxtrpostunset}[1]{%
2198   \@glstxtr@entryunitcount@org@unset{##1}%
2199   \@glst@increment@currunitcount{##1}%
2200 }%
2201 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
2202 \renewcommand*{\glstxtrpostlocalunset}[1]{%
2203   \@glstxtr@entryunitcount@org@localunset{##1}%
2204   \@glst@local@increment@currunitcount{##1}%
2205 }%
2206 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
2207 \renewcommand*{\glstxtrpostreset}[1]{%
2208   \glsthasattribute{##1}{unitcount}%
2209   {%
2210     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
2211     \ifcsundef{\@glstxtr@csname}%
2212       {}%
2213       {\csgdef{\@glstxtr@csname}{0}}%

```

```

2214 }%
2215 {}%
2216 }%
2217 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
2218 \renewcommand*{\glxtrpostlocalreset}[1]{%
2219   \@glxtr@entryunitcount@org@localreset{##1}%
2220   \glshasattribute{##1}{unitcount}%
2221   {%
2222     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
2223     \ifcsundef{\@glxtr@csname}%
2224     {}%
2225     {\csdef{\@glxtr@csname}{0}}%
2226   }%
2227   {}%
2228 }%

```

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

```

2229 \let\@cgl@\@cgl@
2230 \let\@cgl@sp1\@cgl@sp1
2231 \let\@cGLS\@cGLS
2232 \let\@cGL@sp1\@cGL@sp1
2233 \let\@cGLS\@cGLS
2234 \let\@cGL@sp1\@cGL@sp1

```

Write information to the aux file.

```

2235 \AtEndDocument{\@gl@write@entryunitcounts}%
2236 \renewcommand*{\@gl@entry@unitcount}[3]{%
2237   \csgdef{glo@gl@detoklabel{##1}@prevunit@##3}{##2}%
2238   \ifcsundef{glo@gl@detoklabel{##1}@prevunittotal}%
2239   {\csgdef{glo@gl@detoklabel{##1}@prevunittotal}{##2}}%
2240   {%
2241     \csxdef{glo@gl@detoklabel{##1}@prevunittotal}{
2242       \number\numexpr\csuse{glo@gl@detoklabel{##1}@prevunittotal}+##2}%
2243     }%
2244     \ifcsundef{glo@gl@detoklabel{##1}@prevunitmax}%
2245     {\csgdef{glo@gl@detoklabel{##1}@prevunitmax}{##2}}%
2246     {%
2247       \ifnum\csuse{glo@gl@detoklabel{##1}@prevunitmax}<##2
2248       \csgdef{glo@gl@detoklabel{##1}@prevunitmax}{##2}%
2249       \fi
2250     }%
2251   }%
2252 \let\gl@enable@entryunitcount\relax
2253 \renewcommand*{\gl@enable@entrycount}{%
2254   \PackageError{glossaries-extra}{\string\gl@enable@entrycount\space
2255     can't be used with \string\gl@enable@entryunitcount}%
2256   {Use one or other but not both commands}%
2257 }%
2258 }

```

```

2259 \@onlypreamble\glsenableentryunitcount

entry@unitcount
2260 \newcommand*{\@gls@entry@unitcount}[3]{%

ryunitcounts@do
2261 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
2262   \immediate\write\@auxout
2263   {\string\@gls@entry@unitcount
2264     {\@glsentry}%
2265     {\@glsxtr@currunitcount{\@glsentry}{#1}%
2266     }%
2267     {#1}}%
2268 }

entryunitcounts
2269 \newcommand*{\@gls@write@entryunitcounts}{%
2270   \immediate\write\@auxout
2271   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
2272   \count@=0\relax
2273   \forallglsentries{\@glsentry}{%
2274     \glsattribute{\@glsentry}{unitcount}%
2275     {%
2276       \ifglsused{\@glsentry}%
2277       {%
2278         \forlistcsloop
2279           {\@gls@write@entryunitcounts@do}%
2280           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
2281       }%
2282     }%
2283     \advance\count@ by \@ne
2284   }%
2285 }%
2286 }%
2287 \ifnum\count@=0
2288   \GlossariesExtraWarningNoLine{Entry counting has been enabled
2289   \MessageBreak with \string\glsenableentryunitcount\space but the
2290   \MessageBreak attribute ‘unitcount’ hasn’t
2291   \MessageBreak been assigned to any of the defined
2292   \MessageBreak entries}%
2293 \fi
2294 }

```

entryUnitCounting 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.

```

2295 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

  Enable entry counting:

2296   \glsenableentryunitcount

```

Redefine \gls etc:

```
2297 \renewcommand*{\gls}{\cglss}%
2298 \renewcommand*{\Gls}{\cGls}%
2299 \renewcommand*{\glspl}{\cglsspl}%
2300 \renewcommand*{\Glspl}{\cGlspl}%
2301 \renewcommand*{\GLS}{\cGLS}%
2302 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
2303 \@glstr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
2304 \let\GlsXtrEnableEntryUnitCounting\@glstr@setentryunitcountunsetattr
2305 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
2306 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
2307 can't be used with \string\GlsXtrEnableEntryUnitCounting}%
2308 {Use one or other but not both commands}}%
2309 }
```

tcountunsetattr

```
2310 \newcommand*{\@glstr@setentryunitcountunsetattr}[3]{%
2311 \@for\@glstr@cat:=#1\do
2312 {%
2313 \ifdefempty{\@glstr@cat}{}%
2314 {%
2315 \glssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
2316 \glssetcategoryattribute{\@glstr@cat}{unitcount}{#3}%
2317 }%
2318 }%
2319 }
```

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.

nericNewAcronym

```
2320 \renewcommand*{\SetGenericNewAcronym}{%
2321 \let\@Gls@entryname\@Gls@acrenryname
2322 \renewcommand{\newacronym}[4][ ]{%
2323 \ifdefempty{\@glsacronymlists}%
2324 {%
2325 \def\@glo@type{\acronymtype}%
2326 \setkeys{glossentry}{##1}%
2327 \DeclareAcronymList{\@glo@type}%
```

```

2328 }%
2329 {}%
2330 \glskeylisttok{##1}%
2331 \glslabeltok{##2}%
2332 \glsshorttok{##3}%
2333 \glslongtok{##4}%
2334 \newacronymhook
2335 \protected@edef\@do@newglossaryentry{%
2336   \noexpand\newglossaryentry{\the\glslabeltok}%
2337   {%
2338     type=\acronymtype,%
2339     name={\expandonce{\acronymentry{##2}}},%
2340     sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
2341     text={\the\glsshorttok},%
2342     short={\the\glsshorttok},%
2343     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2344     long={\the\glslongtok},%
2345     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
2346     category=acronym,
2347     \GenericAcronymFields,%
2348     \the\glskeylisttok
2349   }%
2350 }%
2351 \@do@newglossaryentry
2352 }%
2353 \renewcommand*{\acrfullfmt}[3]{%
2354   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
2355 \renewcommand*{\Acrfullfmt}[3]{%
2356   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
2357 \renewcommand*{\ACRfullfmt}[3]{%
2358   \glslink[##1]{##2}{%
2359     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
2360 \renewcommand*{\acrfullplfmt}[3]{%
2361   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
2362 \renewcommand*{\Acrfullplfmt}[3]{%
2363   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
2364 \renewcommand*{\ACRfullplfmt}[3]{%
2365   \glslink[##1]{##2}{%
2366     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
2367 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
2368 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
2369 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
2370 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
2371 }

```

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:

```

2372 \let\@glxtr@org@setacronymstyle\setacronymstyle

```

```
2373 \let\@glxtr@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.

```
2374 \newcommand*\MakeAcronymsAbbreviations}{%
2375   \renewcommand*\newacronym}[4][]{%
2376     \newabbreviation[type=\acronymtype,category=acronym,##1]{##2}{##3}{##4}%
2377   }%
2378   \renewcommand*\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
2379   \renewcommand*\acronymfont}[1]{\glsabbrvfont{##1}}%
2380   \renewcommand*\setacronymstyle}[1]{%
2381     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
2382     unavailable.
2383     Use \string\setabbreviationstyle\space instead.
2384     The original acronym interface can be restored with
2385     \string\RestoreAcronyms}{}%
2386   }%
2387   \renewcommand*\newacronymstyle}[1]{%
2388     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
2389     available unless you restore the original acronym interface with
2390     \string\RestoreAcronyms}%
2391     \@glxtr@org@newacronymstyle{##1}%
2392   }%
2393 }
```

Switch acronyms to abbreviations:

```
2394 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```
2395 \newcommand*\RestoreAcronyms}{%
2396   \SetGenericNewAcronym
2397   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
2398   \renewcommand*\acronymfont}[1]{##1}%
2399   \let\setacronymstyle\@glxtr@org@setacronymstyle
2400   \let\newacronymstyle\@glxtr@org@newacronymstyle
```

Need to restore the original definition of \@gls@link@checkfirsthyper but \glxtrifwasfirstuse still needs setting for the benefit of the post-link hook.

```
2401 \renewcommand*\@gls@link@checkfirsthyper{%
2402   \ifglused{\glslabel}%
2403   {\let\glxtrifwasfirstuse\@secondoftwo}
2404   {\let\glxtrifwasfirstuse\@firstoftwo}%
2405   \@glxtr@org@checkfirsthyper
2406 }
2407 \glssetcategoryattribute{acronym}{regular}{false}%
2408 \setacronymstyle{long-short}%
2409 }
```

`\glsacspace` Allow the user to customise the maximum value.

```
2410 \renewcommand*{\glsacspace}[1]{%
2411   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
2412   \ifdim\dimen@<\glsacspacemax~\else\space\fi
2413 }
```

`\glsacspacemax` Value used in the above.

```
2414 \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`.

`reg@glosslist`

```
2415 \newcommand*{@glsxtr@reg@glosslist}{}

Save the original definition of \makeglossaries:
```

```
2416 \let@glsxtr@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`

```
2417 \renewcommand*{\makeglossaries}[1][]{%
2418   \ifblank{#1}%
2419   {@glsxtr@org@makeglossaries}%
2420   {%
2421     \edef@glsxtr@reg@glosslist{#1}%
2422     \ifundef{\glswrite}{\newwrite\glswrite}{}%
2423     \protected@write\@auxout{}{\string\providecommand
2424       \string@glsorder[1]}%
2425     \protected@write\@auxout{}{\string\providecommand
2426       \string@istfilename[1]}%
2427     \protected@write\@auxout{}{\string@istfilename{\istfilename}}%
2428     \protected@write\@auxout{}{\string@glsorder{\glsorder}}%
2429     \protected@write\@auxout{}{\string@glsxtr@makeglossaries{#1}}%
2430     \write\@auxout{\string\providecommand\string@gls@reference[3]}%

```

Iterate through each supplied glossary type and activate it.

```
2431   \@for@glo@type:=#1\do{%
2432     \ifdefempty{@glo@type}{\@makeglossary{@glo@type}}%
2433   }%
```


New glossaries must be created before \makeglossaries:

```
2434 \renewcommand*\newglossary[4] [] {%
2435 \PackageError{glossaries}{New glossaries
2436 must be created before \string\makeglossaries}{You need
2437 to move \string\makeglossaries\space after all your
2438 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
2439 \let\makeglossary\relax
2440 \let\makeglossary\relax
2441 \renewcommand\makeglossaries[1] [] {}%
```

Disable all commands that have no effect after \makeglossaries

```
2442 \@disable@onlypremakeg
```

Allow see key:

```
2443 \let\gls@checkseeallowed\relax
```

Adjust \@do@seeglossary

```
2444 \let\glsxtr@org@doseeglossary\@do@seeglossary
2445 \renewcommand*{\@do@seeglossary}[2] {%
2446 \edef\gls@label{\glsdetoklabel{##1}}%
2447 \edef\gls@type{\csname glo@\gls@label @type\endcsname}%
2448 \expandafter\DTLifinlist\expandafter{\gls@type}{\glsxtr@reg@glosslist}%
2449 {\glsxtr@org@doseeglossary{##1}{##2}}%
2450 {%
2451 \protected@write\@auxout{}{%
2452 \string\gls@reference
2453 {\gls@type}{\gls@label}{\string\glsseeformat##2{}}%
2454 }%
2455 }%
2456 }%
```

Adjust \@do@@@wrglossary

```
2457 \let\glsxtr@@do@@wrglossary\@@do@@wrglossary
2458 \def\@@do@@wrglossary{%
2459 \edef\gls@type{\csname glo@\gls@label @type\endcsname}%
2460 \expandafter\DTLifinlist\expandafter{\gls@type}{\glsxtr@reg@glosslist}%
2461 {\glsxtr@@do@@wrglossary}%
2462 {\gls@noidxglossary}%
2463 }%
```

Suppress warning about no \makeglossaries

```
2464 \let\warn@nomakeglossaries\relax
2465 \def\warn@noprntglossary{%
2466 \GlossariesWarningNoLine{No \string\prntglossary\space
2467 or \string\prntglossaries\space
2468 found.^^J(Remove \string\makeglossaries\space if you don't want
2469 any glossaries.)^^JThis document will not have a glossary}%
2470 }%
```

Only warn for glossaries not listed.

```

2471 \renewcommand{\@gls@noref@warn}[1]{%
2472 \edef\@gls@type{##1}%
2473 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
2474 {%
2475 \GlossariesExtraWarning{Can't use
2476 \string\printnoidxglossary[type={\@gls@type}]
2477 when '\@gls@type' is listed in the optional argument of
2478 \string\makeglossaries}%
2479 }%
2480 {%
2481 \GlossariesWarning{Empty glossary for
2482 \string\printnoidxglossary[type={##1}].
2483 Rerun may be required (or you may have forgotten to use
2484 commands like \string\gls)}}%
2485 }%
2486 }%

```

Adjust display number list to check for type:

```

2487 \renewcommand*\@glsdisplaynumberlist}[1]{%
2488 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2489 {\@glsxtr@idx@displaynumberlist{##1}}%
2490 {\@glsxtr@noidx@displaynumberlist{##1}}%
2491 }%

```

Adjust entry list:

```

2492 \renewcommand*\@glsentrynumberlist}[1]{%
2493 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2494 {\@glsxtr@idx@entrynumberlist{##1}}%
2495 {\@glsxtr@noidx@entrynumberlist{##1}}%
2496 }%

```

Adjust number list loop

```

2497 \renewcommand*\@glsnumberlistloop}[2]{%
2498 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
2499 {%
2500 \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
2501 not available for glossary '##1'}{%
2502 }%
2503 {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
2504 }%

```

Only sanitize sort for normal indexing glossaries.

```

2505 \renewcommand*\@glsprestandardsort}[3]{%
2506 \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
2507 {%
2508 \glsdosanitizesort
2509 }%
2510 {%
2511 \ifglssanitizesort
2512 \@gls@noidx@sanitizesort

```

```

2513     \else
2514     \@gls@noidx@nosanitizesort
2515     \fi
2516 }%
2517 }%

```

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

```

2518 \renewcommand*{\new@glossaryentry}[2]{%
2519   \PackageError{glossaries-extra}{Glossary entries must be defined
2520     in the preamble\MessageBreak when you use the optional argument
2521     of \string\makeglossaries}{Either move your definitions to the
2522     preamble or don't use the optional argument of
2523     \string\makeglossaries}%
2524 }%

```

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).

```

2525 \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
2526 \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

2527   \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
2528   type=\glsdefaulttype,\@end@glsxtr@gettype
2529   \def\@glo@sorttype{\@glo@default@sorttype}%
2530 }%

```

Check automake setting:

```

2531 \ifglsautomake
2532   \renewcommand*{\@gls@doautomake}{%
2533     \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
2534       \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
2535     }%
2536   }%
2537 \fi
2538 }%
2539 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`rgprintglossary` Save original definition (also needed for the on-the-fly macro).

```

2540 \let\@glsxtr@orgprintglossary\@printglossary

```

`\@printglossary` Redefine.

```

2541 \renewcommand{\@printglossary}[2]{%
2542   \def\@glsxtr@printglossopts{#1}%
2543   \@glsxtr@orgprintglossary{#1}{#2}%
2544 }

```

@makeglossaries For the benefit of makeglossaries

```
2545 \newcommand*{\glxtr@makeglossaries}[1]{}
```

@glxtr@gettype Get just the type.

```
2546 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
2547   \def\@glo@type{#2}%
2548 }
```

@assign@sortkey Assign the sort key.

```
2549 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
2550   \edef\@glo@type{\@glo@type}%
2551   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
2552   {%
2553     \@glo@no@assign@sortkey{#1}%
2554   }%
2555   {%
2556     \@glo@assign@sortkey{#1}%
2557   }%
2558 }%
```

Display number list for the regular version:

splaynumberlist

```
2559 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
2560 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
2561   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
2562   \ifdef\@gls@loclist
2563   {%
2564     \def\@gls@noidxloclist@sep{%
2565       \def\@gls@noidxloclist@sep{%
2566         \def\@gls@noidxloclist@sep{%
2567           \glsnumlistsep
2568         }%
2569         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
2570       }%
2571     }%
2572     \def\@gls@noidxloclist@finalsep{}%
2573     \def\@gls@noidxloclist@prev{}%
2574     \forlistloop{\@glsnoidxdisplayloclisthandler}{\@gls@loclist}%
2575     \@gls@noidxloclist@finalsep
2576     \@gls@noidxloclist@prev
2577   }%
2578   {%
2579     ??\glsdoifexists{#1}%
2580   }%
```

```

2581      \GlossariesWarning{Missing location list for ‘#1’. Either
2582      a rerun is required or you haven’t referenced the entry.}%
2583    }%
2584  }%
2585}%
2586

```

And for the number list loop:

@numberlistloop

```

2587 \newcommand*{\@glxtr@noidx@numberlistloop}[3]{%
2588   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2589   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
2590   \let\@gls@org@glssseeformat\glssseeformat
2591   \let\glsnoidxdisplayloc#2\relax
2592   \let\glssseeformat#3\relax
2593   \ifdef\@gls@loclist
2594   {%
2595     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
2596   }%
2597   {%
2598     ??\glsdoifexists{#1}%
2599     {%
2600       \GlossariesWarning{Missing location list for ‘##1’. Either
2601       a rerun is required or you haven’t referenced the entry.}%
2602     }%
2603   }%
2604   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
2605   \let\glssseeformat\@gls@org@glssseeformat
2606}%

```

Same for entry number list.

entrynumberlist

```

2607 \newcommand*{\@glxtr@noidx@entrynumberlist}[1]{%
2608   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
2609   \ifdef\@gls@loclist
2610   {%
2611     \glsnoidxloclist{\@gls@loclist}%
2612   }%
2613   {%
2614     ??\glsdoifexists{#1}%
2615     {%
2616       \GlossariesWarning{Missing location list for ‘#1’. Either
2617       a rerun is required or you haven’t referenced the entry.}%
2618     }%
2619   }%
2620}%

```

entrynumberlist

```

2621 \newcommand*{\@glxtr@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

```

2622 \renewcommand{\@print@glossary}{%
2623   \makeatletter
2624   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2625   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
2626   {}%
2627   {\glstrNoGlossaryWarning{\@glo@type}}%
2628   \ifglxindy
2629     \ifcsundef{@xdy@\@glo@type @language}%
2630     {%
2631       \edef\@do@auxoutstuff{%
2632         \noexpand\AtEndDocument{%
2633           \noexpand\immediate\noexpand\write\@auxout{%
2634             \string\providecommand\string\@xdylanguage[2]{}}%
2635           \noexpand\immediate\noexpand\write\@auxout{%
2636             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
2637         }%
2638       }%
2639     }%
2640     {%
2641       \edef\@do@auxoutstuff{%
2642         \noexpand\AtEndDocument{%
2643           \noexpand\immediate\noexpand\write\@auxout{%
2644             \string\providecommand\string\@xdylanguage[2]{}}%
2645           \noexpand\immediate\noexpand\write\@auxout{%
2646             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
2647               @language\endcsname}}%
2648         }%
2649       }%
2650     }%
2651     \@do@auxoutstuff
2652     \edef\@do@auxoutstuff{%
2653       \noexpand\AtEndDocument{%
2654         \noexpand\immediate\noexpand\write\@auxout{%
2655           \string\providecommand\string\@gls@codepage[2]{}}%
2656         \noexpand\immediate\noexpand\write\@auxout{%
2657           \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
2658       }%
2659     }%
2660     \@do@auxoutstuff
2661   \fi
2662   \renewcommand*{\@warn@nomakeglossaries}{%
2663     \GlossariesWarningNoLine{\string\makeglossaries\space
2664       hasn't been used,^^Jthe glossaries will not be updated}%
2665   }%
2666 }
```

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

`oGlsWarningHead` Header message.

```
2667 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
2668   This document is incomplete. The external file associated with
2669   the glossary ‘#1’ (which should be called \texttt{#2})
2670   hasn’t been created.%
2671 }
```

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

```
2672 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
2673   This has probably happened because there are no entries defined
2674   in this glossary.%
2675 }
```

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

```
2676 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
2677   If you don’t want this glossary,
2678   add \texttt{nomain} to your package option list when you load
2679   \texttt{glossaries-extra.sty}. For example:%
2680 }
```

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

```
2681 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
2682   Did you forget to use \texttt{type=#1} when you defined your
2683   entries? If you tried to load entries into this glossary with
2684   \texttt{\string\loadglsentries} did you remember to use
2685   \texttt{[#1]} as the optional argument? If you did, check that
2686   the definitions in the file you loaded all had the type set
2687   to \texttt{\string\glsdefaulttype}.%
2688 }
```

`arningCheckFile` Advisory message to check the file contents.

```
2689 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
2690   Check the contents of the file \texttt{#1}. If
2691   it’s empty, that means you haven’t indexed any of your entries in this
2692   glossary (using commands like \texttt{\string\gls} or
2693   \texttt{\string\glsadd}) so this list can’t be generated.
2694   If the file isn’t empty, the document build process hasn’t been
2695   completed.%
2696 }
```

`WarningAutoMake` Message when automake option has been used.

```
2697 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
2698   You may need to rerun \LaTeX. If you already have, it may be that
2699   \TeX’s shell escape doesn’t allow you to run
2700   \ifglxindy xindy\else makeindex\fi. Check the
2701   transcript file \texttt{\jobname.log}. If the shell escape is
```

```

2702 disabled, try one of the following:
2703
2704 \begin{itemize}
2705   \item Run the external (Lua) application:
2706
2707       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2708
2709   \item Run the external (Perl) application:
2710
2711       \texttt{makeglossaries \string"\jobname\string"}
2712 \end{itemize}
2713
2714 Then rerun \LaTeX\ on this document.
2715 \GlossariesExtraWarning{Rerun required to build the
2716 glossary ‘#1’ or check TeX’s shell escape allows
2717 you to run \ifglxindy xindy\else makeindex\fi}%
2718 }

```

WarningMisMatch Mismatching \makenoidxglossaries.

```

2719 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
2720 You need to either replace \texttt{\string\makenoidxglossaries}
2721 with \texttt{\string\makeglossaries} or replace
2722 \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
2723 \texttt{\string\printnoidxglossary}
2724 (or \texttt{\string\printnoidxglossaries}) and then rebuild
2725 this document.%
2726 }

```

arningBuildInfo Build advice.

```

2727 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
2728 Try one of the following:
2729 \begin{itemize}
2730   \item Add \texttt{automake} to your package option list when you load
2731       \texttt{glossaries-extra.sty}. For example:
2732
2733       \texttt{\string\usepackage[automake]%
2734               \glsgluebrace glossaries-extra\glsgluebrace}
2735
2736   \item Run the external (Lua) application:
2737
2738       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
2739
2740   \item Run the external (Perl) application:
2741
2742       \texttt{makeglossaries \string"\jobname\string"}
2743 \end{itemize}
2744
2745 Then rerun \LaTeX\ on this document.%
2746 }

```


oGlsWarningTail Final paragraph.

```
2747 \newcommand{\GlsXtrNoGlsWarningTail}{%
2748 This message will be removed once the problem has been fixed.%
2749 }
```

GlsWarningNoOut No out file created. Build advice.

```
2750 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
2751 The file \texttt{#1} doesn't exist. This most likely means you haven't used
2752 \texttt{\string\makeglossaries} or you have used
2753 \texttt{\string\nofiles}. If this is just a draft version of the
2754 document, you can suppress this message using the
2755 \texttt{nomissingglstext} package option.%
2756 }
```

glossarywarning

```
2757 \newcommand*{@\glstr@defaultnoglossarywarning}[1]{%
2758 \glossarysection[\glossarytoctitle]{\glossarytitle}
2759 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
2760 \par
2761 \glstrifemptyglossary{#1}%
2762 {%
2763 \GlsXtrNoGlsWarningEmptyStart\space
2764 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
2765 \medskip
2766 \noindent\texttt{\string\usepackage[nomain\ifglssacronym ,acronym\fi]%
2767 \glsoopenbrace glossaries-extra\glsclosebrace}
2768 \medskip
2769 }%
2770 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
2771 }%
2772 {%
2773 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
2774 {%
2775 \GlsXtrNoGlsWarningCheckFile
2776 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
2777
2778 \ifglssautomake
2779
2780 \GlsXtrNoGlsWarningAutoMake{#1}
2781
2782 \else
2783
2784 \ifthenelse{\equal{#1}{main}}%
2785 {%
2786 \GlsXtrNoGlsWarningEmptyMain\par
2787 \medskip
2788 \noindent\texttt{\string\usepackage[nomain]%
2789 \glsoopenbrace glossaries-extra\glsclosebrace}
2790 \medskip
```

```

2791     }%
2792     {}%
2793
2794     \ifdefequal\makeglossaries\@no@makeglossaries
2795     {%
2796         \GlsXtrNoGlsWarningMisMatch
2797     }%
2798     {%
2799         \GlsXtrNoGlsWarningBuildInfo
2800     }%
2801     \fi
2802 }%
2803 {%
2804     \GlsXtrNoGlsWarningNoOut
2805     {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
2806 }%
2807 }%
2808 \par
2809 \GlsXtrNoGlsWarningTail
2810 }

```

Provide some commands to accompany the record option.

xtresourcefile This is provided for the benefit of any external helper application.

```

2811 \newcommand*{\glxtrresourcefile}[2] [] {%
2812     \protected@write\@auxout{}\string\glxtr@resource{#1}{#2}}%
2813     \InputIfFileExists{#2}{}%
2814     {%
2815         \GlossariesExtraWarning{No file ‘#2’}%
2816     }%
2817 }
2818 \@onlypreamble\glxtrresourcefile

```

glxtr@resource

```

2819 \newcommand*{\glxtr@resource}[2] {}

```

ntunsrtglossary Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

2820 \newcommand*{\printunsrtglossary}[1] [type=\glsdefaulttype] {%
2821     \@printglossary{#1}{\@print@unsrt@glossary}%
2822 }

```

unsrtglossaries Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```

2823 \newcommand*{\printunsrtglossaries}{%
2824     \foralllglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
2825 }

```

@unsrt@glossary

```
2826 \newcommand*{\@print@unsrt@glossary}{%
2827   \glossarysection[\glossarytoctitle]{\glossarytitle}%
2828   \glossarypreamble
      check for empty list
2829   \ifcempty{glolist@\@glo@type}
2830   {%
2831     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
2832   }%
2833   {%
2834     \begin{theglossary}%
2835     \glossaryheader
2836     \glsresetentrylist
2837     \def\@gls@currentlettergroup{%
2838       \expandafter\@for\expandafter\glscurrententrylabel\expandafter
2839       : \expandafter=\csname glolist@\@glo@type\endcsname\do{%
2840         \ifdefempty{\glscurrententrylabel}
2841         {%
2842           {\@gls@noidx\do\glscurrententrylabel}%
2843         }%
2844       \end{theglossary}%
2845     }%
2846     \glossarypostamble
2847 }
```

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.

```
2848 \@ifpackageloaded{glossaries-accsupp}
2849 {
```

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

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

```
2850   \newcommand*{\glsaccessname}[1]{%
2851     \glsnameaccessdisplay
2852     {%
2853       \glsentryname{#1}%
2854     }%
2855     {#1}%
2856   }
```

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

```

2857 \newcommand*{\Glsaccessname}[1]{%
2858   \glsnameaccessdisplay
2859   {%
2860     \Glsentryname{#1}%
2861   }%
2862   {#1}%
2863 }

```

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

```

2864 \newcommand*{\Glsaccessname}[1]{%
2865   \glsnameaccessdisplay
2866   {%
2867     \mfirstucMakeUppercase{\Glsentryname{#1}}%
2868   }%
2869   {#1}%
2870 }

```

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

```

2871 \newcommand*{\glsaccessesstext}[1]{%
2872   \glstextaccessdisplay
2873   {%
2874     \glsentrytext{#1}%
2875   }%
2876   {#1}%
2877 }

```

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

```

2878 \newcommand*{\Glsaccessesstext}[1]{%
2879   \glstextaccessdisplay
2880   {%
2881     \Glsentrytext{#1}%
2882   }%
2883   {#1}%
2884 }

```

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

```

2885 \newcommand*{\GLSaccessesstext}[1]{%
2886   \glstextaccessdisplay
2887   {%
2888     \mfirstucMakeUppercase{\glsentrytext{#1}}%
2889   }%
2890   {#1}%
2891 }

```

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

```

2892 \newcommand*{\glsaccessplural}[1]{%
2893   \glspluralaccessdisplay

```

```

2894   {%
2895       \glstryplural{#1}%
2896   }%
2897   {#1}%
2898   }

```

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

```

2899   \newcommand*\Glsaccessplural}[1]{%
2900       \glsppluralaccessdisplay
2901   {%
2902       \glstryplural{#1}%
2903   }%
2904   {#1}%
2905   }

```

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

```

2906   \newcommand*\GLSaccessplural}[1]{%
2907       \glsppluralaccessdisplay
2908   {%
2909       \mfirstucMakeUppercase{\glstryplural{#1}}%
2910   }%
2911   {#1}%
2912   }

```

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

```

2913   \newcommand*\glaccessfirst}[1]{%
2914       \glfirstaccessdisplay
2915   {%
2916       \glstryfirst{#1}%
2917   }%
2918   {#1}%
2919   }

```

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

```

2920   \newcommand*\GLsaccessfirst}[1]{%
2921       \glfirstaccessdisplay
2922   {%
2923       \GLstryfirst{#1}%
2924   }%
2925   {#1}%
2926   }

```

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

```

2927   \newcommand*\GLSaccessfirst}[1]{%
2928       \glfirstaccessdisplay
2929   {%

```

```

2930     \mfirstucMakeUppercase{\glentryfirst{#1}}%
2931   }%
2932   {#1}%
2933 }

```

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

```

2934 \newcommand*{\glsaccessfirstplural}[1]{%
2935   \glsfirstpluralaccessdisplay
2936   {%
2937     \glentryfirstplural{#1}%
2938   }%
2939   {#1}%
2940 }

```

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

```

2941 \newcommand*{\Glsaccessfirstplural}[1]{%
2942   \glsfirstpluralaccessdisplay
2943   {%
2944     \Glentryfirstplural{#1}%
2945   }%
2946   {#1}%
2947 }

```

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

```

2948 \newcommand*{\GLSaccessfirstplural}[1]{%
2949   \glsfirstpluralaccessdisplay
2950   {%
2951     \mfirstucMakeUppercase{\glentryfirstplural{#1}}%
2952   }%
2953   {#1}%
2954 }

```

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

```

2955 \newcommand*{\glsaccesssymbol}[1]{%
2956   \glssymbolaccessdisplay
2957   {%
2958     \glentrysymbol{#1}%
2959   }%
2960   {#1}%
2961 }

```

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

```

2962 \newcommand*{\Glsaccesssymbol}[1]{%
2963   \glssymbolaccessdisplay
2964   {%
2965     \Glentrysymbol{#1}%

```

```

2966     }%
2967     {#1}%
2968 }

```

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

```

2969 \newcommand*{\GLSaccesssymbol}[1]{%
2970   \glssymbolaccessdisplay
2971   {%
2972     \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
2973   }%
2974   {#1}%
2975 }

```

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

```

2976 \newcommand*{\GLSaccesssymbolplural}[1]{%
2977   \glssymbolpluralaccessdisplay
2978   {%
2979     \glsentrysymbolplural{#1}%
2980   }%
2981   {#1}%
2982 }

```

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

```

2983 \newcommand*{\GLSaccesssymbolplural}[1]{%
2984   \glssymbolpluralaccessdisplay
2985   {%
2986     \GLSentrysymbolplural{#1}%
2987   }%
2988   {#1}%
2989 }

```

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

```

2990 \newcommand*{\GLSaccesssymbolplural}[1]{%
2991   \glssymbolpluralaccessdisplay
2992   {%
2993     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
2994   }%
2995   {#1}%
2996 }

```

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

```

2997 \newcommand*{\glSaccessdesc}[1]{%
2998   \glSdescriptionaccessdisplay
2999   {%
3000     \glSentrydesc{#1}%
3001   }%
3002   {#1}%
3003 }

```

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

```

3004 \newcommand*\Glsaccessdesc}[1]{%
3005 \glsdescriptionaccessdisplay
3006 {%
3007 \Glsentrydesc{#1}%
3008 }%
3009 {#1}%
3010 }

```

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

```

3011 \newcommand*\GLSaccessdesc}[1]{%
3012 \glsdescriptionaccessdisplay
3013 {%
3014 \mfirstucMakeUppercase{\glsentrydesc{#1}}%
3015 }%
3016 {#1}%
3017 }

```

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

```

3018 \newcommand*\glsaccessdescplural}[1]{%
3019 \glsdescriptionpluralaccessdisplay
3020 {%
3021 \glsentrydescplural{#1}%
3022 }%
3023 {#1}%
3024 }

```

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

```

3025 \newcommand*\Glsaccessdescplural}[1]{%
3026 \glsdescriptionpluralaccessdisplay
3027 {%
3028 \Glsentrydescplural{#1}%
3029 }%
3030 {#1}%
3031 }

```

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

```

3032 \newcommand*\GLSaccessdescplural}[1]{%
3033 \glsdescriptionpluralaccessdisplay
3034 {%
3035 \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
3036 }%
3037 {#1}%
3038 }

```

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


```

3039 \newcommand*{\glsaccesssshort}[1]{%
3040   \glsshortaccessdisplay
3041   {%
3042     \glsentryshort{#1}%
3043   }%
3044   {#1}%
3045 }

```

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

```

3046 \newcommand*{\Glsaccesssshort}[1]{%
3047   \glsshortaccessdisplay
3048   {%
3049     \Glsentryshort{#1}%
3050   }%
3051   {#1}%
3052 }

```

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

```

3053 \newcommand*{\GLSaccesssshort}[1]{%
3054   \glsshortaccessdisplay
3055   {%
3056     \mfirstucMakeUppercase{\glsentryshort{#1}}%
3057   }%
3058   {#1}%
3059 }

```

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

```

3060 \newcommand*{\glsaccesssshortpl}[1]{%
3061   \glsshortpluralaccessdisplay
3062   {%
3063     \glsentryshortpl{#1}%
3064   }%
3065   {#1}%
3066 }

```

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

```

3067 \newcommand*{\Glsaccesssshortpl}[1]{%
3068   \glsshortpluralaccessdisplay
3069   {%
3070     \Glsentryshortpl{#1}%
3071   }%
3072   {#1}%
3073 }

```

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

```

3074 \newcommand*{\GLSaccesssshortpl}[1]{%

```

```

3075     \glsshortpluralaccessdisplay
3076     {%
3077         \mfirstucMakeUppercase{\glsentryshortpl{#1}}%
3078     }%
3079     {#1}%
3080 }

```

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

```

3081     \newcommand*{\glsaccesslong}[1]{%
3082         \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
3083     }

```

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

```

3084
3085     \newcommand*{\Glsaccesslong}[1]{%
3086         \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
3087     }

```

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

```

3088     \newcommand*{\GLSaccesslong}[1]{%
3089         \glslongaccessdisplay
3090         {%
3091             \mfirstucMakeUppercase{\glsentrylong{#1}}%
3092         }%
3093         {#1}%
3094     }

```

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

```

3095     \newcommand*{\glsaccesslongpl}[1]{%
3096         \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
3097     }

```

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

```

3098
3099     \newcommand*{\Glsaccesslongpl}[1]{%
3100         \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
3101     }

```

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

```

3102     \newcommand*{\GLSaccesslongpl}[1]{%
3103         \glslongpluralaccessdisplay
3104         {%
3105             \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
3106         }%
3107         {#1}%
3108     }

```

End of if part

```

3109 }
3110 {

```

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

`\glsaccessname` Display the name value (no link and no check for existence).
3111 `\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.
3112 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.
3113 `\newcommand*{\GLSaccessname}[1]{%`
3114 `\protect\mfirstucMakeUppercase{\glsentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).
3115 `\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.
3116 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
3117 `\newcommand*{\GLSaccesstext}[1]{%`
3118 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
3119 `\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.
3120 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
3121 `\newcommand*{\GLSaccessplural}[1]{%`
3122 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
3123 `\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.
3124 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
3125 `\newcommand*{\GLSaccessfirst}[1]{%`
3126 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
3127 `\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.

3128 \newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}

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

3129 \newcommand*{\GLSaccessfirstplural}[1]{%

3130 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}

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

3131 \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.

3132 \newcommand*{\GLsaccesssymbol}[1]{\GLsentrysymbol{#1}}

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

3133 \newcommand*{\GLSaccesssymbol}[1]{%

3134 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}

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

3135 \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.

3136 \newcommand*{\GLsaccesssymbolplural}[1]{\GLsentrysymbolplural{#1}}

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

3137 \newcommand*{\GLSaccesssymbolplural}[1]{%

3138 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}

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

3139 \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.

3140 \newcommand*{\GLsaccessdesc}[1]{\GLsentrydesc{#1}}

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

3141 \newcommand*{\GLSaccessdesc}[1]{%

3142 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}

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

3143 \newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}

ccessdesclural Display the desclural value (no link and no check for existence) with the first letter converted to upper case.
3144 \newcommand*{\GLsaccessdesclural}[1]{\Glsentrydesclural{#1}}

ccessdesclural Display the desclural value (no link and no check for existence). converted to upper case.
3145 \newcommand*{\GLSaccessdesclural}[1]{%
3146 \protect\mfirstucMakeUppercase{\glsentrydesclural{#1}}}

\glsaccessshort Display the short form (no link and no check for existence).
3147 \newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}

\GLsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).
3148 \newcommand*{\GLsaccessshort}[1]{\Glsentryshort{#1}}

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.
3149 \newcommand*{\GLSaccessshort}[1]{%
3150 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}

lsaccessshortpl Display the short plural form (no link and no check for existence).
3151 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).
3152 \newcommand*{\GLsaccessshortpl}[1]{\Glsentryshortpl{#1}}

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.
3153 \newcommand*{\GLSaccessshortpl}[1]{%
3154 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}

\glsaccesslong Display the long form (no link and no check for existence).
3155 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}

\GLsaccesslong Display the long form (no link and no check for existence).
3156 \newcommand*{\GLsaccesslong}[1]{\Glsentrylong{#1}}

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.
3157 \newcommand*{\GLSaccesslong}[1]{%
3158 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}

glsaccesslongpl Display the long plural form (no link and no check for existence).
3159 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}

GLsaccesslongpl Display the long plural form (no link and no check for existence).
3160 \newcommand*{\GLsaccesslongpl}[1]{\Glsentrylongpl{#1}}

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

```
3161 \newcommand*{\GLSaccesslongpl}[1]{%
3162 \protect\mfirstucMakeUppercase{\gl Sentrylongpl{#1}}}
```

End of else part

```
3163 }
```

1.5 Categories

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

```
3164 \gl saddstoragekey{category}{general}{\gl scategory}
```

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

```
3165 \newcommand{\gl sifcategory}[4]{%
3166 \ifgl sfieldeq{#1}{category}{#2}{#3}{#4}%
3167 }
```

Categories can have attributes.

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

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

```
3168 \newcommand*{\gl ssetcategoryattribute}[3]{%
3169 \csdef{@gl sxtr@categoryattr@@#1@#2}{#3}%
3170 }
```

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

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

```
3171 \newcommand*{\gl sgetcategoryattribute}[2]{%
3172 \csuse{@gl sxtr@categoryattr@@#1@#2}%
3173 }
```

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

Tests if the category has the given attribute set.

```
3174 \newcommand*{\gl shascategoryattribute}[4]{%
3175 \ifcvoid{@gl sxtr@categoryattr@@#1@#2}{#4}{#3}%
3176 }
```

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

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

```
3177 \newcommand*{\glsssetattribute}[3]{%
3178   \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
3179 }
```

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

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

```
3180 \newcommand*{\glssgetattribute}[2]{%
3181   \glssgetcategoryattribute{\glscategory{#1}}{#2}%
3182 }
```

`\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.

```
3183 \newcommand*{\glsshasattribute}[4]{%
3184   \ifglssentryexists{#1}%
3185   {\glsshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
3186   {#4}%
3187 }
```

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

True if category has the attribute with the given value.

```
3188 \newcommand{\glssifcategoryattribute}[5]{%
3189   \ifcsundef{@glssxtr@categoryattr@#1@#2}%
3190   {#5}%
3191   {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
3192 }
```

`\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.

```
3193 \newcommand{\glsifattribute}[5]{%
3194   \ifglstryexists{#1}%
3195   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
3196   {#5}%
3197 }
```

Set attributes for the default general category:

```
3198 \glssetcategoryattribute{general}{regular}{true}
```

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

```
3199 \glssetcategoryattribute{acronym}{regular}{true}
```

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

```
3200 \newcommand*{\glssetregularcategory}[1]{%
3201   \glssetcategoryattribute{#1}{regular}{true}%
3202 }
```

`ifregularcategory`

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

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

```
3203 \newcommand{\glsifregularcategory}[3]{%
3204   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
3205 }
```

`notregularcategory`

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

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

```
3206 \newcommand{\glsifnotregularcategory}[3]{%
3207   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
3208 }
```

`\glsifregular`

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

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

```
3209 \newcommand{\glsifregular}[3]{%
3210   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
3211 }
```


\glsifnotregular

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

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

```
3212 \newcommand{\glsifnotregular}[3]{%
3213   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
3214 }
```

\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.

```
3215 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
3216   \forallglossaries[#1]{#3}%
3217   {%
3218     \forlgsentries[#3]{#4}%
3219     {%
3220       \glsifcategory{#4}{#2}{#5}{}%
3221     }%
3222   }%
3223 }
```

\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.

```
3224 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
3225   \forallglossaries[#1]{#4}%
3226   {%
3227     \forlgsentries[#4]{#5}%
3228     {%
3229       \glsifattribute{#5}{#2}{#3}{#6}{}%
3230     }%
3231   }%
3232 }
```

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

```
3233 \ifdef\newterm
3234 {%
```

`\newterm`

```
3235 \renewcommand*\newterm}[2] [] {%
3236 \newglossaryentry{#2}%
3237 {type={index},category=index,name={#2},%
3238 description={\glstrpostdescription\nopostdesc},#1}%
3239 }
```

Indexed terms are regular by default.

```
3240 \glsssetcategoryattribute{index}{regular}{true}
```

`trpostdescindex`

```
3241 \newcommand*\glstrpostdescindex{}
3242 }
3243 {}
```

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`.

```
3244 \ifdef\printsymbols
3245 {%
```

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

```
3246 \newcommand*\glstrnewsymbol}[3] [] {%
3247 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
3248 }
```

Symbols are regular by default.

```
3249 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
3250 \newcommand*\glstrpostdescsymbol{}
3251 }
3252 {}
```

Similar for the `numbers` option.

```
3253 \ifdef\printnumbers
3254 {%
```

glsxtrnewnumber

```
3255 \ifdef\printnumbers
3256   \newcommand*{\glsxtrnewnumber}[3][]{%
3257     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
3258   }
```

Numbers are regular by default.

```
3259   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
3260   \newcommand*{\glsxtrpostdescnumber}{}

3261 }
3262 {}
```

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

```
3263 \newcommand*{\glsxtrsetcategory}[2]{%
3264   \@for\@glsxtr@label:=#1\do
3265   {%
3266     \glsfieldxdef{\@glsxtr@label}{category}{#2}%
3267   }%
3268 }
```

categoryforall 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.

```
3269 \newcommand*{\glsxtrsetcategoryforall}[2]{%
3270   \forallglossaries[#1]{\@glsxtr@type}{%
3271     \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
3272     {%
3273       \glsfieldxdef{\@glsxtr@label}{category}{#2}%
3274     }%
3275   }%
3276 }
```

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

Apply title casing to the contents of the given field.

```
3277 \newcommand*{\glsxtrfieldtitlecase}[2]{%
3278   \expandafter\glsxtrfieldtitlecasesecs\expandafter
3279   {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
3280 }
```

fieldtitlecasesecs The command used by `\glsxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
3281 \newcommand*{\glsxtrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}
```

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.

```

3282 \ifpackageloaded{glossaries-accsupp}
3283 {
3284   \renewcommand*{\glossentrydesc}[1]{%
3285     \glsdoifexistsorwarn{#1}%
3286     {%
3287       \glssetabbrvfmt{\glscategory{#1}}%
3288       \glsattribute{#1}{glossdescfont}%
3289       {%
3290         \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
3291         \ifcsdef{\@glxtr@attrval}%
3292         {%
3293           \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
3294         }%
3295         {%
3296           \GlossariesExtraWarning{Unknown control sequence name
3297             '\@glxtr@attrval' supplied in glossdescfont attribute
3298             for entry '#1'. Ignoring}%
3299           \let\@glxtr@glossdescfont\@firstofone
3300         }%
3301       }%
3302       {\let\@glxtr@glossdescfont\@firstofone}%
3303       \glsifattribute{#1}{glossdesc}{firstuc}%
3304       {%
3305         \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
3306       }%
3307       {%
3308         \glsifattribute{#1}{glossdesc}{title}%
3309         {%
3310           \@glxtr@do@titlecaps@warn
3311           \glsdescriptionaccessdisplay
3312           {%
3313             \@glxtr@glossdescfont{\@glxtr@fieldtitlecase{#1}{desc}}%
3314           }%
3315           {#1}%
3316         }%
3317         {%
3318           \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
3319         }%
3320       }%
3321     }%
3322   }
3323 }
3324 {

```

```

3325 \renewcommand*{\glossentrydesc}[1]{%
3326   \glsdoifexistsorwarn{#1}%
3327   {%
3328     \glssetabbrvfmt{\glscategory{#1}}%
3329     \glshasattribute{#1}{glossdescfont}%
3330     {%
3331       \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
3332       \ifcsdef{\@glxtr@attrval}%
3333       {%
3334         \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
3335       }%
3336       {%
3337         \GlossariesExtraWarning{Unknown control sequence name
3338           '\@glxtr@attrval' supplied in glossdescfont attribute
3339           for entry '#1'. Ignoring}%
3340         \let\@glxtr@glossdescfont\@firstofone
3341       }%
3342     }%
3343     {\let\@glxtr@glossdescfont\@firstofone}%
3344     \glsifattribute{#1}{glossdesc}{firstuc}%
3345     {%
3346       \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
3347     }%
3348     {%
3349       \glsifattribute{#1}{glossdesc}{title}%
3350       {%
3351         \@glxtr@do@titlecaps@warn
3352         \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
3353       }%
3354       {%
3355         \@glxtr@glossdescfont{\glentrydesc{#1}}%
3356       }%
3357     }%
3358   }%
3359 }
3360 }

```

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

```

3361 \ifpackageloaded{glossaries-accsupp}
3362 {
3363   \renewcommand*{\glossentryname}[1]{%
3364     \glsdoifexistsorwarn{#1}%
3365     {%
3366       \glssetabbrvfmt{\glscategory{#1}}%

```

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

```

3367   \glshasattribute{#1}{glossnamefont}%
3368   {%
3369     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%

```

```

3370 \ifcsdef{\@glxtr@attrval}%
3371 {%
3372 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
3373 }%
3374 {%
3375 \GlossariesExtraWarning{Unknown control sequence name
3376 '\@glxtr@attrval' supplied in glossnamefont attribute
3377 for entry '#1'. Reverting to default \string\glsnamefont}%
3378 \let\@glxtr@glossnamefont\glsnamefont
3379 }%
3380 }%
3381 {\let\@glxtr@glossnamefont\glsnamefont}%
3382 \glsifattribute{#1}{glossname}{firstuc}%
3383 {%
3384 \glsnameaccessdisplay
3385 {%
3386 \@glxtr@glossnamefont{\Glsentryname{#1}}%
3387 }%
3388 {#1}%
3389 }%
3390 {%
3391 \glsifattribute{#1}{glossname}{title}%
3392 {%
3393 \@glxtr@do@titlecaps@warn
3394 \glsnameaccessdisplay
3395 {%
3396 \@glxtr@glossnamefont{\@glxtr@field@titlecase{#1}{name}}%
3397 }%
3398 {#1}%
3399 }%
3400 {%
3401 \glsifattribute{#1}{glossname}{uc}%
3402 {%
3403 \glsnameaccessdisplay
3404 {%

```

Hide the label from the upper-casing command.

```

3405 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
3406 \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
3407 }%
3408 {#1}%
3409 }%
3410 {%
3411 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
3412 \glsnameaccessdisplay
3413 {%
3414 \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
3415 }%
3416 {#1}%
3417 }%

```

```

3418     }%
3419 }%

Do post-name hook:
3420     \glxtrpostnamehook{#1}%
3421 }%
3422 }
3423 }
3424 {
3425     \renewcommand*{\glossentryname}[1]{%
3426         \@glsdoifexistsorwarn{#1}%
3427         {%
3428             \glsetabbrvfmt{\glscategory{#1}}%
3429             \glshasattribute{#1}{glossnamefont}%
3430             {%
3431                 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
3432                 \ifcsdef{\@glxtr@attrval}%
3433                 {%
3434                     \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
3435                 }%
3436                 {%
3437                     \GlossariesExtraWarning{Unknown control sequence name
3438                         '\@glxtr@attrval' supplied in glossnamefont attribute
3439                         for entry '#1'. Reverting to default \string\glsgnamefont}%
3440                     \let\@glxtr@glossnamefont\glsgnamefont
3441                 }%
3442             }%
3443             {\let\@glxtr@glossnamefont\glsgnamefont}%
3444             \glsgifattribute{#1}{glossname}{firstuc}%
3445             {%
3446                 \@glxtr@glossnamefont{\Glsentryname{#1}}%
3447             }%
3448             {%
3449                 \glsgifattribute{#1}{glossname}{title}%
3450                 {%
3451                     \@glxtr@do@titlecaps@warn
3452                     \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
3453                 }%
3454                 {%
3455                     \glsgifattribute{#1}{glossname}{uc}%
3456                     {%

```

Hide the label from the upper-casing command.

```

3457         \letcs{\glo@name}{\glo@\glsgdetoklabel{#1}@name}%
3458         \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
3459     }%
3460 }%

```

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

```

3461         \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
3462         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
3463     }%
3464 }%
3465 }%

```

Do post-name hook.

```

3466     \glxtrpostnamehook{#1}%
3467 }%
3468 }
3469 }

```

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

```

3470 \ifpackageloaded{glossaries-accsupp}
3471 {
3472     \renewcommand*{\Glossentryname}[1]{%
3473         \@glsdofexistsorwarn{#1}%
3474         {%
3475             \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

3476     \glshasattribute{#1}{glossnamefont}%
3477     {%
3478         \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
3479         \ifcsdef{\@glxtr@attrval}%
3480         {%
3481             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
3482         }%
3483         {%
3484             \GlossariesExtraWarning{Unknown control sequence name
3485                 '\@glxtr@attrval' supplied in glossnamefont attribute
3486                 for entry '#1'. Reverting to default \string\glsnamefont}%
3487             \let\@glxtr@glossnamefont\glsnamefont
3488         }%
3489     }%
3490     {\let\@glxtr@glossnamefont\glsnamefont}%
3491     \glsnameaccessdisplay
3492     {%
3493         \@glxtr@glossnamefont{\Glsentryname{#1}}%
3494     }%
3495     {#1}%

```

Do post-name hook:

```

3496     \glxtrpostnamehook{#1}%
3497 }%
3498 }
3499 }
3500 {
3501     \renewcommand*{\Glossentryname}[1]{%
3502         \@glsdofexistsorwarn{#1}%

```



```

3503   {%
3504     \glsetabbrvfmt{\glscategory{#1}}%
3505     \glshasattribute{#1}{glossnamefont}%
3506     {%
3507       \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
3508       \ifcsdef{\@glxtr@attrval}%
3509       {%
3510         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
3511       }%
3512     {%
3513       \GlossariesExtraWarning{Unknown control sequence name
3514         '\@glxtr@attrval' supplied in glossnamefont attribute
3515         for entry '#1'. Reverting to default \string\glsgnamefont}%
3516       \let\@glxtr@glossnamefont\glsgnamefont
3517     }%
3518   }%
3519   {\let\@glxtr@glossnamefont\glsgnamefont}%
3520   \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

3521     \glxtrpostnamehook{#1}%
3522   }%
3523 }
3524 }

```

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.

```

3525 \newcommand*{\glxtrpostnamehook}[1]{%
3526   \def\@glsgnumberformat{\glsgnumberformat}%
3527   \glxtrdoautoindexname{#1}{indexname}%

```

Allow categories to hook in here.

```

3528   \csuse{\glxtrpostname\glscategory{\glsgcurrententrylabel}}%
3529 }

```

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

```

3530 \newif\ifglxtr@format@override
3531 \@glxtr@format@overridefalse

```

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

`xFormatOverride`

```

3532 \@ifpackageloaded{hyperref}
3533 {

```

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

```

3534 \ifHy@hyperindex
3535   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3536     \@glstr@format@override true
3537     \appto\theindex{\let\glshypernumber\@firstofone}%
3538   }
3539 \else
3540   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3541     \@glstr@format@override true
3542     \appto\theindex{\let\glshypernumber\hyperpage}%
3543   }
3544 \fi
3545 }
3546 {
3547   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
3548     \@glstr@format@override true
3549   }
3550 }
3551 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

3552 \newcommand*{\glstrdoautoindexname}[2]{%
3553   \glshasattribute{#1}{#2}%
3554   {%

```

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.

```

3555   \@glstr@autoindex@setname{#1}%

```

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

```

3556   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
3557   \if@glstr@format@override
3558     \ifdefstring{\@glstr@numberformat}{\glstr@numberformat}{}%
3559     {\let\@glstr@attrval\@glstr@numberformat}%
3560   \fi
3561   \ifdefstring{\@glstr@attrval}{true}%
3562   {%
3563     {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
3564     \expandafter\index\expandafter{\@glo@name}%
3565   }%
3566   {%
3567 }

```

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

```

3568 \newcommand*{\@glstr@autoindex@setname}[1]{%
3569   \def\@glo@name{\string\glstryname{#1}}%
3570   \glsetentryfield{\@glo@sort}{#1}{sort}%
3571   \@gls@checkmkidxchars\@glo@sort

```

```

3572 \@glstr@autoindex@doextra@esc\@glo@sort
3573 \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
3574 }

```

dex@doextra@esc

```

3575 \newcommand*{\@glstr@autoindex@doextra@esc}[1]{%

```

Escape the escape character unless it has already been escaped.

```

3576 \ifx\@glstr@autoindex@esc\@gls@quotechar
3577 \else
3578 \def\@gls@checkedmkidx{%
3579 \edef\@glstr@checkspch{%
3580 \noexpand\@glstr@autoindex@escquote\expandonce{#1}%
3581 \noexpand\@empty\@glstr@autoindex@esc\noexpand\@nnil
3582 \@glstr@autoindex@esc\noexpand\@empty\noexpand\@glstr@endescspch}%
3583 \@glstr@checkspch
3584 \let#1\@gls@checkedmkidx\relax
3585 \fi

```

Escape actual character unless it has already been escaped.

```

3586 \ifx\@glstr@autoindex@at\@gls@actualchar
3587 \else
3588 \def\@gls@checkedmkidx{%
3589 \edef\@glstr@checkspch{%
3590 \noexpand\@glstr@autoindex@escat\expandonce{#1}%
3591 \noexpand\@empty\@glstr@autoindex@at\noexpand\@nnil
3592 \@glstr@autoindex@at\noexpand\@empty\noexpand\@glstr@endescspch}%
3593 \@glstr@checkspch
3594 \let#1\@gls@checkedmkidx\relax
3595 \fi

```

Escape level character unless it has already been escaped.

```

3596 \ifx\@glstr@autoindex@level\@gls@levelchar
3597 \else
3598 \def\@gls@checkedmkidx{%
3599 \edef\@glstr@checkspch{%
3600 \noexpand\@glstr@autoindex@esclevel\expandonce{#1}%
3601 \noexpand\@empty\@glstr@autoindex@level\noexpand\@nnil
3602 \@glstr@autoindex@level\noexpand\@empty\noexpand\@glstr@endescspch}%
3603 \@glstr@checkspch
3604 \let#1\@gls@checkedmkidx\relax
3605 \fi

```

Escape encap character unless it has already been escaped.

```

3606 \ifx\@glstr@autoindex@encap\@gls@encapchar
3607 \else
3608 \def\@gls@checkedmkidx{%
3609 \edef\@glstr@checkspch{%
3610 \noexpand\@glstr@autoindex@escencap\expandonce{#1}%
3611 \noexpand\@empty\@glstr@autoindex@encap\noexpand\@nnil
3612 \@glstr@autoindex@encap\noexpand\@empty\noexpand\@glstr@endescspch}%

```

```

3613 \@@glxtr@checkspch
3614 \let#1\@glx@checkedmkidx\relax
3615 \fi
3616 }

```

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`.

```
3617 \newcommand*{\@glxtr@autoindex@at}{}
```

`trSetActualChar` Set the actual character.

```

3618 \newcommand*{\GlsXtrSetActualChar}[1]{%
3619 \gdef\@glxtr@autoindex@at{#1}%
3620 \def\@glxtr@autoindex@escat##1#1##2#1##3\@glxtr@endescspch{%
3621 \@@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escat}{##1}{##2}{##3}%
3622 }%
3623 }
3624 \@onlypreamble\GlsXtrSetActualChar
3625 \makeatother
3626 \GlsXtrSetActualChar{@}
3627 \makeatletter

```

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

```
3628 \newcommand*{\@glxtr@autoindex@encap}{}
```

`XtrSetEncapChar` Set the encap character.

```

3629 \newcommand*{\GlsXtrSetEncapChar}[1]{%
3630 \gdef\@glxtr@autoindex@encap{#1}%
3631 \def\@glxtr@autoindex@escencap##1#1##2#1##3\@glxtr@endescspch{%
3632 \@@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
3633 }%
3634 }
3635 \GlsXtrSetEncapChar{|}
3636 \@onlypreamble\GlsXtrSetEncapChar

```

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

```
3637 \newcommand*{\@glxtr@autoindex@level}{}
```

`XtrSetLevelChar` Set the encap character.

```

3638 \newcommand*{\GlsXtrSetLevelChar}[1]{%
3639 \gdef\@glxtr@autoindex@level{#1}%
3640 \def\@glxtr@autoindex@esclevel##1#1##2#1##3\@glxtr@endescspch{%
3641 \@@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
3642 }%
3643 }
3644 \GlsXtrSetLevelChar{!}
3645 \@onlypreamble\GlsXtrSetLevelChar

```

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

```
3646 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lXtrSetEscChar Set the escape character.

```
3647 \newcommand*{\GlsXtrSetEscChar}[1]{%
3648   \gdef\@glsxtr@autoindex@esc{#1}%
3649   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
3650     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
3651   }%
3652 }
3653 \GlsXtrSetEscChar{"}
3654 \@onlypreamble\GlsXtrSetEscChar
```

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

```
3655 \ifdef\actualchar
3656 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
3657 {}
```

Quote character \quotechar:

```
3658 \ifdef\quotechar
3659 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
3660 {}
```

Level character \levelchar:

```
3661 \ifdef\levelchar
3662 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
3663 {}
```

Encap character \encapchar:

```
3664 \ifdef\encapchar
3665 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
3666 {}
```

leto@endescspch

```
3667 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

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

```
3668 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
3669   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3670   \toks@={#3}%
3671   \ifx\@nnil#3\relax
3672     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
3673   \else
3674     \ifx\@nnil#4\relax
3675       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3676       \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```

3677         #4#5\@glxtr@endescspch}%
3678     \else
3679         \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3680         \@glxtr@autoindex@esc#1}%
3681         \def\@@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
3682     \fi
3683 \fi
3684 \@glxtr@checkspch
3685 }

```

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

```

3686 \renewcommand*{\Glossentrydesc}[1]{%
3687     \glsdoifexistsorwarn{#1}%
3688     {%
3689         \glssetabbrvfmt{\glscategory{#1}}%
3690         \Glsaccessdesc{#1}%
3691     }%
3692 }

```

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

```

3693 \renewcommand*{\Glossentrysymbol}[1]{%
3694     \glsdoifexistsorwarn{#1}%
3695     {%
3696         \glssetabbrvfmt{\glscategory{#1}}%
3697         \Glsaccesssymbol{#1}%
3698     }%
3699 }

```

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

```

3700 \renewcommand*{\Glossentrysymbol}[1]{%
3701     \glsdoifexistsorwarn{#1}%
3702     {%
3703         \glssetabbrvfmt{\glscategory{#1}}%
3704         \Glsaccesssymbol{#1}%
3705     }%
3706 }

```

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

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

```

3707 \newcommand*{\GlsXtrEnableInitialTagging}{%
3708     \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
3709 }
3710 \@onlypreamble\GlsXtrEnableInitialTagging

```

\GlsXtrEnableInitialTagging Starred version undefines command.

```

3711 \newcommand*{\s@glxstr@enabletagging}[2]{%
3712   \undef#2%
3713   \@glxstr@enabletagging{#1}{#2}%
3714 }

```

r@enabletagging Internal command.

```

3715 \newcommand*{\@glxstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
3716   \@for\@glxstr@cat:=#1\do
3717   {%
3718     \ifdefempty\@glxstr@cat
3719     {}%
3720     {\glsssetcategoryattribute{\@glxstr@cat}{tagging}{true}}%
3721   }%
3722   \newrobustcmd*#2[1]{##1}%
3723   \def\@glxstr@taggingcs{#2}%
3724   \renewcommand*{\@glxstr@activate@initialtagging{%
3725     \let#2\@glxstr@tag
3726   }%
3727   \ifundef\@glspreglossaryhook
3728   {\GlossariesExtraWarning{Initial tagging requires at least
3729     glossaries.sty v4.19 to work correctly}}%
3730   {}%
3731 }

```

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

```

3732 \ifundef\mfu@checkword@do
3733 {
3734   \newcommand*{\mfu@checkword@do}[1]{%
3735     \ifdefstring{\mfu@checkword@arg}{#1}%
3736     {%
3737       \let\@mfu@domakefirstuc\@firstofone
3738       \listbreak
3739     }%
3740     {}%
3741   }

```

\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.

```

3742   \ifundef\mfu@checkword
3743   {
3744     \newcommand{\@glxstr@do@titlecaps@warn}{%
3745       \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
3746         support not available}%

```

One warning should suffice.

```
3747 \let\@glsxtr@do@titlecaps@warn\relax
3748 }
3749 }
3750 {
3751 \renewcommand*{\mfu@checkword}[1]{%
3752 \def\mfu@checkword@arg{#1}%
3753 \let\@mfu@domakefirstuc\makefirstuc
3754 \forlistloop\mfu@checkword@do\@mfu@nocaplist
3755 }
3756 }
3757 }
3758 {}% no patch required
```

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

```
3759 \newcommand*{\@glsxtr@do@titlecaps@warn}{}%
```

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

```
3760 \newcommand*{\@glsxtr@activate@initialtagging}{}%
```

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

```
3761 \newrobustcmd*{\@glsxtr@tag}[1]{%
3762 \glsifattribute{\glscurrententrylabel}{tagging}{true}%
3763 {\glsxtrtagfont{#1}}{#1}%
3764 }
```

\glsxtrtagfont Used in the glossary.

```
3765 \newcommand*{\glsxtrtagfont}[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.

```
3766 \ifdef\@gls@preglossaryhook
3767 {
3768 \renewcommand*{\@gls@preglossaryhook}{%
3769 \@glsxtr@activate@initialtagging
3770 \let\@glsxtr@org@postdescription\glspostdescription
3771 \renewcommand*{\glspostdescription}{%
3772 \ifglsentryexists{\glscurrententrylabel}%
3773 {%
3774 \glsxtrpostdescription
3775 \@glsxtr@org@postdescription
3776 }{}%
3777 }%
}
```

Enable the options used by \@glsxtrp:

```
3778 \glossxtrsetpopts
```



```

3779 }%
3780 }
3781 {}

postdescription This command will only be used if \@gls@preglossaryhook is available and the glossary
style uses \glspostdescription without modifying it. (\nopostdesc will suppress this.)
The glossaries-extra-stylemods package will add the post description hook to all the prede-
fined styles that don't include it.
3782 \newcommand*{\glsxtrpostdescription}{%
3783 \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}}%
3784 }

postdescgeneral
3785 \newcommand*{\glsxtrpostdescgeneral}{}

xtrpostdescterm
3786 \newcommand*{\glsxtrpostdescterm}{}

postdescacronym
3787 \newcommand*{\glsxtrpostdescacronym}{}

descabbreviation
3788 \newcommand*{\glsxtrpostdescabbreviation}{}

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.
3789 \renewcommand*{\glspostlinkhook}{%
3790 \ifglseentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
3791 }

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.
3792 \newcommand*{\glsxtrpostlinkhook}{%
3793 \glsxtrdiscardperiod{\glslabel}%
3794 {\glsxtrpostlinkendsentence}%
3795 {\glsxtrpostlink}%
3796 }

\glsxtrpostlink
3797 \newcommand*{\glsxtrpostlink}{%
3798 \csuse{glsxtrpostlink\glscategory{\glslabel}}}%
3799 }

linkendsentence Done by \glsxtrpostlinkhook if a full stop is discarded.
3800 \newcommand*{\glsxtrpostlinkendsentence}{%
3801 \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
3802 {%
3803 \csuse{glsxtrpostlink\glscategory{\glslabel}}%

```

Put the full stop back.

```
3804 .\spacefactor\sffcode'\. \relax
3805 }%
3806 {%
```

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

```
3807 \spacefactor\sffcode'\. \relax
3808 }%
3809 }
```

DescOnFirstUse 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.

```
3810 \newcommand*\glstrpostlinkAddDescOnFirstUse{%
3811 \glstrifwasfirstuse{\space(\glssaccessdesc{\glslabel})}{}%
3812 }
```

SymbolOnFirstUse 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.

```
3813 \newcommand*\glstrpostlinkAddSymbolOnFirstUse{%
3814 \glstrifwasfirstuse
3815 {%
3816 \ifglshassymbol{\glslabel}{\space(\glssaccesssymbol{\glslabel})}{}%
3817 }%
3818 {}%
3819 }
```

trdiscardperiod 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).

```
3820 \newcommand*\glstrdiscardperiod}[3]{%
3821 \glstrifwasfirstuse
3822 {%
3823 \glusifattribute{#1}{retainfirstuseperiod}{true}%
3824 {#3}%
3825 {%
3826 \glusifattribute{#1}{discardperiod}{true}%
3827 {%
3828 \glusifplural
3829 {%
3830 \glusifattribute{#1}{pluraldiscardperiod}{true}%
3831 {\glstrifperiod{#2}{#3}}%
3832 {#3}%
3833 }%
3834 {%
3835 \glstrifperiod{#2}{#3}%
3836 }%
```

```

3837     }%
3838     {#3}%
3839   }%
3840 }%
3841 {%
3842   \glsifattribute{#1}{discardperiod}{true}%
3843   {%
3844     \glsifplural
3845     {%
3846       \glsifattribute{#1}{pluraldiscardperiod}{true}%
3847       {\glsxtrifperiod{#2}{#3}}%
3848       {#3}%
3849     }%
3850     {%
3851       \glsxtrifperiod{#2}{#3}%
3852     }%
3853   }%
3854   {#3}%
3855 }%
3856 }

```

`\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`

```

3857 \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 ' ').

```

3858 \newcommand*{\glsxtr@punclist}{.,:;?!}

```

`punctuationmark` Add character to punctuation list.

```

3859 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`punctuationmarks` Reset the punctuation list.

```

3860 \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`.)

```

3861 \newcommand*{\glsxtrifnextpunc}[2]{%
3862   \def\reserved@a{#1}%
3863   \def\reserved@b{#2}%
3864   \futurelet\@glspunc@token\glsxtr@ifnextpunc
3865 }

```

sxtr@ifnextpunc

```
3866 \newcommand*{\glsxtr@ifnextpunc}{%
3867   \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
3868   \reserved@b
3869 }
```

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

```
3870 \newcommand*{\glsxtr@ifpunctoken}[1]{%
3871   \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
3872 }
```

xtr@ifpunctoken

```
3873 \def\@glsxtr@ifpunctoken#1#2{%
3874   \let\reserved@d=#2%
3875   \ifx\reserved@d\@nnil
3876     \let\glsxtr@next\@glsxtr@notfoundinlist
3877   \else
3878     \ifx#1\reserved@d
3879       \let\glsxtr@next\@glsxtr@foundinlist
3880     \else
3881       \let\glsxtr@next\@glsxtr@ifpunctoken
3882     \fi
3883   \fi
3884   \glsxtr@next#1%
3885 }
```

xtr@foundinlist

```
3886 \def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
3887 \def\@glsxtr@notfoundinlist#1{\@secondoftwo}
```

glsxtrdopostpunc

```
\glsxtrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
3888 \newcommand{\glsxtrdopostpunc}[1]{%
3889   \glsxtr@ifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
3890 }
```

@glsxtr@swaptwo

```
3891 \newcommand{\@glsxtr@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.

```
3892 \define@key{glstrabbrv}{category}{%
3893   \edef\glscategorylabel{#1}%
3894   \ifcsdef{@glstrabbrv@current@#1}%
3895   {%
```

Warning should already have been issued.

```
3896   \let\@glstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
3897   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
3898   \glstr@applyabbrvstyle{\csname @glstrabbrv@current@#1\endcsname}%
3899   \let\GlsXtrWarnDeprecatedAbbrStyle\@glstr@orgwarndep
3900 }%
3901 {}%
3902 }
```

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

```
3903 \define@key{glstrabbrv}{shortplural}{%
3904   \def\@glstr@shortpl{#1}%
3905 }
```

Similarly for the long plural form.

```
3906 \define@key{glstrabbrv}{longplural}{%
3907   \def\@glstr@longpl{#1}%
3908 }
```

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

`\glsshortpltok`

```
3909 \newtoks\glsshortpltok
```

`\glslongpltok`

```
3910 \newtoks\glslongpltok
```

`\glstr@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.

```
3911 \newcommand*{\@glstr@insertdots}[2]{%
3912   \def#1{%
3913     \@glstr@insert@dots#1#2\@nnil
3914   }
```

xtr@insert@dots

```
3915 \newcommand*{\@glsxtr@insert@dots}[2]{%
3916   \ifx\@nnil#2\relax
3917   \let\@glsxtr@insert@dots@next\@gobble
3918   \else
3919   \ifx\relax#2\relax
3920   \else
3921     \appto#1{#2.}%
3922   \fi
3923   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
3924   \fi
3925   \@glsxtr@insert@dots@next#1%
3926 }
```

newabbreviation Define a new generic abbreviation.

```
3927 \newcommand*{\newabbreviation}[4][]{%
3928   \glskeylisttok{#1}%
3929   \glslabeltok{#2}%
3930   \glsshorttok{#3}%
3931   \glslongtok{#4}%

   Get the category.
3932   \def\glscategorylabel{abbreviation}%
3933   \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
3934   \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%

   Set the default long plural
3935   \def\@gls@longpl{#4\glspluralsuffix}%

   Has the insertdots attribute been set?
3936   \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
3937   {%
3938     \@glsxtr@insertdots\@gls@short{#3}%
3939     \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
3940     \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3941     {%
3942       \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3943         '\abbrvpluralsuffix}%
3944     }%
3945     {%
3946       \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
3947       {%
3948         \let\@gls@shortpl\@gls@short
3949       }%
3950     }%
3951     \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
3952       \abbrvpluralsuffix}%
3953   }%
3954 }%
3955 }%
3956 }
```

insertdots not true.

```
3957 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
3958 {%
3959 \def\@gls@shortpl{#3'\abbrvpluralsuffix}%
3960 }%
3961 {%
3962 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
3963 {%
3964 \def\@gls@shortpl{#3}%
3965 }%
3966 {%
3967 \def\@gls@shortpl{#3\abbrvpluralsuffix}%
3968 }%
3969 }%
3970 }%
```

Hook for further customisation if required:

```
3971 \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

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

```
3972 \setkeys*{glsxtrabbrv}[category]{#1}%
```

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

```
3973 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
3974 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Do any extra setup provided by hook:

```
3975 \newabbreviationhook
```

Define this entry:

```
3976 \protected@edef\@do@newglossaryentry{%
3977 \noexpand\newglossaryentry{\the\glslabeltok}%
3978 {%
3979 type=\glsxtrabbrvtype,%
3980 category=abbreviation,%
3981 short={\the\glsshorttok},%
3982 shortplural={\the\glsshortpltok},%
3983 long={\the\glslongtok},%
3984 longplural={\the\glslongpltok},%
3985 name={\the\glsshorttok},%
3986 \CustomAbbreviationFields,%
3987 \the\glskeylisttok
3988 }%
3989 }%
3990 \@do@newglossaryentry
3991 \GlsXtrPostNewAbbreviation
3992 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

```
3993 \newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```

NewAbbreviation Hook used by abbreviation styles.

```
3994 \newcommand*{\GlsXtrPostNewAbbreviation}{}
```

bbreviationhook Hook for use with `\newabbreviation`.

```
3995 \newcommand*{\newabbreviationhook}{}%
```

reviationFields

```
3996 \newcommand*{\CustomAbbreviationFields}{}%
```

lsxtrfullformat Full format without case change.

```
3997 \newcommand*{\glsxtrfullformat}[2]{%
3998   \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
3999   (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
4000 }
```

lsxtrfullformat Full format with case change.

```
4001 \newcommand*{\Glsxtrfullformat}[2]{%
4002   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
4003   (\protect\glsfirstabbrvfont{\Glsaccessshort{#1}})%
4004 }
```

xtrfullplformat Plural full format without case change.

```
4005 \newcommand*{\glsxtrfullplformat}[2]{%
4006   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
4007   (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
4008 }
```

xtrfullplformat Plural full format with case change.

```
4009 \newcommand*{\Glsxtrfullplformat}[2]{%
4010   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
4011   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
4012 }
```

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

```
4013 \newcommand*{\glsxtrfullsep}[1]{\space}
```

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

inlinefullformat Full format without case change.

```
4014 \newcommand*{\glsxtrininlinefullformat}{\glsxtrfullformat}
```

inlinefullformat Full format with case change.

```
4015 \newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
4016 \newcommand*{\glsxtrininlinefullplformat}{\glsxtrfullplformat}
```


inlinefullplformat Plural full format with case change.
4017 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}

Redefine \glsentryfull 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 \glsxtrfull set of commands instead.

\glsentryfull
4018 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}

\Glsentryfull
4019 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}

\glsentryfullpl
4020 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}

\Glsentryfullpl
4021 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}

gsfirstabbrvfont Font changing command used for the abbreviation on first use or in the full format.
4022 \newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}

gsabbrvdefaultfont Font changing command used for the abbreviation on first use or in the full format.
4023 \newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}

\glsabbrvfont Font changing command used for the abbreviation on subsequent use.
4024 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}

gsabbrvdefaultfont
4025 \newcommand*{\glsabbrvdefaultfont}[1]{#1}

\glslongfont Font changing command used for the long form in commands like \glsxtrlong.
4026 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}

gslongdefaultfont Default font changing command used for the long form in commands like \glsxtrlong.
4027 \newcommand*{\glslongdefaultfont}[1]{#1}

glsfirstlongfont Font changing command used for the long form on first use or in the full format.
4028 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}

gslongdefaultfont
4029 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}

gsbrvpluralsuffix Default plural suffix.
4030 \newcommand*{\abbrvpluralsuffix}{\glspluralsuffix}

`\glxtrfull` Full form (no case-change).

```
4031 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\ns@glxtrfull}
4032 \newcommand*\ns@glxtrfull[2][\%
4033   \new@ifnextchar[{\@glxtr@full{#1}{#2}}]{%
4034     {\@glxtr@full{#1}{#2}[]}%
4035 }
```

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

```
4036 \def\@glxtr@full#1#2[#3]{%
4037   \glsdoifexists{#2}%
4038   {%
4039     \glssetabbrvfmt{\glscategory{#2}}%
4040     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4041     \let\glsifplural\@secondoftwo
4042     \let\glscapscase\@firstofthree
4043     \let\glsinsert\@empty
4044     \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.

```
4045   \glxtrsetupfulldefs
4046   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4047   }%
4048   \glspostlinkhook
4049 }
```

`trsetupfulldefs`

```
4050 \newcommand*{\glxtrsetupfulldefs}{%
4051   \let\glxtrifwasfirstuse\@firstoftwo
4052 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
4053 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
4054 \newcommand*\ns@Glsxtrfull[2][\%
4055   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}]{%
4056     {\@Glsxtr@full{#1}{#2}[]}%
4057 }
```

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

```
4058 \def\@Glsxtr@full#1#2[#3]{%
4059   \glsdoifexists{#2}%
4060   {%
4061     \glssetabbrvfmt{\glscategory{#2}}%
4062     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4063     \let\glsifplural\@secondoftwo
4064     \let\glscapscase\@secondofthree

```

```

4065 \let\glsinsert\@empty
4066 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
4067 \glsxtrsetupfulldefs
4068 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4069 }%
4070 \glspostlinkhook
4071 }

```

\Glsxtrfull Full form (all uppercase).

```

4072 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\@ns@Glsxtrfull}
4073 \newcommand*\ns@Glsxtrfull[2][{}]{%
4074 \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
4075 {\@Glsxtr@full{#1}{#2}[]}%
4076 }

```

\@Glsxtr@full Low-level macro:

```

4077 \def\@Glsxtr@full#1#2[#3]{%
4078 \glsdoifexists{#2}%
4079 {%
4080 \glssetabbrvfmt{\glscategory{#2}}%
4081 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4082 \let\glsifplural\@secondoftwo
4083 \let\glscapscase\@thirdofthree
4084 \let\glsinsert\@empty
4085 \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
4086 \glsxtrsetupfulldefs
4087 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4088 }%
4089 \glspostlinkhook
4090 }

```

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

```

4091 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\@ns@glsxtrfullpl}
4092 \newcommand*\ns@glsxtrfullpl[2][{}]{%
4093 \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
4094 {\@glsxtr@fullpl{#1}{#2}[]}%
4095 }

```

\@glsxtr@fullpl Low-level macro:

```

4096 \def\@glsxtr@fullpl#1#2[#3]{%
4097 \glsdoifexists{#2}%
4098 {%
4099 \glssetabbrvfmt{\glscategory{#2}}%
4100 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4101 \let\glsifplural\@firstoftwo
4102 \let\glscapscase\@firstofthree
4103 \let\glsinsert\@empty
4104 \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
4105 \glsxtrsetupfulldefs

```

```

4106 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4107 }%
4108 \glspostlinkhook
4109 }

```

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

```

4110 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
4111 \newcommand*\ns@Glsxtrfullpl[2][]{\%
4112 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
4113 {\@Glsxtr@fullpl{#1}{#2}[]}%
4114 }

```

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

```

4115 \def\@Glsxtr@fullpl#1#2[#3]{\%
4116 \glsdoifexists{#2}%
4117 {\%
4118 \glssetabbrvfmt{\glscategory{#2}}%
4119 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4120 \let\glsifplural\@firstoftwo
4121 \let\glscapscase\@secondofthree
4122 \let\glsinsert\@empty
4123 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
4124 \glsxtrsetupfulldefs
4125 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4126 }%
4127 \glspostlinkhook
4128 }

```

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

```

4129 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
4130 \newcommand*\ns@GLSxtrfullpl[2][]{\%
4131 \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
4132 {\@GLSxtr@fullpl{#1}{#2}[]}%
4133 }

```

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

```

4134 \def\@GLSxtr@fullpl#1#2[#3]{\%
4135 \glsdoifexists{#2}%
4136 {\%
4137 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4138 \let\glsifplural\@firstoftwo
4139 \let\glscapscase\@thirdofthree
4140 \let\glsinsert\@empty
4141 \def\glscustomtext{\%
4142 \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
4143 \glsxtrsetupfulldefs
4144 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4145 }%
4146 \glspostlinkhook

```

4147 }

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

`\glsxtrshort`

4148 `\newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\ns@glsxtrshort}`

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

4149 `\newcommand*{\ns@glsxtrshort}[2][\%`

4150 `\new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2}[]}%`

4151 `}`

Read in the final optional argument:

4152 `\def\@glsxtrshort#1#2[#3]{\%`

4153 `\glsdoifexists{#2}%`

4154 `{\%`

Need to make sure `\glsabbrvfont` is set correctly.

4155 `\glssetabbrvfmt{\glscategory{#2}}%`

4156 `\let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper`

4157 `\let\glsxtrifwasfirstuse\@secondoftwo`

4158 `\let\glsifplural\@secondoftwo`

4159 `\let\glsapscase\@firstofthree`

4160 `\let\glsinsert\@empty`

4161 `\def\glscustomtext{\%`

4162 `\glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%`

4163 `\ifglsxtrininsertinside\else#3\fi`

4164 `}%`

4165 `\@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%`

4166 `}%`

4167 `\glspostlinkhook`

4168 `}`

`\Glsxtrshort`

4169 `\newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}`

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

4170 `\newcommand*{\ns@Glsxtrshort}[2][\%`

4171 `\new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2}[]}%`

4172 `}`

Read in the final optional argument:

4173 `\def\@Glsxtrshort#1#2[#3]{\%`

4174 `\glsdoifexists{#2}%`

4175 `{\%`

4176 `\glssetabbrvfmt{\glscategory{#2}}%`

4177 `\let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper`

4178 `\let\glsxtrifwasfirstuse\@secondoftwo`

4179 `\let\glsifplural\@secondoftwo`

4180 `\let\glsapscase\@secondofthree`

4181 `\let\glsinsert\@empty`

```

4182 \def\glscustomtext{%
4183 \glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
4184 \ifglxtrinsertinside\else#3\fi
4185 }%
4186 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4187 }%
4188 \glspostlinkhook
4189 }

```

\GLSxtrshort

```

4190 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
4191 \newcommand*{\ns@GLSxtrshort}[2][{}]{%
4192 \new@ifnextchar[\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2}[{}]}%
4193 }
    Read in the final optional argument:
4194 \def\@GLSxtrshort#1#2[#3]{%
4195 \glsdoifexists{#2}%
4196 {%
4197 \glssetabbrvfmt{\glscategory{#2}}%
4198 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4199 \let\glxtrifwasfirstuse\@secondoftwo
4200 \let\glsifplural\@secondoftwo
4201 \let\glscapscase\@thirdofthree
4202 \let\glsinsert\@empty
4203 \def\glscustomtext{%
4204 \mfirstucMakeUppercase
4205 {\glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
4206 \ifglxtrinsertinside\else#3\fi
4207 }%
4208 }%
4209 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4210 }%
4211 \glspostlinkhook
4212 }

```

\glxtrlong

```

4213 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
4214 \newcommand*{\ns@glxtrlong}[2][{}]{%
4215 \new@ifnextchar[\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2}[{}]}%
4216 }
    Read in the final optional argument:
4217 \def\@glxtrlong#1#2[#3]{%
4218 \glsdoifexists{#2}%
4219 {%
4220 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4221 \let\glxtrifwasfirstuse\@secondoftwo
4222 \let\glxifplural\@secondoftwo
4223 \let\glscapscase\@firstofthree
4224 \let\glxinsert\@empty
4225 \def\glscustomtext{%
4226   \glslongfont{\glssaccesslong{#2}\ifglxtrinsertinside#3\fi}%
4227   \ifglxtrinsertinside\else#3\fi
4228 }%
4229 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4230 }%
4231 \glspostlinkhook
4232 }

```

\Glsxtrlong

```

4233 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
4234 \newcommand*{\ns@Glsxtrlong}[2][]{%
4235   \new@ifnextchar[\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2}[]}%
4236 }

    Read in the final optional argument:
4237 \def\@Glsxtrlong#1#2[#3]{%
4238   \glsdofexists{#2}%
4239   {%
4240     \let\do@glx@link@checkfirsthyper\@glx@link@nocheckfirsthyper
4241     \let\glxtrifwasfirstuse\@secondoftwo
4242     \let\glxifplural\@secondoftwo
4243     \let\glscapscase\@secondofthree
4244     \let\glxinsert\@empty
4245     \def\glscustomtext{%
4246       \glslongfont{\Glsaccesslong{#2}\ifglxtrinsertinside#3\fi}%
4247       \ifglxtrinsertinside\else#3\fi
4248     }%
4249     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4250   }%
4251   \glspostlinkhook
4252 }

```

\GLSxtrlong

```

4253 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
4254 \newcommand*{\ns@GLSxtrlong}[2][]{%
4255   \new@ifnextchar[\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
4256 }

    Read in the final optional argument:
4257 \def\@GLSxtrlong#1#2[#3]{%
4258   \glsdofexists{#2}%
4259   {%

```

```

4260 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4261 \let\glxtrifwasfirstuse\@secondoftwo
4262 \let\gl@ifplural\@secondoftwo
4263 \let\gl@scapscase\@thirdofthree
4264 \let\gl@insert\@empty
4265 \def\glscustomtext{%
4266 \mfirstucMakeUppercase
4267 {\gl@longfont{\gl@accesslong{#2}\ifglxtrininsertinside#3\fi}%
4268 \ifglxtrininsertinside\else#3\fi
4269 }%
4270 }%
4271 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4272 }%
4273 \glspostlinkhook
4274 }

```

Plural short forms:

\glxtrshortpl

```

4275 \newrobustcmd*{\glxtrshortpl}{\@gl@hyp@opt\ns@glxtrshortpl}
Define the un-starred form. Need to determine if there is a final optional argument
4276 \newcommand*{\ns@glxtrshortpl}[2][{}]{%
4277 \new@ifnextchar[{\@glxtrshortpl{#1}{#2}}{\@glxtrshortpl{#1}{#2}[]}%
4278 }

```

Read in the final optional argument:

```

4279 \def\@glxtrshortpl#1#2[#3]{%
4280 \gl@doifexists{#2}%
4281 {%
4282 \gl@setabbrvfmt{\gl@category{#2}}%
4283 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4284 \let\glxtrifwasfirstuse\@secondoftwo
4285 \let\gl@ifplural\@firstoftwo
4286 \let\gl@scapscase\@firstofthree
4287 \let\gl@insert\@empty
4288 \def\glscustomtext{%
4289 \gl@abbrvfont{\gl@accessshortpl{#2}\ifglxtrininsertinside#3\fi}%
4290 \ifglxtrininsertinside\else#3\fi
4291 }%
4292 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4293 }%
4294 \glspostlinkhook
4295 }

```

\Glsxtrshortpl

```

4296 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\ns@Glsxtrshortpl}
Define the un-starred form. Need to determine if there is a final optional argument
4297 \newcommand*{\ns@Glsxtrshortpl}[2][{}]{%
4298 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%

```


4299 }

Read in the final optional argument:

```
4300 \def\@GLSxtrshortpl#1#2[#3]{%
4301   \glsdoifexists{#2}%
4302   {%
4303     \glssetabbrvfmt{\glscategory{#2}}%
4304     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4305     \let\glxtrifwasfirstuse\@secondoftwo
4306     \let\glsifplural\@firstoftwo
4307     \let\glscapscase\@secondofthree
4308     \let\glsinsert\@empty
4309     \def\glscustomtext{%
4310       \glsabbrvfont{\glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
4311       \ifglxtrininsertinside\else#3\fi
4312     }%
4313     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4314   }%
4315   \glspostlinkhook
4316 }
```

\GLSxtrshortpl

```
4317 \newrobustcmd*{\GLSxtrshortpl}{\@gl@hyp@opt\@ns@GLSxtrshortpl}
```

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

```
4318 \newcommand*{\ns@GLSxtrshortpl}[2][ ]{%
4319   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[ ]}%
4320 }
```

Read in the final optional argument:

```
4321 \def\@GLSxtrshortpl#1#2[#3]{%
4322   \glsdoifexists{#2}%
4323   {%
4324     \glssetabbrvfmt{\glscategory{#2}}%
4325     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4326     \let\glxtrifwasfirstuse\@secondoftwo
4327     \let\glsifplural\@firstoftwo
4328     \let\glscapscase\@thirdofthree
4329     \let\glsinsert\@empty
4330     \def\glscustomtext{%
4331       \mfirstucMakeUppercase
4332       {\glsabbrvfont{\glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
4333       \ifglxtrininsertinside\else#3\fi
4334     }%
4335     }%
4336     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4337   }%
4338   \glspostlinkhook
4339 }
```

Plural long forms:

\glxstrlongpl

```
4340 \newrobustcmd*{\glxstrlongpl}{\@gls@hyp@opt\@ns@glxstrlongpl}
```

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

```
4341 \newcommand*{\ns@glxstrlongpl}[2] [] {%
```

```
4342   \new@ifnextchar[{\@glxstrlongpl{#1}{#2}}{\@glxstrlongpl{#1}{#2} []}%
```

```
4343 }
```

Read in the final optional argument:

```
4344 \def\@glxstrlongpl#1#2[#3] {%
```

```
4345   \glsdoifexists{#2}%
```

```
4346   {%
```

```
4347     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4348     \let\glxstrifwasfirstuse\@secondoftwo
```

```
4349     \let\glsifplural\@firstoftwo
```

```
4350     \let\glscapscase\@firstofthree
```

```
4351     \let\glsinsert\@empty
```

```
4352     \def\glscustomtext{%
```

```
4353       \glslongfont{\glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
```

```
4354       \ifglxtrininsertinside\else#3\fi
```

```
4355     }%
```

```
4356     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
4357   }%
```

```
4358   \glspostlinkhook
```

```
4359 }
```

\Glsxtrlongpl

```
4360 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
```

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

```
4361 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
```

```
4362   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
```

```
4363 }
```

Read in the final optional argument:

```
4364 \def\@Glsxtrlongpl#1#2[#3] {%
```

```
4365   \glsdoifexists{#2}%
```

```
4366   {%
```

```
4367     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4368     \let\glxstrifwasfirstuse\@secondoftwo
```

```
4369     \let\glsifplural\@firstoftwo
```

```
4370     \let\glscapscase\@secondofthree
```

```
4371     \let\glsinsert\@empty
```

```
4372     \def\glscustomtext{%
```

```
4373       \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
```

```
4374       \ifglxtrininsertinside\else#3\fi
```

```
4375     }%
```

```
4376     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
4377   }%
```

```
4378   \glspostlinkhook
```

```
4379 }
```

```

\GLSxtrlongpl
4380 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
4381 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
4382   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
4383 }

    Read in the final optional argument:
4384 \def\@GLSxtrlongpl#1#2[#3] {%
4385   \glsdoifexists{#2}%
4386   {%
4387     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4388     \let\glsxtrifwasfirstuse\@secondoftwo
4389     \let\glsifplural\@firstoftwo
4390     \let\glscapscase\@thirdofthree
4391     \let\glsinsert\@empty
4392     \def\glscustomtext{%
4393       \mfirstucMakeUppercase
4394       {\glslongfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
4395       \ifglsxtrininsertinside\else#3\fi
4396     }%
4397   }%
4398   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4399   }%
4400   \glspostlinkhook
4401 }

\glssetabbrvfmt   Set the current format for the given category (or the abbreviation category if unset).
4402 \newcommand*{\glssetabbrvfmt}[1] {%
4403   \ifcsdef{\glsabbrv@current@#1}%
4404   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
4405   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
4406 }

sxtrgenabbrvfmt   Similar to \glsacfm, but for abbreviations.
4407 \newcommand*{\glsxtrgenabbrvfmt}{%
4408   \ifdefempty\glscustomtext
4409   {%
4410     \ifglsused\glslabel
4411     {%
4412       \glsifplural
4413       {%
4414         \glscapscase
4415         {%

```

Subsequent plural form, don't adjust case:

```
4416      \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
4417      }%
4418      {%
```

Subsequent plural form, make first letter upper case:

```
4419      \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
4420      }%
4421      {%
```

Subsequent plural form, all caps:

```
4422      \mfirstucMakeUppercase
4423      {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
4424      }%
4425      }%
4426      {%
```

Subsequent singular form

```
4427      \glscapscase
4428      {%
```

Subsequent singular form, don't adjust case:

```
4429      \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
4430      }%
4431      {%
```

Subsequent singular form, make first letter upper case:

```
4432      \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
4433      }%
4434      {%
```

Subsequent singular form, all caps:

```
4435      \mfirstucMakeUppercase
4436      {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
4437      }%
4438      }%
4439      }%
4440      {%
```

First use:

```
4441      \glsifplural
4442      {%
```

First use plural form:

```
4443      \glscapscase
4444      {%
```

First use plural form, don't adjust case:

```
4445      \glstrfullplformat{\glslabel}{\glsinsert}%
4446      }%
4447      {%
```

First use plural form, make first letter upper case:

```
4448      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
4449      }%
4450      {%
```

First use plural form, all caps:

```
4451      \mfirstucMakeUppercase
4452      {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
4453      }%
4454      }%
4455      {%
```

First use singular form

```
4456      \glscapscase
4457      {%
```

First use singular form, don't adjust case:

```
4458      \glsxtrfullformat{\glslabel}{\glsinsert}%
4459      }%
4460      {%
```

First use singular form, make first letter upper case:

```
4461      \Glsxtrfullformat{\glslabel}{\glsinsert}%
4462      }%
4463      {%
```

First use singular form, all caps:

```
4464      \mfirstucMakeUppercase
4465      {\glsxtrfullformat{\glslabel}{\glsinsert}}%
4466      }%
4467      }%
4468      }%
4469      }%
4470      {%
```

User supplied text.

```
4471      \glscustomtext
4472      }%
4473 }
```

1.6.1 Abbreviation Styles Setup

breivationstyle

```
4474 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
4475   \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
4476   {%
4477     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
4478   }%
4479   {%
```

Have abbreviations already been defined for this category?

```

4480 \ifcsstring{@glsabbrv@current@#1}{#2}%
4481 {%
    Style already set.
4482 }%
4483 {%
4484 \def\@glsxtr@dostylewarn{%
4485 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
4486 {%
4487 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
4488 style has been switched \MessageBreak
4489 for category ‘#1’, \MessageBreak
4490 but there have already been entries \MessageBreak
4491 defined for this category. Unwanted \MessageBreak
4492 side-effects may result}}%
4493 \@endfortrue
4494 }%
4495 \@glsxtr@dostylewarn
    Set up the style for the given category.
4496 \csdef{@glsabbrv@current@#1}{#2}%
4497 \glsxtr@applyabbrvstyle{#2}%
4498 }%
4499 }%
4500 }

```

`\glsxtr@applyabbrvstyle` Apply the abbreviation style without existence check.

```

4501 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
4502 \csuse{@glsabbrv@dispstyle@setup@#1}%
4503 \csuse{@glsabbrv@dispstyle@fmts@#1}%
4504 }

```

`\glsxtr@applyabbrvfmt` Only apply the style formats.

```

4505 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
4506 \csuse{@glsabbrv@dispstyle@fmts@#1}%
4507 }

```

`\glsxtr@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.

```

4508 \newcommand*{\newabbreviationstyle}[3]{%
4509 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
4510 {%
4511 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
4512 defined}{}%
4513 }%
4514 {%
4515 \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```
4516 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
4517 #2}%
4518 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
4519 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
4520 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
4521 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
4522 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
4523 #3}%
4524 }%
4525 }
```

breivationstyle

```
4526 \newcommand*{\renewabbreviationstyle}[3]{%
4527 \ifcsundef{@glsabbrv@dispstyle@setup@#1}
4528 {%
4529 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
4530 }%
4531 {%
4532 \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
4533 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
4534 #2}%
4535 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
4536 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
4537 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
4538 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
4539 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
4540 #3}%
4541 }%
4542 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
4543 \newcommand*{\letabbreviationstyle}[2]{%
4544 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
4545 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4546 }
```

ecated@abbrstyle

```
\glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

4547 \newcommand*{\@GlsXtr@deprecated@abbrstyle}[2]{%
4548   \csdef{@glsabbrv@dispstyle@setup@#1}{%
4549     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
4550     \csuse{@glsabbrv@dispstyle@setup@#2}%
4551   }%
4552   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
4553 }

```

ecatedAbbrStyle Generate warning for deprecated style use.

```

4554 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
4555   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
4556   use ‘#2’ instead}%
4557 }

```

eAbbrStyleSetup

```

4558 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
4559   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
4560   {%
4561     \PackageError{glossaries-extra}%
4562     {Unknown abbreviation style definitions ‘#1’}{}%
4563   }%
4564   {%
4565     \csname @glsabbrv@dispstyle@setup@#1\endcsname
4566   }%
4567 }

```

seAbbrStyleFmts

```

4568 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
4569   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
4570   {%
4571     \PackageError{glossaries-extra}%
4572     {Unknown abbreviation style formats ‘#1’}{}%
4573   }%
4574   {%
4575     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
4576   }%
4577 }

```

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.

```
4578 \newif\ifglxtrinsertinside
4579 \glxtrinsertinsidefalse
```

long-short

```
4580 \newabbreviationstyle{long-short}%
4581 {%
4582   \renewcommand*{\CustomAbbreviationFields}{%
4583     name={\protect\glsabbrvfont{\the\glsshorttok}},
4584     sort={\the\glsshorttok},
4585     first={\protect\glsfirstlongfont{\the\glslongtok}%
4586       \protect\glxtrfullsep{\the\glslabeltok}%
4587       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4588     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4589       \protect\glxtrfullsep{\the\glslabeltok}%
4590       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
4591     plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
4592     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
4593 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4594   \glshasattribute{\the\glslabeltok}{regular}%
4595   {%
4596     \glissetattribute{\the\glslabeltok}{regular}{false}%
4597   }%
4598   {%
4599   }%
4600 }%
4601 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4602 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4603 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4604 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4605 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4606 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
4607 \renewcommand*{\glxtrfullformat}[2]{%
4608   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4609   \ifglxtrinsertinside\else##2\fi
4610   \glxtrfullsep{##1}%
4611   (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4612 }%
4613 \renewcommand*{\glxtrfullplformat}[2]{%
4614   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4615   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4616   (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4617 }%
```

```

4618 \renewcommand*{\Glsxtrfullformat}[2]{%
4619   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
4620   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4621   (\glsfirstabbrvfont{\glsaccessshort{##1}})%
4622 }%
4623 \renewcommand*{\Glsxtrfullplformat}[2]{%
4624   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
4625   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4626   (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
4627 }%
4628 }

```

Set this as the default style for general abbreviations:

```

4629 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

4630 \newcommand*{\glxtrlongshortdescsort}{\the\glslongtok\space(\the\glsshorttok)}

```

long-short-desc User supplies description. The long form is included in the name.

```

4631 \newabbreviationstyle{long-short-desc}%
4632 {%
4633   \renewcommand*{\CustomAbbreviationFields}{%
4634     name={\protect\glxtrfullformat{\the\glslabeltok}{}},
4635     sort={\glxtrlongshortdescsort},%
4636     first={\protect\glsfirstlongfont{\the\glslongtok}%
4637       \protect\glxtrfullsep{\the\glslabeltok}%
4638       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
4639     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
4640       \protect\glxtrfullsep{\the\glslabeltok}%
4641       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%

```

The text key should only have the short form.

```

4642     text={\protect\glsabbrvfont{\the\glsshorttok}},%
4643     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
4644   }%

```

Unset the regular attribute if it has been set.

```

4645 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4646   \glshasattribute{\the\glslabeltok}{regular}%
4647   {%
4648     \glissetattribute{\the\glslabeltok}{regular}{false}%
4649   }%
4650   {}%
4651 }%
4652 }%
4653 {%
4654   \GlsXtrUseAbbrStyleFmts{long-short}%
4655 }

```

short-long Short form followed by long form in parenthesis on first use.

```
4656 \newabbreviationstyle{short-long}%
4657 {%
4658   \renewcommand*{\CustomAbbreviationFields}{%
4659     name={\protect\glsabbrvfont{\the\glsshorttok}},
4660     sort={\the\glsshorttok},
4661     description={\the\glslongtok},%
4662     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
4663     \protect\glsxtrfullsep{\the\glslabeltok}%
4664     (\protect\glsfirstlongfont{\the\glslongtok}}),%
4665     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
4666     \protect\glsxtrfullsep{\the\glslabeltok}%
4667     (\protect\glsfirstlongfont{\the\glslongpltok}}),%
4668     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
4669 }
```

Unset the regular attribute if it has been set.

```
4669 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4670   \glshasattribute{\the\glslabeltok}{regular}%
4671   {%
4672     \glissetattribute{\the\glslabeltok}{regular}{false}%
4673   }%
4674   {}%
4675 }%
4676 }%
4677 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4678 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4679 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4680 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4681 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4682 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
4683 \renewcommand*{\glsxtrfullformat}[2]{%
4684   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4685   \ifglsxtrininsertinside\else##2\fi
4686   \glsxtrfullsep{##1}%
4687   (\glsfirstlongfont{\glsaccesslong{##1}})%
4688 }%
4689 \renewcommand*{\glsxtrfullplformat}[2]{%
4690   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
4691   \ifglsxtrininsertinside\else##2\fi
4692   \glsxtrfullsep{##1}%
4693   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4694 }%
4695 \renewcommand*{\Glsxtrfullformat}[2]{%
4696   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4697   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4698   (\glsfirstlongfont{\Glsaccesslong{##1}})%
4699 }
```

```

4699 }%
4700 \renewcommand*{\Glsxtrfullplformat}[2]{%
4701   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4702   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4703   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
4704 }%
4705 }

```

short-long-desc User supplies description. The long form is included in the name.

```

4706 \newabbreviationstyle{short-long-desc}%
4707 {%
4708   \renewcommand*{\CustomAbbreviationFields}{%
4709     name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
4710     sort={\the\glsshorttok},%
4711     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
4712       \protect\glsxtrfullsep{\the\glslabeltok}%
4713       (\protect\glsfirstlongfont{\the\glslongtok})},%
4714     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
4715       \protect\glsxtrfullsep{\the\glslabeltok}%
4716       (\protect\glsfirstlongfont{\the\glslongpltok})},%
4717     text={\protect\glsabbrvfont{\the\glsshorttok}},%
4718     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
4719   }%

```

Unset the regular attribute if it has been set.

```

4720 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4721   \glshasattribute{\the\glslabeltok}{regular}%
4722   {%
4723     \glssetattribute{\the\glslabeltok}{regular}{false}%
4724   }%
4725   {}%
4726 }%
4727 }%
4728 {%
4729   \GlsXtrUseAbbrStyleFmts{short-long}%
4730 }

```

ongfootnotefont Only used by the “footnote” styles.

```

4731 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{##1}}%

```

ongfootnotefont Only used by the “footnote” styles.

```

4732 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{##1}}%

```

xtrabbrvfootnote `\glsxtrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument *<long>* includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
4733 \newcommand*{\glxtrabbrvfootnote}[2]{\footnote{#2}}
```

footnote Short form followed by long form in footnote on first use.

```
4734 \newabbreviationstyle{footnote}%
4735 {%
4736   \renewcommand*{\CustomAbbreviationFields}{%
4737     name={\protect\glsabbrvfont{\the\glsshorttok}},
4738     sort={\the\glsshorttok},
4739     description={\the\glslongtok},%
4740     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
4741       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
4742       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
4743     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
4744       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
4745       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
4746     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
4747 }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
4747 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4748   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
4749   \glshasattribute{\the\glslabeltok}{regular}%
4750   {%
4751     \glssetattribute{\the\glslabeltok}{regular}{false}%
4752   }%
4753   {}%
4754 }%
4755 }%
4756 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
4757 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4758 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4759 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4760 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
4761 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
4762 \renewcommand*{\glxtrfullformat}[2]{%
4763   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
4764   \ifglxtrininsertinside\else##2\fi
4765   \protect\glxtrabbrvfootnote{##1}%
4766   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
4767 }%
4768 \renewcommand*{\glxtrfullplformat}[2]{%
```

```

4769 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4770 \ifglxtrinsertinside\else##2\fi
4771 \protect\glxtrabbrvfootnote{##1}%
4772 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
4773 }%
4774 \renewcommand*{\Glsxtrfullformat}[2]{%
4775 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4776 \ifglxtrinsertinside\else##2\fi
4777 \protect\glxtrabbrvfootnote{##1}%
4778 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
4779 }%
4780 \renewcommand*{\Glsxtrfullplformat}[2]{%
4781 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4782 \ifglxtrinsertinside\else##2\fi
4783 \protect\glxtrabbrvfootnote{##1}%
4784 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
4785 }%

```

The first use full form and the inline full form use the short (long) style.

```

4786 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4787 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4788 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4789 (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
4790 }%
4791 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4792 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4793 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4794 (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
4795 }%
4796 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4797 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4798 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4799 (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
4800 }%
4801 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4802 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4803 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4804 (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
4805 }%
4806 }

```

short-footnote

```
4807 \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. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```
4808 \newabbreviationstyle{postfootnote}%
4809 {%

```

```

4810 \renewcommand*{\CustomAbbreviationFields}{%
4811   name={\protect\glsabbrvfont{\the\glsshorttok}},
4812   sort={\the\glsshorttok},
4813   description={\the\glslongtok},%
4814   first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
4815   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
4816   plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

4817 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4818   \csdef{glsxtrpostlink\glscategorylabel}{%
4819     \glsxtrifwasfirstuse
4820     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

4821       \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
4822       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
4823   }%
4824   {}%
4825 }%
4826 \glsattribute{\the\glslabeltok}{regular}%
4827 {%
4828   \glssetattribute{\the\glslabeltok}{regular}{false}%
4829   }%
4830   {}%
4831 }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

4832 \renewcommand*{\glsxtrsetupfulldefs}{%
4833   \let\glsxtrifwasfirstuse\@secondoftwo
4834 }%
4835 }%
4836 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4837 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4838 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4839 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4840 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
4841 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

4842 \renewcommand*{\glsxtrfullformat}[2]{%
4843   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
4844   \ifglsxtrininsertinside\else##2\fi
4845 }%
4846 \renewcommand*{\glsxtrfullplformat}[2]{%
4847   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

4848 \ifglxtrinsertinside\else##2\fi
4849 }%
4850 \renewcommand*{\Glsxtrfullformat}[2]{%
4851 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4852 \ifglxtrinsertinside\else##2\fi
4853 }%
4854 \renewcommand*{\Glsxtrfullplformat}[2]{%
4855 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4856 \ifglxtrinsertinside\else##2\fi
4857 }%

```

The first use full form and the inline full form use the short (long) style.

```

4858 \renewcommand*{\glxtrinlinefullformat}[2]{%
4859 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4860 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4861 (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
4862 }%
4863 \renewcommand*{\glxtrinlinefullplformat}[2]{%
4864 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4865 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4866 (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
4867 }%
4868 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4869 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4870 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4871 (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
4872 }%
4873 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4874 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4875 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4876 (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
4877 }%
4878 }

```

rt-postfootnote

```

4879 \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.

```

4880 \newabbreviationstyle{short}%
4881 {%
4882 \renewcommand*{\CustomAbbreviationFields}{%
4883 name={\protect\glsabbrvfont{\the\glsshorttok}},
4884 sort={\the\glsshorttok},
4885 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4886 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4887 text={\protect\glsabbrvfont{\the\glsshorttok}},

```



```

4888 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4889 description={\the\glslongtok}}%
4890 \renewcommand*\GlsXtrPostNewAbbreviation}{%
4891 \glssetattribute{\the\glslabeltok}{regular}{true}}%
4892 }%
4893 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4894 \renewcommand*\abbrvpluralsuffix{\glspluralsuffix}%
4895 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
4896 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
4897 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
4898 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

4899 \renewcommand*\glsxtrinlinefullformat[2]{%
4900 \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4901 \ifglsxtrininsertinside##2\fi}%
4902 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4903 (\glsfirstlongfont{\glsaccesslong{##1}})%
4904 }%
4905 \renewcommand*\glsxtrinlinefullplformat[2]{%
4906 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4907 \ifglsxtrininsertinside##2\fi}%
4908 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4909 (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4910 }%
4911 \renewcommand*\Glsxtrinlinefullformat[2]{%
4912 \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
4913 \ifglsxtrininsertinside##2\fi}%
4914 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4915 (\glsfirstlongfont{\Glsaccesslong{##1}})%
4916 }%
4917 \renewcommand*\Glsxtrinlinefullplformat[2]{%
4918 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
4919 \ifglsxtrininsertinside##2\fi}%
4920 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
4921 (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
4922 }%

```

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.

```

4923 \renewcommand*\glsxtrfullformat[2]{%
4924 \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
4925 \ifglsxtrininsertinside\else##2\fi
4926 }%
4927 \renewcommand*\glsxtrfullplformat[2]{%
4928 \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
4929 \ifglsxtrininsertinside\else##2\fi
4930 }%
4931 \renewcommand*\Glsxtrfullformat[2]{%

```

```

4932 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4933 \ifglxtrinsertinside\else##2\fi
4934 }%
4935 \renewcommand*{\Glsxtrfullplformat}[2]{%
4936 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4937 \ifglxtrinsertinside\else##2\fi
4938 }%
4939 }

```

Set this as the default style for acronyms:

```

4940 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

4941 \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.

```

4942 \newabbreviationstyle{short-desc}%
4943 {%
4944 \renewcommand*{\CustomAbbreviationFields}{%
4945 name={\protect\glsxtrinlinefullformat{\the\glslabeltok}{}},
4946 sort={\the\glsshorttok},
4947 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
4948 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
4949 text={\protect\glsabbrvfont{\the\glsshorttok}},
4950 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
4951 description={\the\glslongtok}}%
4952 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
4953 \glssetattribute{\the\glslabeltok}{regular}{true}}%
4954 }%
4955 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

4956 \renewcommand*{\abbrvpluralsuffix}{\glspluralsuffix}%
4957 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
4958 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
4959 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
4960 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

4961 \renewcommand*{\glsxtrinlinefullformat}[2]{%
4962 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4963 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4964 (\glsfirstlongfont{\glsaccesslong{##1}})%
4965 }%
4966 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
4967 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4968 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
4969 (\glsfirstlongfont{\glsaccesslongpl{##1}})%
4970 }%

```

```

4971 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
4972   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4973   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4974   (\glsfirstlongfont{\Glsaccesslong{##1}})%
4975 }%
4976 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
4977   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4978   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
4979   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
4980 }%

```

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.

```

4981 \renewcommand*{\glxtrfullformat}[2]{%
4982   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4983   \ifglxtrinsertinside\else##2\fi
4984 }%
4985 \renewcommand*{\glxtrfullplformat}[2]{%
4986   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4987   \ifglxtrinsertinside\else##2\fi
4988 }%
4989 \renewcommand*{\Glsxtrfullformat}[2]{%
4990   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
4991   \ifglxtrinsertinside\else##2\fi
4992 }%
4993 \renewcommand*{\Glsxtrfullplformat}[2]{%
4994   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
4995   \ifglxtrinsertinside\else##2\fi
4996 }%
4997 }

```

ort-nolong-desc

```
4998 \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.

```

4999 \newabbreviationstyle{long-desc}%
5000 {%
5001   \renewcommand*{\CustomAbbreviationFields}{%
5002     name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
5003     sort={\the\glslongtok},
5004     first={\protect\glsfirstlongfont{\the\glslongtok}},
5005     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
5006     text={\the\glslongtok},
5007     plural={\the\glslongpltok}%
5008   }%
5009   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5010     \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

5011 }%
 5012 {%

In case the user wants to mix and match font styles, these are redefined here.

```
5013 \renewcommand*\abbrvpluralsuffix{\glspluralsuffix}%
5014 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
5015 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
5016 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
5017 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
5018 \renewcommand*\glsxtrinlinefullformat[2]{%
5019   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
5020   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5021   (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
5022 }%
5023 \renewcommand*\glsxtrinlinefullplformat[2]{%
5024   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
5025   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5026   (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
5027 }%
5028 \renewcommand*\Glsxtrinlinefullformat[2]{%
5029   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
5030   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5031   (\protect\glsfirstabbrvfont{\Glsaccessshort{##1}})%
5032 }%
5033 \renewcommand*\Glsxtrinlinefullplformat[2]{%
5034   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
5035   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5036   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}})%
5037 }%
```

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.

```
5038 \renewcommand*\glsxtrfullformat[2]{%
5039   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
5040   \ifglsxtrinsertinside\else##2\fi
5041 }%
5042 \renewcommand*\glsxtrfullplformat[2]{%
5043   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
5044   \ifglsxtrinsertinside\else##2\fi
5045 }%
5046 \renewcommand*\Glsxtrfullformat[2]{%
5047   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
5048   \ifglsxtrinsertinside\else##2\fi
5049 }%
5050 \renewcommand*\Glsxtrfullplformat[2]{%
5051   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
5052   \ifglsxtrinsertinside\else##2\fi
5053 }%
5054 }
```

ng-noshort-desc Provide a synonym that matches similar styles.

```
5055 \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.

```
5056 \newabbreviationstyle{long}%
5057 {%
5058   \renewcommand*{\CustomAbbreviationFields}{%
5059     name={\protect\glsabbrvfont{\the\glsshorttok}},
5060     sort={\the\glsshorttok},
5061     first={\protect\glsfirstlongfont{\the\glslongtok}},
5062     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
5063     text={\the\glslongtok},
5064     plural={\the\glslongpltok},%
5065     description={\the\glslongtok}%
5066   }%
5067   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5068     \glssetattribute{\the\glslabeltok}{regular}{true}}%
5069 }%
5070 {%
5071   \GlsXtrUseAbbrStyleFmts{long-desc}%
5072 }
```

long-noshort Provide a synonym that matches similar styles.

```
5073 \letabbreviationstyle{long-noshort}{long}
```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

\glsxtrscfont

```
5074 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

sxtrfirstscfont

```
5075 \newcommand*{\glsxtrfirstscfont}[1]{\glsxtrscfont{#1}}
```

and for the default short form suffix:

\glsxtrscsuffix

```
5076 \newcommand*{\glsxtrscsuffix}{\glstextup{\glspluralsuffix}}
```

long-short-sc

```
5077 \newabbreviationstyle{long-short-sc}%
5078 {%
5079   \GlsXtrUseAbbrStyleSetup{long-short}%
5080 }%
5081 {%
```

Mostly as long-short style:

```
5082 \GlsXtrUseAbbrStyleFmts{long-short}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5083 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5084 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
5085 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
5086 }
```

g-short-sc-desc

```
5087 \newabbreviationstyle{long-short-sc-desc}%
5088 {%
5089 \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5090 }%
5091 {%
```

Mostly as long-short-desc style:

```
5092 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5093 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5094 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
5095 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
5096 }
```

Now the short (long) version

```
5097 \newabbreviationstyle{short-sc-long}%
5098 {%
5099 \GlsXtrUseAbbrStyleSetup{short-long}%
5100 }%
5101 {%
```

Mostly as short-long style:

```
5102 \GlsXtrUseAbbrStyleFmts{short-long}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5103 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5104 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
5105 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
5106 }
```

As before but user provides description

```
5107 \newabbreviationstyle{short-sc-long-desc}%
5108 {%
5109 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5110 }%
5111 {%
```

Mostly as short-long-desc style:

```
5112 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5113 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5114 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
5115 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
5116 }
```

short-sc

```
5117 \newabbreviationstyle{short-sc}%
5118 {%
5119 \GlsXtrUseAbbrStyleSetup{short-nolong}%
5120 }%
5121 {%
```

Mostly as short style:

```
5122 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5123 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5124 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
5125 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
5126 }
```

short-sc-nolong

```
5127 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
5128 \newabbreviationstyle{short-sc-desc}%
5129 {%
5130 \GlsXtrUseAbbrStyleSetup{short-desc}%
5131 }%
5132 {%
```

Mostly as short style:

```
5133 \GlsXtrUseAbbrStyleFmts{short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5134 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5135 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
5136 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
5137 }
```

-sc-nolong-desc

```
5138 \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 \glsshort.

```
5139 \newabbreviationstyle{long-noshort-sc}%
5140 {%
5141 \GlsXtrUseAbbrStyleSetup{long-noshort}%
5142 }%
5143 {%
```

Mostly as long style:

```
5144 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5145 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5146 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
5147 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
5148 }
```

long-sc Backward compatibility:

```
5149 \@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 \glsshort.

```
5150 \newabbreviationstyle{long-noshort-sc-desc}%
5151 {%
5152 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
5153 }%
5154 {%
```

Mostly as long style:

```
5155 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5156 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5157 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
5158 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
5159 }
```

long-desc-sc Backward compatibility:

```
5160 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
5161 \newabbreviationstyle{short-sc-footnote}%
5162 {%
5163 \GlsXtrUseAbbrStyleSetup{short-footnote}%
5164 }%
5165 {%
```

Mostly as long style:

```
5166 \GlsXtrUseAbbrStyleFmts{short-footnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5167 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
5168 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
5169 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
5170 }
```

footnote-sc Backward compatibility:

```
5171 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```


sc-postfootnote

```
5172 \newabbreviationstyle{short-sc-postfootnote}%
5173 {%
5174   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
5175 }%
5176 {%
```

Mostly as long style:

```
5177   \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
5178   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrscsuffix}%
5179   \renewcommand*{\glsabbrvfont}[1]{\glstrscfont{##1}}%
5180   \renewcommand*{\glsfirstabbrvfont}[1]{\glstrfirstscfont{##1}}%
5181 }
```

postfootnote-sc Backward compatibility:

```
5182 \@glstr@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:

\glstrsmfont

```
5183 \newcommand*{\glstrsmfont}[1]{\textsmaller{#1}}
```

strfirstsmfont

```
5184 \newcommand*{\glstrfirstsmfont}[1]{\glstrsmfont{#1}}
```

and for the default short form suffix:

\glstrsmsuffix

```
5185 \newcommand*{\glstrsmsuffix}{\glspluralsuffix}
```

long-short-sm

```
5186 \newabbreviationstyle{long-short-sm}%
5187 {%
5188   \GlsXtrUseAbbrStyleSetup{long-short}%
5189 }%
5190 {%
```

Mostly as long-short style:

```
5191   \GlsXtrUseAbbrStyleFmts{long-short}%
5192   \renewcommand*{\glsabbrvfont}[1]{\glstrsmfont{##1}}%
5193   \renewcommand*{\glsfirstabbrvfont}[1]{\glstrfirstsmfont{##1}}%
5194   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
5195 }
```

g-short-sm-desc

```
5196 \newabbreviationstyle{long-short-sm-desc}%  
5197 {%  
5198   \GlsXtrUseAbbrStyleSetup{long-short-desc}%  
5199 }%  
5200 {%
```

Mostly as long-short-desc style:

```
5201   \GlsXtrUseAbbrStyleFmts{long-short-desc}%  
5202   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
5203   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
5204   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
5205 }
```

short-sm-long Now the short (long) version

```
5206 \newabbreviationstyle{short-sm-long}%  
5207 {%  
5208   \GlsXtrUseAbbrStyleSetup{short-long}%  
5209 }%  
5210 {%
```

Mostly as short-long style:

```
5211   \GlsXtrUseAbbrStyleFmts{short-long}%  
5212   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
5213   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
5214   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
5215 }
```

rt-sm-long-desc As before but user provides description

```
5216 \newabbreviationstyle{short-sm-long-desc}%  
5217 {%  
5218   \GlsXtrUseAbbrStyleSetup{short-long-desc}%  
5219 }%  
5220 {%
```

Mostly as short-long-desc style:

```
5221   \GlsXtrUseAbbrStyleFmts{short-long-desc}%  
5222   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
5223   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
5224   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
5225 }
```

short-sm

```
5226 \newabbreviationstyle{short-sm}%  
5227 {%  
5228   \GlsXtrUseAbbrStyleSetup{short-nolong}%  
5229 }%  
5230 {%
```

Mostly as short style:

```
5231 \GlsXtrUseAbbrStyleFmts{short-nolong}%
5232 \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
5233 \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
5234 \renewcommand*\abbrvpluralsuffix{\protect\glstrsmsuffix}%
5235 }
```

short-sm-nolong

```
5236 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
5237 \newabbreviationstyle{short-sm-desc}%
5238 {%
5239   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
5240 }%
5241 {%
```

Mostly as short style:

```
5242 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
5243 \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
5244 \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
5245 \renewcommand*\abbrvpluralsuffix{\protect\glstrsmsuffix}%
5246 }
```

-sm-nolong-desc

```
5247 \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.

```
5248 \newabbreviationstyle{long-noshort-sm}%
5249 {%
5250   \GlsXtrUseAbbrStyleSetup{long-noshort}%
5251 }%
5252 {%
```

Mostly as long style:

```
5253 \GlsXtrUseAbbrStyleFmts{long-noshort}%
5254 \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
5255 \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
5256 \renewcommand*\abbrvpluralsuffix{\protect\glstrsmsuffix}%
5257 }
```

long-sm Backward compatibility:

```
5258 \@glstr@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.

```
5259 \newabbreviationstyle{long-noshort-sm-desc}%
5260 {%
```

```

5261 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
5262 }%
5263 {%

```

Mostly as long style:

```

5264 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
5265 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
5266 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
5267 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
5268 }

```

long-desc-sm Backward compatibility:

```

5269 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

5270 \newabbreviationstyle{short-sm-footnote}%
5271 {%
5272 \GlsXtrUseAbbrStyleSetup{short-footnote}%
5273 }%
5274 {%

```

Mostly as long style:

```

5275 \GlsXtrUseAbbrStyleFmts{short-footnote}%
5276 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
5277 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
5278 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
5279 }

```

footnote-sm Backward compatibility:

```

5280 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

5281 \newabbreviationstyle{short-sm-postfootnote}%
5282 {%
5283 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
5284 }%
5285 {%

```

Mostly as long style:

```

5286 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
5287 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
5288 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
5289 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
5290 }

```

postfootnote-sm Backward compatibility:

```

5291 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.6.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```
5292 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%
```

`\glsfirstabbrvemfont`

```
5293 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%
```

`\glsfirstlongemfont` Only used by the “long-em” styles.

```
5294 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
5295 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`\long-short-em`

```
5296 \newabbreviationstyle{long-short-em}%
5297 {%
5298   \GlsXtrUseAbbrStyleSetup{long-short}%
5299 }%
5300 {%
```

Mostly as long-short style:

```
5301   \GlsXtrUseAbbrStyleFmts{long-short}%
5302   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5303 }
```

`\long-short-em-desc`

```
5304 \newabbreviationstyle{long-short-em-desc}%
5305 {%
5306   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5307 }%
5308 {%
```

Mostly as long-short-desc style:

```
5309   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
5310   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5311 }
```

`\long-em-short-em`

```
5312 \newabbreviationstyle{long-em-short-em}%
5313 {%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
5314   \renewcommand*{\CustomAbbreviationFields}{%
5315     name={\protect\glsabbrvfont{\the\glsshorttok}},
5316     sort={\the\glsshorttok},
5317     first={\protect\glsfirstlongfont{\the\glslongtok}}%
5318     \protect\glsextrfullsep{\the\glslabeltok}}%
```

```

5319     (\protect\glsfirstabbrvfont{\the\glsshorttok}}),%
5320     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
5321     \protect\glstrfullsep{\the\glslabeltok}%
5322     (\protect\glsfirstabbrvfont{\the\glsshortpltok}}),%
5323     plural={\protect\glsabbvfont{\the\glsshortpltok}},%
5324     description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

5325 \renewcommand*\GlsXtrPostNewAbbreviation{%
5326     \glshasattribute{\the\glslabeltok}{regular}%
5327     {%
5328         \glssetattribute{\the\glslabeltok}{regular}{false}%
5329     }%
5330     {}}%
5331 }%
5332 }%
5333 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5334 \GlsXtrUseAbbrStyleFmts{long-short}%
5335 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5336 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5337 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
5338 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
5339 }

```

m-short-em-desc

```

5340 \newabbreviationstyle{long-em-short-em-desc}%
5341 {%
5342     \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5343 }%
5344 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5345 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
5346 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5347 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5348 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
5349 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
5350 }

```

short-em-long Now the short (long) version

```

5351 \newabbreviationstyle{short-em-long}%
5352 {%
5353     \GlsXtrUseAbbrStyleSetup{short-long}%
5354 }%
5355 {%

```

Mostly as short-long style:

```

5356 \GlsXtrUseAbbrStyleFmts{short-long}%
5357 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%

```

```

5358 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5359 }

```

rt-em-long-desc As before but user provides description

```

5360 \newabbreviationstyle{short-em-long-desc}%
5361 {%
5362   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5363 }%
5364 {%

```

Mostly as short-long-desc style:

```

5365   \GlsXtrUseAbbrStyleFmts{short-long-desc}%
5366   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5367   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5368   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
5369   \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
5370 }

```

hort-em-long-em

```

5371 \newabbreviationstyle{short-em-long-em}%
5372 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

5373   \renewcommand*{\CustomAbbreviationFields}{%
5374     name={\protect\glsabbrvfont{\the\glsshorttok}},
5375     sort={\the\glsshorttok},
5376     description={\protect\glslongemfont{\the\glslongtok}},%
5377     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
5378     \protect\glsxtrfullsep{\the\glslabeltok}%
5379     (\protect\glsfirstlongfont{\the\glslongtok}}),%
5380     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
5381     \protect\glsxtrfullsep{\the\glslabeltok}%
5382     (\protect\glsfirstlongfont{\the\glslongpltok}}),%
5383     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

5384   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5385     \glshasattribute{\the\glslabeltok}{regular}%
5386     {%
5387       \glssetattribute{\the\glslabeltok}{regular}{false}%
5388     }%
5389     {}%
5390   }%
5391 }%
5392 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5393   \GlsXtrUseAbbrStyleFmts{short-long}%
5394   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5395   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5396   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%

```

```

5397 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
5398 }

```

em-long-em-desc

```

5399 \newabbreviationstyle{short-em-long-em-desc}%
5400 {%
5401 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5402 }%
5403 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5404 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
5405 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5406 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
5407 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
5408 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
5409 }

```

short-em

```

5410 \newabbreviationstyle{short-em}%
5411 {%
5412 \GlsXtrUseAbbrStyleSetup{short-nolong}%
5413 }%
5414 {%

```

Mostly as short style:

```

5415 \GlsXtrUseAbbrStyleFmts{short-nolong}%
5416 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5417 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
5418 }

```

short-em-nolong

```

5419 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

5420 \newabbreviationstyle{short-em-desc}%
5421 {%
5422 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
5423 }%
5424 {%

```

Mostly as short style:

```

5425 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
5426 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5427 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
5428 }

```

-em-nolong-desc

```

5429 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```


long-noshort-em The short form is explicitly invoked through commands like \glssshort.

```
5430 \newabbreviationstyle{long-noshort-em}%
5431 {%
5432   \GlsXtrUseAbbrStyleSetup{long-noshort}%
5433 }%
5434 {%
```

Mostly as long-noshort style:

```
5435   \GlsXtrUseAbbrStyleFmts{long-noshort}%
5436   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5437   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
5438 }
```

long-em Backward compatibility:

```
5439 \@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

g-em-noshort-em The short form is explicitly invoked through commands like \glssshort.

```
5440 \newabbreviationstyle{long-em-noshort-em}%
5441 {%
5442   \renewcommand*{\CustomAbbreviationFields}{%
5443     name={\protect\glsabbrvfont{\the\glssshorttok}},
5444     sort={\the\glssshorttok},
5445     first={\protect\glsfirstlongfont{\the\glslongtok}},
5446     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
5447     text={\the\glslongtok},
5448     plural={\the\glslongpltok},%
5449     description={\protect\glslongemfont{\the\glslongtok}}%
5450   }%
5451   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5452     \glssetattribute{\the\glslabeltok}{regular}{true}}%
5453 }%
5454 {%
```

Mostly as long-noshort style:

```
5455   \GlsXtrUseAbbrStyleFmts{long-noshort}%
5456   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
5457   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
5458   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
5459   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
5460 }
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glssshort.

```
5461 \newabbreviationstyle{long-noshort-em-desc}%
5462 {%
5463   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
5464 }%
5465 {%
```

Mostly as long style:

```
5466 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
5467 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5468 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5469 }
```

long-desc-em Backward compatibility:

```
5470 \@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.

```
5471 \newabbreviationstyle{long-em-noshort-em-desc}%
5472 {%
5473 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
5474 }%
5475 }
```

Mostly as long style:

```
5476 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
5477 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5478 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5479 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
5480 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
5481 }
```

ort-em-footnote

```
5482 \newabbreviationstyle{short-em-footnote}%
5483 {%
5484 \GlsXtrUseAbbrStyleSetup{short-footnote}%
5485 }%
5486 }
```

Mostly as long style:

```
5487 \GlsXtrUseAbbrStyleFmts{short-footnote}%
5488 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5489 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5490 }
```

footnote-em Backward compatibility:

```
5491 \@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

```
5492 \newabbreviationstyle{short-em-postfootnote}%
5493 {%
5494 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
5495 }%
5496 }
```

Mostly as long style:

```
5497 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
5498 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
5499 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
5500 }
```

postfootnote-em Backward compatibility:

```
5501 \@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.

```
5502 \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.

```
5503 \ifdef\glscurrentfieldvalue
5504 {
5505   \newcommand*\glsxtruserparen[2]{%
5506     \glsxtrfullsep{#2}%
5507     (#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{})%
5508   }
5509 }
5510 {
5511   \newcommand*\glsxtruserparen[2]{%
5512     \glsxtrfullsep{#2}%
5513     (#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{})%
5514   }
5515 }
```

Font used for short form:

lssabbrvuserfont

```
5516 \newcommand*\glsabbrvuserfont[1]{#1}
```

Font used for short form on first use:

stabbrvuserfont

```
5517 \newcommand*\glsfirstabbrvuserfont[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
5518 \newcommand*\glslonguserfont[1]{#1}
```

Font used for long form on first use:

rstlonguserfont

```
5519 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
5520 \newcommand*{\glsxtrusersuffix}{\glspluralsuffix}
```

long-short-user

```
5521 \newabbreviationstyle{long-short-user}%
```

```
5522 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
5523 \renewcommand*{\CustomAbbreviationFields}{%
```

```
5524   name={\protect\glsabbrvfont{\the\glsshorttok}},
```

```
5525   sort={\the\glsshorttok},
```

```
5526   first={\protect\glsfirstlongfont{\the\glslongtok}%
```

```
5527     \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
```

```
5528   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
```

```
5529     \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}},
```

```
5530   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
```

```
5531   description={\protect\glslonguserfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
5532 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
5533   \glsasattribute{\the\glslabeltok}{regular}%
```

```
5534   {%
```

```
5535     \glssetattribute{\the\glslabeltok}{regular}{false}%
```

```
5536   }%
```

```
5537   {}%
```

```
5538 }%
```

```
5539 }%
```

```
5540 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
5541 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
```

```
5542 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
```

```
5543 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
```

```
5544 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
```

```
5545 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
5546 \renewcommand*{\glsxtrfullformat}[2]{%
```

```
5547   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
```

```
5548   \ifglsxtrinsertinside\else##2\fi
```

```
5549   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
```

```
5550 }%
```

```
5551 \renewcommand*{\glsxtrfullplformat}[2]{%
```

```
5552   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
```

```
5553   \ifglsxtrinsertinside\else##2\fi
```

```
5554   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
```

```

5555 }%
5556 \renewcommand*{\Glsxtrfullformat}[2]{%
5557   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
5558   \ifglxtrinsertinside\else##2\fi
5559   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
5560 }%
5561 \renewcommand*{\Glsxtrfullplformat}[2]{%
5562   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
5563   \ifglxtrinsertinside\else##2\fi
5564   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
5565 }%
5566 }

```

short-user-desc

```

5567 \newabbreviationstyle{long-short-user-desc}%
5568 {%
5569   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5570 }%
5571 {%
5572   \GlsXtrUseAbbrStyleFmts{long-short-user}%
5573 }

```

short-long-user

```

5574 \newabbreviationstyle{short-long-user}%
5575 {%
5576   \glslonguserfont is used in the description since \glsdesc doesn't set the style.
5577   \renewcommand*{\CustomAbbreviationFields}{%
5578     name={\protect\glsabbrvfont{\the\glsshorttok}},
5579     sort={\the\glsshorttok},
5580     description={\protect\glslonguserfont{\the\glslongtok}},%
5581     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
5582     \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}},%
5583     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
5584     \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
5585     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

5585 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5586   \glshasattribute{\the\glslabeltok}{regular}%
5587   {%
5588     \glssetattribute{\the\glslabeltok}{regular}{false}%
5589   }%
5590 }%
5591 }%
5592 }%
5593 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5594 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%

```

```

5595 \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
5596 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvuserfont{##1}}%
5597 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonguserfont{##1}}%
5598 \renewcommand*\glslongfont[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5599 \renewcommand*\glsxtrfullformat[2]{%
5600   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5601   \ifglsxtrininsertinside\else##2\fi
5602   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
5603 }%
5604 \renewcommand*\glsxtrfullplformat[2]{%
5605   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5606   \ifglsxtrininsertinside\else##2\fi
5607   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
5608 }%
5609 \renewcommand*\Glsxtrfullformat[2]{%
5610   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5611   \ifglsxtrininsertinside\else##2\fi
5612   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslong{##1}}}{##1}%
5613 }%
5614 \renewcommand*\Glsxtrfullplformat[2]{%
5615   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5616   \ifglsxtrininsertinside\else##2\fi
5617   \glsxtruserparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}{##1}%
5618 }%
5619 }

```

-long-user-desc

```

5620 \newabbreviationstyle{short-long-user-desc}%
5621 {%
5622   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5623 }%
5624 {%
5625   \GlsXtrUseAbbrStyleFmts{short-long-user}%
5626 }

```

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 `\glsabbrvfont`. 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 `\glxtrshort` 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:

```
5627 \let\@glxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
5628 \renewcommand*{\markright}[1]{%
```

```
5629 \glxtrmarkhook
```

```
5630 \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
```

```
5631 \glxtrrestoremarkhook
```

```
5632 }
```

`\markboth` Save original definition:

```
5633 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
5634 \renewcommand*{\markboth}[2]{%
```

```
5635 \glxtrmarkhook
```

```
5636 \@glxtr@org@markboth
```

```
5637 {\@glxtrinmark#1\@glxtrnotinmark}%
```

```
5638 {\@glxtrinmark#2\@glxtrnotinmark}%
```

```
5639 \glxtrrestoremarkhook
```

```
5640 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

`\glxtrRevertMarks`

```
5641 \newcommand*{\glxtrRevertMarks}{%
```

```
5642 \let\markright\@glxtr@org@markright
```

```
5643 \let\markboth\@glxtr@org@markboth
```

```
5644 }
```

`\glxtrifinmark`

```
5645 \newcommand*{\glxtrifinmark}[2]{#2}
```

`\@glxtrinmark`

```
5646 \newrobustcmd*{\@glxtrinmark}{%
```

```

5647 \let\glxstrifinmark\@firstoftwo
5648 }

```

glxstrnotinmark

```

5649 \newrobustcmd*{\@glxstrnotinmark}{%
5650 \let\glxstrifinmark\@secondoftwo
5651 }

```

\glxstrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```

5652 \newcommand*{\glxstrmarkhook}{%

```

Save current definitions:

```

5653 \let\@glxstr@org@MakeUppercase\MakeUppercase
5654 \let\@glxstr@org@glxstrtitleshort\glxstrtitleshort
5655 \let\@glxstr@org@glxstrtitleshortpl\glxstrtitleshortpl
5656 \let\@glxstr@org@Glsxtrtitleshort\Glsxtrtitleshort
5657 \let\@glxstr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
5658 \let\@glxstr@org@glxstrtitletext\glxstrtitletext
5659 \let\@glxstr@org@Glsxtrtitletext\Glsxtrtitletext
5660 \let\@glxstr@org@glxstrtitleplural\glxstrtitleplural
5661 \let\@glxstr@org@Glsxtrtitleplural\Glsxtrtitleplural
5662 \let\@glxstr@org@glxstrtitlefirst\glxstrtitlefirst
5663 \let\@glxstr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
5664 \let\@glxstr@org@glxstrtitlefirstplural\glxstrtitlefirstplural
5665 \let\@glxstr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
5666 \let\@glxstr@org@glxstrtitlelong\glxstrtitlelong
5667 \let\@glxstr@org@glxstrtitlelongpl\glxstrtitlelongpl
5668 \let\@glxstr@org@Glsxtrtitlelong\Glsxtrtitlelong
5669 \let\@glxstr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
5670 \let\@glxstr@org@glxstrtitlefull\glxstrtitlefull
5671 \let\@glxstr@org@glxstrtitlefullpl\glxstrtitlefullpl
5672 \let\@glxstr@org@Glsxtrtitlefull\Glsxtrtitlefull
5673 \let\@glxstr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

5674 \let\glxstrifinmark\@firstoftwo
5675 \let\MakeUppercase\MakeTextUppercase
5676 \let\glxstrtitleshort\glxstrheadshort
5677 \let\glxstrtitleshortpl\glxstrheadshortpl
5678 \let\Glsxtrtitleshort\Glsxtrheadshort
5679 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
5680 \let\glxstrtitletext\glxstrheadtext
5681 \let\Glsxtrtitletext\Glsxtrheadtext
5682 \let\glxstrtitleplural\glxstrheadplural
5683 \let\Glsxtrtitleplural\Glsxtrheadplural
5684 \let\glxstrtitlefirst\glxstrheadfirst
5685 \let\Glsxtrtitlefirst\Glsxtrheadfirst
5686 \let\glxstrtitlefirstplural\glxstrheadfirstplural
5687 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural

```



```

5688 \let\glxstrtitlelong\glxstrheadlong
5689 \let\glxstrtitlelongpl\glxstrheadlongpl
5690 \let\Glsxstrtitlelong\Glsxstrheadlong
5691 \let\Glsxstrtitlelongpl\Glsxstrheadlongpl
5692 \let\glxstrtitlefull\glxstrheadfull
5693 \let\glxstrtitlefullpl\glxstrheadfullpl
5694 \let\Glsxstrtitlefull\Glsxstrheadfull
5695 \let\Glsxstrtitlefullpl\Glsxstrheadfullpl
5696 }

```

`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.)

```

5697 \newcommand*{\glxstrrestoremarkhook}{%
5698   \let\glxstrifinmark\@secondoftwo
5699   \let\MakeUppercase\@glxstr@org@MakeUppercase
5700   \let\glxstrtitleshort\@glxstr@org@glxstrtitleshort
5701   \let\glxstrtitleshortpl\@glxstr@org@glxstrtitleshortpl
5702   \let\Glsxstrtitleshort\@glxstr@org@Glsxstrtitleshort
5703   \let\Glsxstrtitleshortpl\@glxstr@org@Glsxstrtitleshortpl
5704   \let\glxstrtitletext\@glxstr@org@glxstrtitletext
5705   \let\Glsxstrtitletext\@glxstr@org@Glsxstrtitletext
5706   \let\glxstrtitleplural\@glxstr@org@glxstrtitleplural
5707   \let\Glsxstrtitleplural\@glxstr@org@Glsxstrtitleplural
5708   \let\glxstrtitlefirst\@glxstr@org@glxstrtitlefirst
5709   \let\Glsxstrtitlefirst\@glxstr@org@Glsxstrtitlefirst
5710   \let\glxstrtitlefirstplural\@glxstr@org@glxstrtitlefirstplural
5711   \let\Glsxstrtitlefirstplural\@glxstr@org@Glsxstrtitlefirstplural
5712   \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
5713   \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
5714   \let\Glsxstrtitlelong\@glxstr@org@Glsxstrtitlelong
5715   \let\Glsxstrtitlelongpl\@glxstr@org@Glsxstrtitlelongpl
5716   \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
5717   \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
5718   \let\Glsxstrtitlefull\@glxstr@org@Glsxstrtitlefull
5719   \let\Glsxstrtitlefullpl\@glxstr@org@Glsxstrtitlefullpl
5720 }

```

Instead of using one document-wide conditional, use `headuc` attribute to determine whether or not to use the all upper case form.

`glxstrheadshort` Command used to display short form in the page header.

```

5721 \newcommand*{\glxstrheadshort}[1]{%
5722   \protect\NoCaseChange
5723   {%
5724     \gl@ifattribute{#1}{headuc}{true}%
5725     {%
5726       \GLSxtrshort[noindex,hyper=false]{#1}[]%

```

```

5727 }%
5728 {%
5729 \glxstrshort[noindex,hyper=false]{#1}[]%
5730 }%
5731 }%
5732 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

5733 \newrobustcmd*{\lsxtrtitleshort}[1]{%
5734 \glxstrshort[noindex,hyper=false]{#1}[]%
5735 }

```

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.

```

5736 \newcommand*{\lsxtrheadshortpl}[1]{%
5737 \protect\NoCaseChange
5738 {%
5739 \gl@ifattribute{#1}{headuc}{true}%
5740 {%
5741 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
5742 }%
5743 {%
5744 \glxstrshortpl[noindex,hyper=false]{#1}[]%
5745 }%
5746 }%
5747 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

5748 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
5749 \glxstrshortpl[noindex,hyper=false]{#1}[]%
5750 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

5751 \newcommand*{\Glsxtrheadshort}[1]{%
5752 \protect\NoCaseChange
5753 {%
5754 \gl@ifattribute{#1}{headuc}{true}%
5755 {%
5756 \GLSxtrshort[noindex,hyper=false]{#1}[]%
5757 }%
5758 {%
5759 \Glsxtrshort[noindex,hyper=false]{#1}[]%
5760 }%
5761 }%
5762 }

```

`\lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5763 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
5764   \Glsxtrshort[noindex,hyper=false]{#1}[]%
5765 }
```

`\sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
5766 \newcommand*{\Glsxtrheadshortpl}[1]{%
5767   \protect\NoCaseChange
5768   {%
5769     \glsifattribute{#1}{headuc}{true}%
5770     {%
5771       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
5772     }%
5773     {%
5774       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5775     }%
5776   }%
5777 }
```

`\xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5778 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
5779   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
5780 }
```

`\glsxtrheadtext` As above but for the text value.

```
5781 \newcommand*{\glsxtrheadtext}[1]{%
5782   \protect\NoCaseChange
5783   {%
5784     \glsifattribute{#1}{headuc}{true}%
5785     {%
5786       \GLStext[noindex,hyper=false]{#1}[]%
5787     }%
5788     {%
5789       \glstext[noindex,hyper=false]{#1}[]%
5790     }%
5791   }%
5792 }
```

`\glsxtrtitletext` Command to display text value in section title and table of contents.

```
5793 \newrobustcmd*{\glsxtrtitletext}[1]{%
5794   \glstext[noindex,hyper=false]{#1}[]%
5795 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
5796 \newcommand*{\Glsxtrheadtext}[1]{%
```

```

5797 \protect\NoCaseChange
5798 {%
5799   \glsifattribute{#1}{headuc}{true}%
5800   {%
5801     \GLStext[noindex,hyper=false]{#1}[]%
5802   }%
5803   {%
5804     \GLstext[noindex,hyper=false]{#1}[]%
5805   }%
5806 }%
5807 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

5808 \newrobustcmd*{\Glsxtrtitletext}[1]{%
5809   \GLstext[noindex,hyper=false]{#1}[]%
5810 }

```

lsxtrheadplural As above but for the plural value.

```

5811 \newcommand*{\lsxtrheadplural}[1]{%
5812   \protect\NoCaseChange
5813   {%
5814     \glsifattribute{#1}{headuc}{true}%
5815     {%
5816       \GLSplural[noindex,hyper=false]{#1}[]%
5817     }%
5818     {%
5819       \glsplural[noindex,hyper=false]{#1}[]%
5820     }%
5821   }%
5822 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

5823 \newrobustcmd*{\lsxtrtitleplural}[1]{%
5824   \glsplural[noindex,hyper=false]{#1}[]%
5825 }

```

lsxtrheadplural Convert first letter to upper case.

```

5826 \newcommand*{\Glsxtrheadplural}[1]{%
5827   \protect\NoCaseChange
5828   {%
5829     \glsifattribute{#1}{headuc}{true}%
5830     {%
5831       \GLSplural[noindex,hyper=false]{#1}[]%
5832     }%
5833     {%
5834       \Glsplural[noindex,hyper=false]{#1}[]%
5835     }%
5836   }%

```

5837 }

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
5838 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
5839   \Glsplural[noindex,hyper=false]{#1}[]%
5840 }
```

glxtrheadfirst As above but for the first value.

```
5841 \newcommand*{\glxtrheadfirst}[1]{%
5842   \protect\NoCaseChange
5843   {%
5844     \glsifattribute{#1}{headuc}{true}%
5845     {%
5846       \GLSfirst[noindex,hyper=false]{#1}[]%
5847     }%
5848     {%
5849       \glsfirst[noindex,hyper=false]{#1}[]%
5850     }%
5851   }%
5852 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
5853 \newrobustcmd*{\lsxtrtitlefirst}[1]{%
5854   \glsfirst[noindex,hyper=false]{#1}[]%
5855 }
```

Glsxtrheadfirst First letter converted to upper case

```
5856 \newcommand*{\Glsxtrheadfirst}[1]{%
5857   \protect\NoCaseChange
5858   {%
5859     \glsifattribute{#1}{headuc}{true}%
5860     {%
5861       \GLSfirst[noindex,hyper=false]{#1}[]%
5862     }%
5863     {%
5864       \Glsfirst[noindex,hyper=false]{#1}[]%
5865     }%
5866   }%
5867 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
5868 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
5869   \Glsfirst[noindex,hyper=false]{#1}[]%
5870 }
```

headfirstplural As above but for the firstplural value.

```

5871 \newcommand*{\glxtrheadfirstplural}[1]{%
5872   \protect\NoCaseChange
5873   {%
5874     \glusifattribute{#1}{headuc}{true}%
5875     {%
5876       \GLSfirstplural[noindex,hyper=false]{#1}[]%
5877     }%
5878     {%
5879       \glSfirstplural[noindex,hyper=false]{#1}[]%
5880     }%
5881   }%
5882 }

```

`\titlefirstplural` Command to display firstplural value in section title and table of contents.

```

5883 \newrobustcmd*{\glxtrtitlefirstplural}[1]{%
5884   \glSfirstplural[noindex,hyper=false]{#1}[]%
5885 }

```

`\headfirstplural` First letter converted to upper case

```

5886 \newcommand*{\Glsxtrheadfirstplural}[1]{%
5887   \protect\NoCaseChange
5888   {%
5889     \glusifattribute{#1}{headuc}{true}%
5890     {%
5891       \GLSfirstplural[noindex,hyper=false]{#1}[]%
5892     }%
5893     {%
5894       \glSfirstplural[noindex,hyper=false]{#1}[]%
5895     }%
5896   }%
5897 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

5898 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
5899   \glSfirstplural[noindex,hyper=false]{#1}[]%
5900 }

```

`\glxtrheadlong` Command used to display long form in the page header.

```

5901 \newcommand*{\glxtrheadlong}[1]{%
5902   \protect\NoCaseChange
5903   {%
5904     \glusifattribute{#1}{headuc}{true}%
5905     {%
5906       \GLSxtrlong[noindex,hyper=false]{#1}[]%
5907     }%
5908     {%
5909       \glSxtrlong[noindex,hyper=false]{#1}[]%
5910     }%

```

```
5911 }%
5912 }
```

`\glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
5913 \newrobustcmd*{\glxstrtitlelong}[1]{%
5914   \glxstrlong[noindex,hyper=false]{#1}[]%
5915 }
```

`\lsxtrheadlongpl` 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.

```
5916 \newcommand*{\lsxtrheadlongpl}[1]{%
5917   \protect\NoCaseChange
5918   {%
5919     \gl@ifattribute{#1}{headuc}{true}%
5920     {%
5921       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5922     }%
5923     {%
5924       \glxstrlongpl[noindex,hyper=false]{#1}[]%
5925     }%
5926   }%
5927 }
```

`\sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
5928 \newrobustcmd*{\sxstrtitlelongpl}[1]{%
5929   \glxstrlongpl[noindex,hyper=false]{#1}[]%
5930 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
5931 \newcommand*{\Glsxtrheadlong}[1]{%
5932   \protect\NoCaseChange
5933   {%
5934     \gl@ifattribute{#1}{headuc}{true}%
5935     {%
5936       \GLSxtrlong[noindex,hyper=false]{#1}[]%
5937     }%
5938     {%
5939       \Glsxtrlong[noindex,hyper=false]{#1}[]%
5940     }%
5941   }%
5942 }
```

`\Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5943 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
5944   \Glsxtrlong[noindex,hyper=false]{#1}[]%
5945 }
```

`\lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
5946 \newcommand*{\Glsxtrheadlongpl}[1]{%
5947   \protect\NoCaseChange
5948   {%
5949     \glsifattribute{#1}{headuc}{true}%
5950     {%
5951       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5952     }%
5953     {%
5954       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
5955     }%
5956   }%
5957 }
```

`\sxttitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
5958 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
5959   \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
5960 }
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
5961 \newcommand*{\glsxtrheadfull}[1]{%
5962   \protect\NoCaseChange
5963   {%
5964     \glsifattribute{#1}{headuc}{true}%
5965     {%
5966       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5967     }%
5968     {%
5969       \glsxtrfull[noindex,hyper=false]{#1}[]%
5970     }%
5971   }%
5972 }
```

`\glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
5973 \newrobustcmd*{\glsxtrtitlefull}[1]{%
5974   \glsxtrfull[noindex,hyper=false]{#1}[]%
5975 }
```

`\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.

```
5976 \newcommand*{\glsxtrheadfullpl}[1]{%
5977   \protect\NoCaseChange
5978   {%
5979     \glsifattribute{#1}{headuc}{true}%
5980     {%
```



```

5981      \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
5982    }%
5983    {%
5984      \glSxtrfullpl[noindex,hyper=false]{#1}[]%
5985    }%
5986  }%
5987 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

5988 \newrobustcmd*{\glSxtrtitlefullpl}[1]{%
5989   \glSxtrfullpl[noindex,hyper=false]{#1}[]%
5990 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

5991 \newcommand*{\Glsxtrheadfull}[1]{%
5992   \protect\NoCaseChange
5993   {%
5994     \glSifattribute{#1}{headuc}{true}%
5995     {%
5996       \GLSxtrfull[noindex,hyper=false]{#1}[]%
5997     }%
5998     {%
5999       \Glsxtrfull[noindex,hyper=false]{#1}[]%
6000     }%
6001   }%
6002 }

```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

6003 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
6004   \Glsxtrfull[noindex,hyper=false]{#1}[]%
6005 }

```

lSxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

6006 \newcommand*{\Glsxtrheadfullpl}[1]{%
6007   \protect\NoCaseChange
6008   {%
6009     \glSifattribute{#1}{headuc}{true}%
6010     {%
6011       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
6012     }%
6013     {%
6014       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
6015     }%
6016   }%
6017 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
6018 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
6019   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
6020 }
```

`\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.

```
6021 \ifdef\texorpdfstring
6022 {
6023   \newcommand*{\glsfmtshort}[1]{%
6024     \texorpdfstring
6025       {\Glsxtrtitleshort{#1}}%
6026       {\Glsentryshort{#1}}%
6027   }
6028 }
6029 {
6030   \newcommand*{\glsfmtshort}[1]{%
6031     \Glsxtrtitleshort{#1}}
6032 }
```

Similarly for the plural version.

`\glsfmtshortpl`

```
6033 \ifdef\texorpdfstring
6034 {
6035   \newcommand*{\glsfmtshortpl}[1]{%
6036     \texorpdfstring
6037       {\Glsxtrtitleshortpl{#1}}%
6038       {\Glsentryshortpl{#1}}%
6039   }
6040 }
6041 {
6042   \newcommand*{\glsfmtshortpl}[1]{%
6043     \Glsxtrtitleshortpl{#1}}
6044 }
```

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).

```
6045 \ifdef\texorpdfstring
6046 {
6047   \newcommand*{\Glsfmtshort}[1]{%
6048     \texorpdfstring
6049       {\Glsxtrtitleshort{#1}}%
6050       {\Glsentryshort{#1}}%
6051   }
6052 }
```

```

6053 {
6054   \newcommand*{\Glsfmtshort}[1]{%
6055     \Glsxtrtitleshort{#1}}
6056 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

6057 \ifdef\teorpdfstring
6058 {
6059   \newcommand*{\Glsfmtshortpl}[1]{%
6060     \teorpdfstring
6061     {\Glsxtrtitleshortpl{#1}}%
6062     {\glsentryshortpl{#1}}%
6063   }
6064 }
6065 {
6066   \newcommand*{\Glsfmtshortpl}[1]{%
6067     \Glsxtrtitleshortpl{#1}}
6068 }

```

`\glsfmttext` As above but for the text value.

```

6069 \ifdef\teorpdfstring
6070 {
6071   \newcommand*{\glsfmttext}[1]{%
6072     \teorpdfstring
6073     {\glsxtrtitletext{#1}}%
6074     {\glsentrytext{#1}}%
6075   }
6076 }
6077 {
6078   \newcommand*{\glsfmttext}[1]{%
6079     \glsxtrtitletext{#1}}
6080 }

```

`\Glsfmttext` First letter converted to upper case.

```

6081 \ifdef\teorpdfstring
6082 {
6083   \newcommand*{\Glsfmttext}[1]{%
6084     \teorpdfstring
6085     {\Glsxtrtitletext{#1}}%
6086     {\glsentrytext{#1}}%
6087   }
6088 }
6089 {
6090   \newcommand*{\Glsfmttext}[1]{%
6091     \Glsxtrtitletext{#1}}
6092 }

```

`\glsfmtplural` As above but for the plural value.

```

6093 \ifdef\teorpdfstring

```

```

6094 {
6095   \newcommand*{\glsfmtplural}[1]{%
6096     \texorpdfstring
6097     {\glsxtrtitleplural{#1}}%
6098     {\glsentryplural{#1}}%
6099   }
6100 }
6101 {
6102   \newcommand*{\glsfmtplural}[1]{%
6103     \glsxtrtitleplural{#1}}
6104 }

```

`\Glsfmtplural` First letter converted to upper case.

```

6105 \ifdef\texorpdfstring
6106 {
6107   \newcommand*{\Glsfmtplural}[1]{%
6108     \texorpdfstring
6109     {\Glsxtrtitleplural{#1}}%
6110     {\glsentryplural{#1}}%
6111   }
6112 }
6113 {
6114   \newcommand*{\Glsfmtplural}[1]{%
6115     \Glsxtrtitleplural{#1}}
6116 }

```

`\glsfmtfirst` As above but for the first value.

```

6117 \ifdef\texorpdfstring
6118 {
6119   \newcommand*{\glsfmtfirst}[1]{%
6120     \texorpdfstring
6121     {\glsxtrtitlefirst{#1}}%
6122     {\glsentryfirst{#1}}%
6123   }
6124 }
6125 {
6126   \newcommand*{\glsfmtfirst}[1]{%
6127     \glsxtrtitlefirst{#1}}
6128 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

6129 \ifdef\texorpdfstring
6130 {
6131   \newcommand*{\Glsfmtfirst}[1]{%
6132     \texorpdfstring
6133     {\Glsxtrtitlefirst{#1}}%
6134     {\glsentryfirst{#1}}%
6135   }
6136 }

```

```

6137 {
6138   \newcommand*{\Glsfmtfirst}[1]{%
6139     \Glsxtrtitlefirst{#1}}
6140 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

6141 \ifdef\texorpdfstring
6142 {
6143   \newcommand*{\glsfmtfirstpl}[1]{%
6144     \texorpdfstring
6145       {\Glsxtrtitlefirstplural{#1}}%
6146       {\Glsentryfirstplural{#1}}%
6147   }
6148 }
6149 {
6150   \newcommand*{\glsfmtfirstpl}[1]{%
6151     \Glsxtrtitlefirstplural{#1}}
6152 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

6153 \ifdef\texorpdfstring
6154 {
6155   \newcommand*{\Glsfmtfirstpl}[1]{%
6156     \texorpdfstring
6157       {\Glsxtrtitlefirstplural{#1}}%
6158       {\Glsentryfirstplural{#1}}%
6159   }
6160 }
6161 {
6162   \newcommand*{\Glsfmtfirstpl}[1]{%
6163     \Glsxtrtitlefirstplural{#1}}
6164 }

```

`\glsfmtlong` As above but for the long value.

```

6165 \ifdef\texorpdfstring
6166 {
6167   \newcommand*{\glsfmtlong}[1]{%
6168     \texorpdfstring
6169       {\Glsxtrtitlelong{#1}}%
6170       {\Glsentrylong{#1}}%
6171   }
6172 }
6173 {
6174   \newcommand*{\glsfmtlong}[1]{%
6175     \Glsxtrtitlelong{#1}}
6176 }

```

`\Glsfmtlong` First letter converted to upper case.

```

6177 \ifdef\texorpdfstring

```

```

6178 {
6179   \newcommand*{\Glsfmtlong}[1]{%
6180     \texorpdfstring
6181     {\Glsxtrtitlelong{#1}}%
6182     {\glsentrylong{#1}}%
6183   }
6184 }
6185 {
6186   \newcommand*{\Glsfmtlong}[1]{%
6187     \Glsxtrtitlelong{#1}}
6188 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

6189 \ifdef\texorpdfstring
6190 {
6191   \newcommand*{\glsfmtlongpl}[1]{%
6192     \texorpdfstring
6193     {\glsxtrtitlelongpl{#1}}%
6194     {\glsentrylongpl{#1}}%
6195   }
6196 }
6197 {
6198   \newcommand*{\glsfmtlongpl}[1]{%
6199     \glsxtrtitlelongpl{#1}}
6200 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

6201 \ifdef\texorpdfstring
6202 {
6203   \newcommand*{\Glsfmtlongpl}[1]{%
6204     \texorpdfstring
6205     {\Glsxtrtitlelongpl{#1}}%
6206     {\glsentrylongpl{#1}}%
6207   }
6208 }
6209 {
6210   \newcommand*{\Glsfmtlongpl}[1]{%
6211     \Glsxtrtitlelongpl{#1}}
6212 }

```

`\glsfmtfull` In-line full format.

```

6213 \ifdef\texorpdfstring
6214 {
6215   \newcommand*{\glsfmtfull}[1]{%
6216     \texorpdfstring
6217     {\glsxtrtitlefull{#1}}%
6218     {\glsxtrinlinefullformat{#1}{}}%
6219   }
6220 }

```

```

6221 {
6222   \newcommand*{\glsfmtfull}[1]{%
6223     \glsxtrtitlefull{#1}}
6224 }

```

\Glsfmtfull First letter converted to upper case.

```

6225 \ifdef\txorpdfstring
6226 {
6227   \newcommand*{\Glsfmtfull}[1]{%
6228     \txorpdfstring
6229     {\Glsxtrtitlefull{#1}}%
6230     {\Glsxtrinlinefullformat{#1}-{}}%
6231   }
6232 }
6233 {
6234   \newcommand*{\Glsfmtfull}[1]{%
6235     \Glsxtrtitlefull{#1}}
6236 }

```

\glsfmtfullpl In-line full plural format.

```

6237 \ifdef\txorpdfstring
6238 {
6239   \newcommand*{\glsfmtfullpl}[1]{%
6240     \txorpdfstring
6241     {\glsxtrtitlefullpl{#1}}%
6242     {\glsxtrinlinefullplformat{#1}-{}}%
6243   }
6244 }
6245 {
6246   \newcommand*{\glsfmtfullpl}[1]{%
6247     \glsxtrtitlefullpl{#1}}
6248 }

```

\Glsfmtfullpl First letter converted to upper case.

```

6249 \ifdef\txorpdfstring
6250 {
6251   \newcommand*{\Glsfmtfullpl}[1]{%
6252     \txorpdfstring
6253     {\Glsxtrtitlefullpl{#1}}%
6254     {\Glsxtrinlinefullplformat{#1}-{}}%
6255   }
6256 }
6257 {
6258   \newcommand*{\Glsfmtfullpl}[1]{%
6259     \Glsxtrtitlefullpl{#1}}
6260 }

```

1.8 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

sariesExtraLang

```
6261 \newcommand*{\RequireGlossariesExtraLang}[1]{%
6262   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
6263 }
```

sariesExtraLang

```
6264 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
6265   \ProvidesFile{glossariesxtr-#1.ldf}%
6266 }
```

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.)

```
6267 \@ifpackageloaded{tracklang}
6268 {%
6269   \AnyTrackedLanguages
6270   {%
6271     \ForEachTrackedDialect{\this@dialect}{%
6272       \IfTrackedLanguageFileExists{\this@dialect}%
6273       {glossariesxtr-}% prefix
6274       {.ldf}%
6275       {%
6276         \RequireGlossariesExtraLang{\CurrentTrackedTag}%
6277       }%
6278     }%
6279   }%
6280 }%
6281 }%
6282 {}%
6283 }
6284 {}
```

Load `glossaries-extra-stylemods` if required.

```
6285 \@glsxtr@redefstyles
```

and set the style:

```
6286 \@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:

```
6287 \NeedsTeXFormat{LaTeX2e}
6288 \ProvidesPackage{glossaries-extra-stylemods}[2016/12/17 v1.10 (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`

```
6289 \newcommand*{\@glxtr@loadstyles}{%
6290 \DeclareOption*{%
6291   \IfFileExists{glossary-\CurrentOption.sty}%
6292   {\eappto\@glxtr@loadstyles{%
6293     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
6294   {\PackageError{glossaries-extra-styles}%
6295     {Unknown option '\CurrentOption'}{}}
6296 }
```

Process the package options:

```
6297 \ProcessOptions
```

Load the required packages:

```
6298 \@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`

```
6299 \providecommand{\renewglossarystyle}[2]{%
6300   \ifcsundef{@glstyle@#1}%
6301   {%
6302     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%
```

```

6303 }%
6304 {%
6305   \csdef{@glsstyle@#1}{#2}%
6306 }%
6307 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```

6308 \ifcsdef{@glsstyle@listdotted}
6309 {%
6310   \renewglossarystyle{listdotted}{%
6311     \setglossarystyle{list}%
6312     \renewcommand*{\glossentry}[2]{%
6313       \item[]\makebox[\glslistdottedwidth][l]{%
6314         \glsentryitem{##1}%
6315         \glstarget{##1}{\glossentryname{##1}}%
6316         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
6317         \glossentrydesc{##1}\glspostdescription}%
6318     \renewcommand*{\subglossentry}[3]{%
6319       \item[]\makebox[\glslistdottedwidth][l]{%
6320         \glssubentryitem{##2}%
6321         \glstarget{##2}{\glossentryname{##2}}%
6322         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
6323         \glossentrydesc{##2}\glspostdescription}%
6324   }
6325 }
6326 {}

```

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.

```

6327 \ifcsdef{@glsstyle@long3col}
6328 {%
6329   \renewglossarystyle{long3col}{%
6330     \renewenvironment{theglossary}%
6331       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
6332       {\end{longtable}}%
6333     \renewcommand*{\glossaryheader}{}%
6334     \renewcommand*{\glsgroupheading}[1]{}%
6335     \renewcommand{\glossentry}[2]{%
6336       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6337       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline

```

```

6338 }%
6339 \renewcommand{\subglossentry}[3]{%
6340     &
6341     \glssubentryitem{##2}%
6342     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6343     ##3\tabularnewline
6344 }%
6345 \renewcommand*\{ \glsgroupskip }{%
6346     \ifglsgnoglobskip\else & & \tabularnewline\fi}%
6347 }
6348 }
6349 {}

```

Four column style:

```

6350 \ifcsdef{@glsstyle@long4col}
6351 {%
6352     \renewglossarystyle{long4col}{%
6353         \renewenvironment{theglossary}%
6354             {\begin{longtable}{llll}}%
6355             {\end{longtable}}%
6356         \renewcommand*\{ \glossaryheader }{%
6357             \renewcommand*\{ \glsgroupheading }[1]{%
6358                 \renewcommand{\glossentry}[2]{%
6359                     \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6360                     \glossentrydesc{##1}\glspostdescription &
6361                     \glossentrysymbol{##1} &
6362                     ##2\tabularnewline
6363                 }%
6364                 \renewcommand{\subglossentry}[3]{%
6365                     &
6366                     \glssubentryitem{##2}%
6367                     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6368                     \glossentrysymbol{##2} & ##3\tabularnewline
6369                 }%
6370                 \renewcommand*\{ \glsgroupskip }{%
6371                     \ifglsgnoglobskip\else & & \tabularnewline\fi}%
6372             }
6373         }
6374     }

```

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.

```

6375 \ifcsdef{@glsstyle@longragged3col}
6376 {%
6377     \renewglossarystyle{longragged3col}{%

```

```

6378 \renewenvironment{theglossary}%
6379     {\begin{longtable}{l>{\raggedright}p{\glsglscwidth}%
6380      >{\raggedright}p{\glspagelistwidth}}}%
6381     {\end{longtable}}}%
6382 \renewcommand*{\glossaryheader}{}%
6383 \renewcommand*{\glsgroupheading}[1]{}%
6384 \renewcommand{\glossentry}[2]{%
6385     \glsglscitem{##1}\glsglsc{##1}{\glossentryname{##1}} &
6386     \glossentrydesc{##1}\glsglscpostdescription & ##2\glsglscnewline
6387 }%
6388 \renewcommand{\subglossentry}[3]{%
6389     &
6390     \glsglscsubentryitem{##2}%
6391     \glsglsc{##2}{\strut}\glossentrydesc{##2}\glsglscpostdescription &
6392     ##3\glsglscnewline
6393 }%
6394 \renewcommand*{\glsgroupskip}{%
6395     \ifglsglscnogroupskip\else & \glsglscnewline\fi}%
6396 }
6397 }
6398 {}

```

Four column style:

```

6399 \ifcsdef{@glsglscstyle@altlongragged4col}
6400 {%
6401     \renewglossarystyle{altlongragged4col}{%
6402         \renewenvironment{theglossary}%
6403             {\begin{longtable}{l>{\raggedright}p{\glsglscwidth}l%
6404              >{\raggedright}p{\glspagelistwidth}}}%
6405             {\end{longtable}}}%
6406         \renewcommand*{\glossaryheader}{}%
6407         \renewcommand*{\glsgroupheading}[1]{}%
6408         \renewcommand{\glossentry}[2]{%
6409             \glsglscitem{##1}\glsglsc{##1}{\glossentryname{##1}} &
6410             \glossentrydesc{##1}\glsglscpostdescription & \glossentrysymbol{##1} &
6411             ##2\glsglscnewline
6412         }%
6413         \renewcommand{\subglossentry}[3]{%
6414             &
6415             \glsglscsubentryitem{##2}%
6416             \glsglsc{##2}{\strut}\glossentrydesc{##2}\glsglscpostdescription &
6417             \glossentrysymbol{##2} & ##3\glsglscnewline
6418         }%
6419         \renewcommand*{\glsgroupskip}{%
6420             \ifglsglscnogroupskip\else & & \glsglscnewline\fi}%
6421     }
6422 }
6423 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```

6424 \ifcsdef{@glsstyle@super3col}
6425 {%
6426   \renewglossarystyle{super3col}{%
6427     \renewenvironment{theglossary}%
6428       {\tablehead{}\tabletail{}}%
6429       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
6430       {\end{supertabular}}}%
6431   \renewcommand*{\glossaryheader}{}%
6432   \renewcommand*{\glsgroupheading}[1]{}%
6433   \renewcommand{\glossentry}[2]{%
6434     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6435     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
6436   }%
6437   \renewcommand{\subglossentry}[3]{%
6438     &
6439     \glssubentryitem{##2}%
6440     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6441     ##3\tabularnewline
6442   }%
6443   \renewcommand*{\glsgroupskip}{%
6444     \ifglsnogroupskip\else & &\tabularnewline\fi}%
6445 }
6446 }
6447 {}

```

Four column styles:

```

6448 \ifcsdef{@glsstyle@super4col}
6449 {%
6450   \renewglossarystyle{super4col}{%
6451     \renewenvironment{theglossary}%
6452       {\tablehead{}\tabletail{}}%
6453       \begin{supertabular}{llll}}}%
6454       \end{supertabular}}}%
6455   \renewcommand*{\glossaryheader}{}%
6456   \renewcommand*{\glsgroupheading}[1]{}%
6457   \renewcommand{\glossentry}[2]{%
6458     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6459     \glossentrydesc{##1}\glspostdescription &
6460     \glossentrysymbol{##1} & ##2\tabularnewline
6461   }%
6462   \renewcommand{\subglossentry}[3]{%
6463     &
6464     \glssubentryitem{##2}%
6465     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6466     \glossentrysymbol{##2} & ##3\tabularnewline
6467   }%
6468   \renewcommand*{\glsgroupskip}{%

```

```

6469     \ifglsnogroupskip\else & & \tabularnewline\fi}%
6470 }
6471 }
6472 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

6473 \ifcsdef{@glsstyle@superragged3col}
6474 {%
6475   \renewglossarystyle{superragged3col}{%
6476     \renewenvironment{theglossary}%
6477       {\tablehead{}\tabletail}%
6478       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
6479         >{\raggedright}p{\glspagelistwidth}}}%
6480       {\end{supertabular}}}%
6481   \renewcommand*{\glossaryheader}{}%
6482   \renewcommand*{\glsgroupheading}[1]{}%
6483   \renewcommand{\glossentry}[2]{%
6484     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6485     \glossentrydesc{##1}\glspostdescription &
6486     ##2\tabularnewline
6487   }%
6488   \renewcommand{\subglossentry}[3]{%
6489     &
6490     \glssubentryitem{##2}%
6491     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6492     ##3\tabularnewline
6493   }%
6494   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
6495     &\tabularnewline\fi}%
6496 }
6497 }
6498 {}

```

Four columns:

```

6499 \ifcsdef{@glsstyle@altsuperragged4col}
6500 {%
6501   \renewglossarystyle{altsuperragged4col}{%
6502     \renewenvironment{theglossary}%
6503       {\tablehead{}\tabletail}%
6504       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
6505         >{\raggedright}p{\glspagelistwidth}}}%
6506       {\end{supertabular}}}%
6507   \renewcommand*{\glossaryheader}{}%
6508   \renewcommand{\glossentry}[2]{%
6509     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
6510     \glossentrydesc{##1}\glspostdescription &
6511     \glossentrysymbol{##1} & ##2\tabularnewline

```

```

6512 }%
6513 \renewcommand{\subglossentry}[3]{%
6514     &
6515     \glssubentryitem{##2}%
6516     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
6517     \glossentrysymbol{##2} & ##3\tabularnewline
6518 }%
6519 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & &
6520     &\tabularnewline\fi}%
6521 }
6522 }
6523 {}

```

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`.

```

6524 \ifdef{\@glsstyle@inline}
6525 {%
6526     \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
        Just use \glxtrpostdescription instead of \glspostdescription.
6527     \renewcommand*{\glsinlinedescformat}[3]{%
6528         \space#1\glxtrpostdescription}
6529     \renewcommand*{\glsinlinesubdescformat}[3]{%
6530         #1\glxtrpostdescription}
6531 }
6532 {}

```

2.8 Tree Styles

The `alttree` style is redefined to make it easier to made minor adjustments.

```

6533 \ifdef{\@glsstyle@alttree}
6534 {%

```

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.

```

6535 \newcommand{\glxtraltrtreeSymbolDescLocation}[2]{%
6536     {%
6537         \let\par\glxtrAltTreePar

```

```

6538      \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
6539      \glossentrydesc{#1}\glspostdescription \space #2\par
6540    }%
6541  }

```

`\trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

6542  \newlength\glxstrAltTreeIndent

```

`\lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

6543  \newcommand{\glxstrAltTreePar}{%
6544    @@par
6545    \glxstrAltTreeSetHangIndent
6546    \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
6547  }

```

`\symbolDescLocation` `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

6548  \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
6549    \glxstralttreeSymbolDescLocation{#2}{#3}%
6550  }

```

`\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.

```

6551  \newlength\glxstrtreeindent

```

`\glxstralttreeInit` User-level initialisation for the alttree style.

```

6552  \newcommand*{\glxstralttreeInit}{%
6553    \settowidth{\glxstrtreeindent}{\glstreenamfmt{\glsgetwidestname\space}}%
6554    \glxstrAltTreeIndent=\parindent
6555  }

```

`\eglissetwidest` The original `\glissetwidest` only uses `\def`. This uses `\protected@csedef`.

```

6556  \newcommand*{\eglissetwidest}[2][0]{%
6557    \protected@csedef{@glswidestname\romannumeral#1}{#2}%
6558  }

```

`\xglissetwidest` Like the above but uses `\protected@csxdef`.

```

6559  \newcommand*{\xglissetwidest}[2][0]{%
6560    \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
6561  }

```

`\lsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

6562  \newcommand*{\lsgetwidestname}{\@glswidestname}

```


`etwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```
6563 \newcommand*\glsgetwidestsubname}[1]{%
6564   \ifcsundef{@glswidestname\romannumeral#1}%
6565     {\@glswidestname}%
6566     {\csuse{@glswidestname\romannumeral#1}}%
6567 }
```

`estTopLevelName` `CamelCase` is easier for long command names. Provide a `CamelCase` synonym of `\glsfindwidesttoplevelname`

```
6568 \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.

```
6569 \newrobustcmd*\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
6570   \dimen@=0pt\relax
6571   \gls@tmplen=0pt\relax
6572   \foralllglossaries[#1]{\@gls@type}%
6573   {%
6574     \forglsentries[\@gls@type]{\@glo@label}%
6575     {%
6576       \ifglsused{\@glo@label}%
6577       {%
6578         \ifglshasparent{\@glo@label}%
6579         {}%
6580         {%
6581           \settowidth{\dimen@}%
6582             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6583           \ifdim\dimen@>\gls@tmplen
6584             \gls@tmplen=\dimen@
6585             \eglssetwidest{\glsentryname{\@glo@label}}%
6586           \fi
6587         }%
6588       }%
6589     }%
6590   }%
6591 }%
6592 }
```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
6593 \newrobustcmd*\glsFindWidestUsedAnyName}[1][\@glo@types]{%
6594   \dimen@=0pt\relax
6595   \gls@tmplen=0pt\relax
6596   \foralllglossaries[#1]{\@gls@type}%
6597   {%
6598     \forglsentries[\@gls@type]{\@glo@label}%
6599     {%
```

```

6600     \ifglused{\@glo@label}%
6601     {%
6602         \settowidth{\dimen@}%
6603         {\glstreenamfmt{\glentryname{\@glo@label}}}%
6604         \ifdim\dimen@>\gls@tmplen
6605             \gls@tmplen=\dimen@
6606             \eglssetwidest{\glentryname{\@glo@label}}%
6607         \fi
6608     }%
6609     {%
6610 }%
6611 }%
6612 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

6613 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
6614     \dimen@=0pt\relax
6615     \gls@tmplen=0pt\relax
6616     \forallglossaries[#1]{\@gls@type}%
6617     {%
6618         \forglsentries[\@gls@type]{\@glo@label}%
6619         {%
6620             \settowidth{\dimen@}%
6621             {\glstreenamfmt{\glentryname{\@glo@label}}}%
6622             \ifdim\dimen@>\gls@tmplen
6623                 \gls@tmplen=\dimen@
6624                 \eglssetwidest{\glentryname{\@glo@label}}%
6625             \fi
6626         }%
6627     }%
6628 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

6629 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
6630     \dimen@=0pt\relax
6631     \dimen@i=0pt\relax
6632     \dimen@ii=0pt\relax
6633     \forallglossaries[#1]{\@gls@type}%
6634     {%
6635         \forglsentries[\@gls@type]{\@glo@label}%
6636         {%
6637             \ifglused{\@glo@label}%
6638             {%
6639                 \ifglshasparent{\@glo@label}%
6640                 {%
6641                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
6642                     \ifglshasparent{\@glo@parent}%
6643                 }%

```

```

6644         \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}}%
6645         \ifglshasparent{\@glo@parent}%
6646         {%
6647         {%
6648             \settowidth{\gls@tmplen}%
6649             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6650             \ifdim\gls@tmplen>\dimen@ii
6651             \dimen@ii=\gls@tmplen
6652             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6653         \fi
6654         }%
6655     }%
6656     {%
6657         \settowidth{\gls@tmplen}%
6658         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6659         \ifdim\gls@tmplen>\dimen@i
6660         \dimen@i=\gls@tmplen
6661         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6662     \fi
6663     }%
6664     }%
6665     {%
6666         \settowidth{\gls@tmplen}%
6667         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6668         \ifdim\gls@tmplen>\dimen@
6669         \dimen@=\gls@tmplen
6670         \eglssetwidest{\glsentryname{\@glo@label}}%
6671     \fi
6672     }%
6673     }%
6674     {}%
6675     }%
6676     }%
6677 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

6678 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
6679     \dimen@=0pt\relax
6680     \dimen@i=0pt\relax
6681     \dimen@ii=0pt\relax
6682     \foralllglossaries[#1]{\@gls@type}%
6683     {%
6684         \forglsentries[\@gls@type]{\@glo@label}%
6685         {%
6686             \ifglshasparent{\@glo@label}%
6687             {%
6688                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
6689                 \ifglshasparent{\@glo@parent}%
6690                 {%

```

```

6691      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
6692      \ifglshasparent{\@glo@parent}%
6693      {%
6694      {%
6695          \settowidth{\gls@tmplen}%
6696              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6697          \ifdim\gls@tmplen>\dimen@ii
6698              \dimen@ii=\gls@tmplen
6699              \eglssetwidest[2]{\glsentryname{\@glo@label}}%
6700          \fi
6701      }%
6702      }%
6703      {%
6704          \settowidth{\gls@tmplen}%
6705              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6706          \ifdim\gls@tmplen>\dimen@i
6707              \dimen@i=\gls@tmplen
6708              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
6709          \fi
6710      }%
6711      }%
6712      {%
6713          \settowidth{\gls@tmplen}%
6714              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6715          \ifdim\gls@tmplen>\dimen@
6716              \dimen@=\gls@tmplen
6717              \eglssetwidest{\glsentryname{\@glo@label}}%
6718          \fi
6719      }%
6720      }%
6721      }%
6722      }

```

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.

```

6723 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
6724     \dimen@=0pt\relax
6725     \gls@tmplen=0pt\relax
6726     #2=0pt\relax
6727     \forallglossaries[#1]{\@gls@type}%
6728     {%
6729         \forglsentries[\@gls@type]{\@glo@label}%
6730         {%
6731             \ifglused{\@glo@label}%
6732             {%
6733                 \settowidth{\dimen@}%
6734                     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6735                 \ifdim\dimen@>\gls@tmplen
6736                     \gls@tmplen=\dimen@

```

```

6737         \eglssetwidest{\glsentryname{\@glo@label}}%
6738     \fi
6739     \settowidth{\dimen@}%
6740     {\glsentrysymbol{\@glo@label}}%
6741     \ifdim\dimen@>#2\relax
6742         #2=\dimen@
6743     \fi
6744 }%
6745 {}%
6746 }%
6747 }%
6748 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

6749 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
6750     \dimen@=0pt\relax
6751     \gls@tmplen=0pt\relax
6752     #2=0pt\relax
6753     \forallglossaries[#1]{\@gls@type}%
6754     {%
6755         \forglsentries[\@gls@type]{\@glo@label}%
6756         {%
6757             \settowidth{\dimen@}%
6758             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
6759             \ifdim\dimen@>\gls@tmplen
6760                 \gls@tmplen=\dimen@
6761                 \eglssetwidest{\glsentryname{\@glo@label}}%
6762             \fi
6763             \settowidth{\dimen@}%
6764             {\glsentrysymbol{\@glo@label}}%
6765             \ifdim\dimen@>#2\relax
6766                 #2=\dimen@
6767             \fi
6768         }%
6769     }%
6770 }

```

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.

```

6771 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
6772     \dimen@=0pt\relax
6773     \gls@tmplen=0pt\relax
6774     #2=0pt\relax
6775     #3=0pt\relax
6776     \forallglossaries[#1]{\@gls@type}%
6777     {%
6778         \forglsentries[\@gls@type]{\@glo@label}%

```

```

6779      {%
6780        \ifglsused{\@glo@label}%
6781      {%
6782        \settowidth{\dimen@}%
6783          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6784        \ifdim\dimen@>\gls@tmplen
6785          \gls@tmplen=\dimen@
6786          \eglssetwidest{\glsentryname{\@glo@label}}%
6787        \fi
6788        \settowidth{\dimen@}%
6789          {\glsentrysymbol{\@glo@label}}%
6790        \ifdim\dimen@>#2\relax
6791          #2=\dimen@
6792        \fi
6793        \settowidth{\dimen@}%
6794          {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6795        \ifdim\dimen@>#3\relax
6796          #3=\dimen@
6797        \fi
6798      }%
6799    }%
6800  }%
6801 }%
6802 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

6803 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
6804   \dimen@=0pt\relax
6805   \gls@tmplen=0pt\relax
6806   #2=0pt\relax
6807   #3=0pt\relax
6808   \foralllglossaries[#1]{\@gls@type}%
6809   {%
6810     \forglsentries[\@gls@type]{\@glo@label}%
6811     {%
6812       \settowidth{\dimen@}%
6813         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
6814       \ifdim\dimen@>\gls@tmplen
6815         \gls@tmplen=\dimen@
6816         \eglssetwidest{\glsentryname{\@glo@label}}%
6817       \fi
6818       \settowidth{\dimen@}%
6819         {\glsentrysymbol{\@glo@label}}%
6820       \ifdim\dimen@>#2\relax
6821         #2=\dimen@
6822       \fi
6823       \settowidth{\dimen@}%
6824         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6825       \ifdim\dimen@>#3\relax

```

```

6826         #3=\dimen@
6827     \fi
6828 }%
6829 }%
6830 }

```

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.

```

6831 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
6832     \dimen@=0pt\relax
6833     \gls@tmplen=0pt\relax
6834     #2=0pt\relax
6835     \forallglossaries[#1]{\@gls@type}%
6836     {%
6837         \forglstentries[\@gls@type]{\@glo@label}%
6838         {%
6839             \ifglstused{\@glo@label}%
6840             {%
6841                 \settowidth{\dimen@}%
6842                 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6843                 \ifdim\dimen@>\gls@tmplen
6844                     \gls@tmplen=\dimen@
6845                     \eglissetwidest{\glstentryname{\@glo@label}}%
6846                 \fi
6847                 \settowidth{\dimen@}%
6848                 {\GlsXtrFormatLocationList{\glstentrynumberlist{\@glo@label}}}%
6849                 \ifdim\dimen@>#2\relax
6850                     #2=\dimen@
6851                 \fi
6852             }%
6853         }%
6854     }%
6855 }%
6856 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

6857 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
6858     \dimen@=0pt\relax
6859     \gls@tmplen=0pt\relax
6860     #2=0pt\relax
6861     \forallglossaries[#1]{\@gls@type}%
6862     {%
6863         \forglstentries[\@gls@type]{\@glo@label}%
6864         {%
6865             \settowidth{\dimen@}%
6866             {\glstreenamfmt{\glstentryname{\@glo@label}}}%
6867             \ifdim\dimen@>\gls@tmplen
6868                 \gls@tmplen=\dimen@

```

```

6869         \eglssetwidest{\glsentryname{\@glo@label}}}%
6870     \fi
6871     \settowidth{\dimen@}%
6872         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
6873     \ifdim\dimen@>#2\relax
6874         #2=\dimen@
6875     \fi
6876 }%
6877 }%
6878 }

```

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`.

```

6879 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
6880     \glstreeindent=\glsxtrtreetopindent\relax
6881 }

```

teTreeSubIndent

```

6882 %\cs{\glsxtrComputeTreeSubIndent}\marg{level}\marg{label}\marg{register}
6883 %\end{macrocode}
6884 % Compute the indent for the sub-entries. The first argument is the
6885 % level, the second argument is the entry label and the third
6886 % argument is the length register used to store the computed indent.
6887 % \begin{macrocode}
6888 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
6889     \ifcsundef{@glswidestname\romannumeral#1}%
6890     {%
6891         \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
6892     }%
6893     {%
6894         \settowidth{#3}{\glstreenamefmt{%
6895             \csname @glswidestname\romannumeral#1\endcsname\space}}%
6896     }%
6897 }

```

eeSetHangIndent Set `\hangindent` for top-level entries:

```

6898 \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

etSubHangIndent Set `\hangindent` for sub-entries:

```

6899 \newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `alttree`:

```

6900 \renewglossarystyle{alttree}{%
6901     \renewenvironment{theglossary}%
6902     {%
6903         \glsxtralttreeInit
6904         \def\@gls@prevlevel{-1}%

```



```

6905     \mbox{}\par}%
6906     {\par}%
6907 \renewcommand*\glossaryheader{}%
6908 \renewcommand*\glsgroupheading[1]{}%
6909 \renewcommand\glossentry[2]{%
6910     \ifnum\@gls@prevlevel=0\relax
6911     \else
6912         \glstrComputeTreeIndent{##1}%
6913     \fi
6914     \parindent\glstreeindent
6915     \glstrAltTreeSetHangIndent
6916     \makebox[Opt][r]%
6917     {%
6918         \glstreenamebox{\glstreeindent}%
6919         {%
6920             \glstryitem{##1}%
6921             \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
6922         }%
6923     }%
6924     \glstralttreeSymbolDescLocation{##1}{##2}%
6925     \def\@gls@prevlevel{0}%
6926 }
6927 \renewcommand\subglossentry[3]{%
6928     \ifnum##1=1\relax
6929         \glssubentryitem{##2}%
6930     \fi
6931     \ifnum\@gls@prevlevel=##1\relax
6932     \else
6933         \glstrComputeTreeSubIndent{##1}{##2}{\@gls@tmplen}%
6934         \ifnum\@gls@prevlevel<##1\relax
6935             \setlength\glstreeindent\@gls@tmplen
6936             \addtolength\glstreeindent\parindent
6937             \parindent\glstreeindent
6938         \else
6939             \ifnum\@gls@prevlevel=0\relax
6940                 \glstrComputeTreeIndent{##2}%
6941             \else
6942                 \glstrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
6943             \fi
6944             \addtolength\parindent{-\glstreeindent}%
6945             \setlength\glstreeindent\parindent
6946         \fi
6947     \fi
6948     \glstrAltTreeSetSubHangIndent{##1}%
6949     \makebox[Opt][r]{\glstreenamebox{\@gls@tmplen}{%
6950         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
6951     \glstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
6952     \def\@gls@prevlevel{##1}%
6953 }%

```

```

6954 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
6955 }
6956 }%
6957 {%

```

Assume the style isn't required if it hasn't already been defined.

```

6958 }

```

Reset the default style

```

6959 \ifx\@glossary@default@style\relax
6960 \else
6961 \setglossarystyle{\@glxtr@current@style}
6962 \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 | 35 |
| General: Initial experimental release | 4 | \@Glsymbolplural@: added | |
| 0.2 (2015-11-30) | | redefinition | 35 |
| \Glsfmtshort: new | 178 | \@Glstext@: added redefinition | 31 |
| \glsfmtshort: new | 178 | \@Glsuseri@: added redefinition | 36 |
| \Glsfmtshortpl: new | 179 | \@Glsuserii@: added redefinition | 36 |
| \glsfmtshortpl: new | 178 | \@Glsuseriii@: added redefinition | 36 |
| short: switched inline full form to short | | \@Glsuseriv@: added redefinition | 36 |
| (long) | 145 | \@Glsuserv@: added redefinition | 37 |
| 0.3 (2015-12-02) | | \@Glsuservi@: added redefinition | 37 |
| \@ACRlong: added redefinition | 40 | \@acrlong: added redefinition | 40 |
| \@ACRlongpl: added redefinition | 41 | \@acrlongpl: added redefinition | 41 |
| \@ACRshort: added redefinition | 38 | \@acrshort: added redefinition | 37 |
| \@ACRshortpl: added redefinition | 39 | \@acrshortpl: added redefinition | 38 |
| \@Acrlong: added redefinition | 40 | \@gls@field@link: added optional | |
| \@Acrlongpl: added redefinition | 41 | argument | 28 |
| \@Acrshort: added redefinition | 38 | \@glsdescplural@: added redefinition | 34 |
| \@Acrshortpl: added redefinition | 39 | \@glsfirst@: added redefinition | 31 |
| \@GLSdesc@: added redefinition | 34 | \@glsfirstplural@: added redefinition | 32 |
| \@GLSdescplural@: added redefinition | 34 | \@glsplural@: added redefinition | 32 |
| \@GLSfirst@: added redefinition | 31 | \@glssymbolplural@: added | |
| \@GLSfirstplural@: added redefinition | 33 | redefinition | 35 |
| \@GLSname@: added redefinition | 33 | \@glsxtr@defaultnoglossarywarning: | |
| \@GLSplural@: added redefinition | 32 | new | 81 |
| \@GLSSymbol@: added redefinition | 35 | \@glsxtr@field@linkdefs: new | 30 |
| \@GLSSymbolplural@: added | | \@glsxtr@insertdots: new | 117 |
| redefinition | 35 | \@print@glossary: added redefinition | 78 |
| \@GLStext@: added redefinition | 30 | \glsabbrvdefaultfont: renamed from | |
| \@GLSuseri@: added redefinition | 36 | \abbrvdefaultfont | 121 |
| \@GLSuserii@: added redefinition | 36 | \glsaccessdesc: new | 87 |
| \@GLSuseriii@: added redefinition | 36 | \glsaccessdescplural: new | 88 |
| \@GLSuseriv@: added redefinition | 37 | \glsaccessfirst: new | 85 |
| \@GLSuserv@: added redefinition | 37 | \glsaccessfirstplural: new | 86 |
| \@GLSuservi@: added redefinition | 37 | \Glsaccesslong: new | 90 |
| \@Glsdesc@: added redefinition | 34 | \glsaccesslong: new | 90 |
| \@Glsdescplural@: added redefinition | 34 | \glsaccessname: new | 83 |
| \@Glsfirst@: added redefinition | 31 | \glsaccessplural: new | 84 |
| \@Glsfirstplural@: added redefinition | 33 | \Glsaccessshort: new | 89 |
| \@Glsname@: added redefinition | 33 | \glsaccessshort: new | 88 |
| \@Glsplural@: added redefinition | 32 | \Glsaccessshortpl: new | 89 |

| | | | |
|---|-----|------------------------------------|-----|
| \glsaccesssshortpl: new | 89 | \cGLSpl: new | 62 |
| \glsaccessssymbol: new | 86 | \cGLSpl@: new | 62 |
| \glsaccessssymbolplural: new | 87 | \glsxtr@setentrycountunsetattr: | |
| \glsaccesstext: new | 84 | new | 57 |
| \glstentryfmt: added check for short .. | 27 | \cGLS: new | 62 |
| \glslongpltok: new | 117 | \cGLSformat: new | 62 |
| \glsshortpltok: new | 117 | \cGLSpl: new | 62 |
| \glsxtrdiscardperiod: added check | | \cGLSplformat: new | 62 |
| for plural | 114 | \GlossariesExtraWarningNoLine: | |
| \GLSxtrlongpl: new | 131 | new | 9 |
| \Glsxtrlongpl: new | 130 | \glsenableentrycount: new | 58 |
| \glsxtrlongpl: new | 130 | \glsfirstabbrvdefaultfont: new .. | 121 |
| \glsxtrNoGlossaryWarning: new | 12 | \glsfirstlongdefaultfont: new ... | 121 |
| \glsxtrpostlinkAddDescOnFirstUse: | | \Glsfmtfirst: new | 180 |
| new | 114 | \Glsfmtfirst: new | 180 |
| \glsxtrpostlinkAddSymbolOnFirstUse: | | \Glsfmtfirstpl: new | 181 |
| new | 114 | \Glsfmtfirstpl: new | 181 |
| \glsxtrpostlinkendsentence: new .. | 113 | \Glsfmtplural: new | 180 |
| \GLSxtrshortpl: new | 129 | \Glsfmtplural: new | 179 |
| \Glsxtrshortpl: new | 128 | \Glsfmtshort: changed to use | |
| \glsxtrshortpl: new | 128 | \Glsxtrtitleshort | 178 |
| short-long-desc: fixed name to use | | renamed from \Glstentryfmtshort .. | 178 |
| \glslabeltok | 140 | \Glsfmtshort: changed to use | |
| \newabbreviation: fixed family name in | | \Glsxtrtitleshort | 178 |
| \setkeys | 118 | renamed from \glstentryfmtshort .. | 178 |
| long-short-desc: fixed name to use | | \Glsfmtshortpl: changed to use | |
| \glslabeltok | 138 | \Glsxtrtitleshortpl | 179 |
| 0.4 (2015-12-03) | | renamed from | |
| \@glsxtr@doabbreviationsdef: added | | \Glstentryfmtshortpl | 179 |
| redefinition of \acronymtype | 10 | \Glsfmtshortpl: changed to use | |
| \Glsfmtshort: changed to use | | \Glsxtrtitleshortpl | 178 |
| \Glsxtrshort | 178 | renamed from | |
| \Glsfmtshort: changed to use | | \glstentryfmtshortpl | 178 |
| \Glsxtrshort | 178 | \Glsfmttext: new | 179 |
| \Glsfmtshortpl: changed to use | | \Glsfmttext: new | 179 |
| \Glsxtrshortpl | 179 | \glshasattribute: new | 95 |
| \Glsfmtshortpl: changed to use | | \glshascategoryattribute: new | 94 |
| \Glsxtrshortpl | 178 | \GlsXtrEnableEntryCounting: new .. | 57 |
| \glsxtrifemptyglossary: new | 15 | \glsxtrifcounttrigger: new | 60 |
| \glsxtrnewnumber: added extra | | \glsxtrscfont: new | 149 |
| argument | 99 | \glsxtrscsuffix: new | 149 |
| \glsxtrnewsymbol: added extra | | \glsxtrsmfont: new | 153 |
| argument | 98 | \glsxtrsmsuffix: new | 153 |
| \MakeAcronymsAbbreviations: set the | | short-em: new | 160 |
| default type to \acronymtype | 71 | short-em-desc: new | 160 |
| \newterm: fixed name argument | 98 | short-em-footnote: new | 162 |
| 0.5 (2015-12-07) | | short-em-long: new | 158 |
| \cGLS: new | 62 | short-em-long-desc: new | 159 |
| \cGLS@: new | 62 | short-em-postfootnote: new | 162 |

| | | | |
|-------------------------------------|-----|---|-----|
| short-sc-footnote: new | 152 | \Glsxtrheadtext: now uses headuc | |
| short-sc-postfootnote: new | 153 | attribute | 171 |
| short-sm: new | 154 | \glxtrheadtext: now uses headuc | |
| short-sm-desc: new | 155 | attribute | 171 |
| short-sm-footnote: new | 156 | short-long: switch off regular attribute | |
| short-sm-long: new | 154 | if set | 139 |
| short-sm-long-desc: new | 154 | short-long-desc: switch off regular | |
| short-sm-postfootnote: new | 156 | attribute if set | 140 |
| long-noshort-em: new | 161 | long-short: switch off regular attribute | |
| long-noshort-em-desc: new | 161 | if set | 137 |
| long-noshort-sm: new | 155 | long-short-desc: switch off regular | |
| long-noshort-sm-desc: new | 155 | attribute if set | 138 |
| long-short-em: new | 157 | footnote: switch off regular attribute if | |
| long-short-em-desc: new | 157 | set | 141 |
| long-short-sm: new | 153 | postfootnote: switch off regular | |
| long-short-sm-desc: new | 154 | attribute if set | 143 |
| 0.5.1 (2015-12-02) | | 0.5.2 (2015-12-08) | |
| \Glsaccessstext: new | 84 | \@GLSdesc@: added accessibility support | 34 |
| 0.5.1 (2015-12-07) | | \@GLSdescplural@: added accessibility | |
| \@glxtr@doaccsupp: new | 12 | support | 34 |
| General: removed \ifglxtruseuhead | 169 | \@GLSfirst@: added accessibility | |
| \Glsaccessdesc: new | 88 | support | 31 |
| \Glsaccessdescplural: new | 88 | \@GLSfirstplural@: added accessibility | |
| \Glsaccessfirst: new | 85 | support | 33 |
| \Glsaccessfirstplural: new | 86 | \@GLSname@: added accessibility support | 33 |
| \Glsaccessname: new | 83 | \@GLSplural@: added accessibility | |
| \Glsaccessplural: new | 85 | support | 32 |
| \Glsaccesssymbol: new | 86 | \@GLSsymbol@: added accessibility | |
| \Glsaccesssymbolplural: new | 87 | support | 35 |
| \Glsxtrheadfirst: now uses headuc | | \@GLSsymbolplural@: added | |
| attribute | 173 | accessibility support | 35 |
| \glxtrheadfirst: now uses headuc | | \@GLStext@: added accessibility support | 30 |
| attribute | 173 | \@GLSdesc@: added accessibility support | 34 |
| \Glsxtrheadfirstplural: now uses | | \@GLSdescplural@: added accessibility | |
| headuc attribute | 174 | support | 34 |
| \glxtrheadfirstplural: now uses | | \@GLSfirst@: added accessibility | |
| headuc attribute | 173 | support | 31 |
| \Glsxtrheadplural: now uses headuc | | \@GLSfirstplural@: added accessibility | |
| attribute | 172 | support | 33 |
| \glxtrheadplural: now uses headuc | | \@GLSname@: add accessibility support .. | 33 |
| attribute | 172 | \@GLSplural@: added accessibility | |
| \Glsxtrheadshort: now uses headuc | | support | 32 |
| attribute | 170 | \@GLSsymbol@: added accessibility | |
| \glxtrheadshort: now uses headuc | | support | 35 |
| attribute | 169 | \@GLSsymbolplural@: added | |
| \Glsxtrheadshortpl: now uses headuc | | accessibility support | 35 |
| attribute | 171 | \@GLStext@: added accessibility support | 31 |
| \glxtrheadshortpl: now uses headuc | | \@GLSdesc@: added accessibility support | 34 |
| attribute | 170 | | |

| | | | |
|--|--------|--|-----|
| \@glsdescplural@: added accessibility support | 34 | \glsxtrnewabbrevpresetkeyhook: new | 119 |
| \@glsfirst@: added accessibility support | 31 | \glsxtrtagfont: new | 112 |
| \@glsfirstplural@: added accessibility support | 32 | \KV@printgloss@nonumberlist: added | 27 |
| \@glsname@: added accessibility support | 33 | \mfu@checkword@do: added | 111 |
| \@glsplural@: added accessibility support | 32 | \setabbreviationstyle: added check for post-definition style switch | 134 |
| \@glsymbol@: added accessibility support | 35 | 0.5.3 (2015-12-09) | |
| \@glsymbolplural@: added accessibility support | 35 | \@glsxtr@autoindex@at: new | 108 |
| \@glstext@: added accessibility support | 30 | \@glsxtr@autoindex@encap: new ... | 108 |
| \@glsxtr@activate@initialtagging: new | 112 | \@glsxtr@autoindex@esc: new | 109 |
| \@glsxtr@do@titlecaps@warn: new | 112 | \@glsxtr@autoindex@level: new ... | 108 |
| \@glsxtr@tag: new | 112 | \@glsxtr@autoindex@setname: new | 106 |
| General: fixed typo in glossaries-accsupp and tidied up code to use just one | | \@glsxtr@doabbreviationsdef: new .. | 9 |
| \@ifpackageloaded | 83 | General: removed | |
| removed \glsxtrabbrvfmt | 131 | \GlsXtrNoGlsWarningNoAutoMakeMain | 80 |
| \glossaryentrynumbers: added | 24 | \glsdescwidth: added | 24 |
| \Glossentrydesc: added | 110 | \glspagelistwidth: added | 24 |
| \Glossentryname: added | 104 | \glsxtrdoautoindexname: new | 106 |
| \Glossentrysymbol: added | 110 | \glsxtrpostnamehook: new | 105 |
| \glossentrysymbol: added | 110 | \if@glsxtr@format@override: new | 105 |
| \GLSaccessdesc: new | 88, 92 | \ProvidesGlossariesExtraLang: new | 184 |
| \GLSaccessdescplural: new | 88, 93 | \RequireGlossariesExtraLang: new | 184 |
| \GLSaccessfirst: new | 85, 91 | 0.5.4 (2015-12-15) | |
| \GLSaccessfirstplural: new | 86, 92 | \@newglossaryentry@defunitcounters: new | 63 |
| \GLSaccesslong: new | 90, 93 | \@GLSxtr@p@acrlong@: new | 50 |
| \GLSaccesslongpl: new | 90, 94 | \@GLSxtr@p@acrlongpl@: new | 51 |
| \Glsaccesslongpl: new | 90 | \@GLSxtr@p@acrshort@: new | 50 |
| \glsaccesslongpl: new | 90 | \@GLSxtr@p@acrshortpl@: new | 50 |
| \GLSaccessname: new | 84, 91 | \@GLSxtr@p@long@: new | 50 |
| \GLSaccessplural: new | 85, 91 | \@GLSxtr@p@longpl@: new | 50 |
| \GLSaccessshort: new | 89, 93 | \@GLSxtr@p@plural@: new | 48 |
| \GLSaccessshortpl: new | 89, 93 | \@GLSxtr@p@short@: new | 49 |
| \GLSaccesssymbol: new | 87, 92 | \@GLSxtr@p@shortpl@: new | 49 |
| \GLSaccesssymbolplural: new ... | 87, 92 | \@GLSxtr@p@text@: new | 48 |
| \GLSaccessstext: new | 84, 91 | \@GlsXtrEnableOnTheFly: new | 20 |
| \glsentryfmt: moved | | \@Glsxtr: new | 21 |
| \glssetabbrvfmt from | | \@Glsxtr@p@acrlong@: new | 50 |
| \glsxtrabbrvfmt to here | 27 | \@Glsxtr@p@acrlongpl@: new | 51 |
| \GlsXtrEnableInitialTagging: new | 110 | \@Glsxtr@p@acrshort@: new | 50 |
| \glsxtrfieldtitlecase: new | 99 | \@Glsxtr@p@acrshortpl@: new | 50 |
| \GlsXtrFormatLocationList: new ... | 25 | \@Glsxtr@p@long@: new | 50 |
| | | \@Glsxtr@p@longpl@: new | 50 |
| | | \@Glsxtr@p@plural@: new | 48 |
| | | \@Glsxtr@p@short@: new | 49 |
| | | \@Glsxtr@p@shortpl@: new | 49 |
| | | \@Glsxtr@p@text@: new | 48 |

| | | | |
|---------------------------------------|-----|------------------------------------|-----|
| \@Glsxtrpl: new | 22 | \glxtr: new | 21 |
| \@alt@glshyp@opt: new | 46 | \glxtrcat: new | 20 |
| \@glscalt@hyp@opt: new | 46 | \glxtrdowrglossaryhook: new | 46 |
| \@glscalt@hyp@opt@char: new | 46 | \GlsXtrEnableEntryUnitCounting: | |
| \@glscalt@hyp@opt@keys: new | 46 | new | 68 |
| \@glsc@increment@currunitcount: | | \GlsXtrEnableOnTheFly: new | 20 |
| new | 64 | \Glsxtrpl: new | 22 |
| \@glsc@local@increment@currunitcount: | | \glxtrpl: new | 21 |
| new | 64 | \glxtrpostlocalreset: new | 57 |
| \@glsc@setdefault@glslink@opts: | | \glxtrpostlocalunset: new | 56 |
| new | 44 | \glxtrpostreset: new | 57 |
| \@glxtr: new | 21 | \glxtrpostunset: new | 56 |
| \@glxtr@addunitcounter: new | 63 | \glxtrprotectlinks: new | 47 |
| \@glxtr@currunitcount: new | 65 | \GlsXtrSetAltModifier: new | 46 |
| \@glxtr@ifunitcounter: new | 64 | \GlsXtrSetDefaultGlsOpts: new | 45 |
| \@glxtr@p@acrlong@: new | 50 | \glxtrstarflywarn: new | 20 |
| \@glxtr@p@acrlongpl@: new | 51 | \GlsXtrWarning: new | 22 |
| \@glxtr@p@acrshort@: new | 50 | \MakeAcronymsAbbreviations: now | |
| \@glxtr@p@acrshortpl@: new | 50 | disables \setacronymstyle | 71 |
| \@glxtr@p@long@: new | 49 | 1.0 (2016-01-24) | |
| \@glxtr@p@longpl@: new | 50 | \@glxtr@autoindexcrossrefs: new .. | 9 |
| \@glxtr@p@plural@: new | 48 | \@glxtr@idx@displaynumberlist: | |
| \@glxtr@p@short@: new | 48 | new | 76 |
| \@glxtr@p@shortpl@: new | 49 | \@glxtr@idx@entrynumberlist: new | 77 |
| \@glxtr@p@text@: new | 48 | \@glxtr@noidx@displaynumberlist: | |
| \@glxtr@prevunitcount: new | 65 | new | 76 |
| \@glxtr@setentryunitcountunsetattr: | | \@glxtr@noidx@entrynumberlist: | |
| new | 69 | new | 77 |
| \@glxtr@unitcountlist: new | 63 | \@glxtr@noidx@numberlistloop: | |
| \@glxtrpl: new | 21 | new | 77 |
| \@newglossaryentryposthook: added | | \@glxtr@reg@glosslist: new | 72 |
| empty see value if not set and added | | \makeglossaries: new | 72 |
| ‘see’ to field key map | 17 | 1.01 (2016-02-02) | |
| \@sGlsXtrEnableOnTheFly: new | 20 | \glxtrdiscardperiod: added check | |
| \cGlsformat: added | 63 | for first use | 114 |
| \cGlsformat: added | 63 | short-desc: fixed typo in | |
| \cGlsplformat: added | 63 | \glxtrinlinefullformat and | |
| \cGlsplformat: added | 63 | added missing second argument ... | 146 |
| \glsdisablehyper: added | 47 | 1.02 (2016-04-25) | |
| \glsdohyperlink: added | 47 | \@glxtr@current@style: new | 23 |
| \glsdonohyperlink: added | 47 | \Glsfmtfull: new | 183 |
| \glsenableentryunitcount: new | 65 | \glsfmtfull: new | 182 |
| \glshasattribute: added check for | | \Glsfmtfullpl: new | 183 |
| entry’s existence | 95 | \glsfmtfullpl: new | 183 |
| \glusifattribute: added check for | | \Glsfmtlong: new | 181 |
| entry’s existence | 96 | \glsfmtlong: new | 181 |
| \glspostlinkhook: added existence | | \Glsfmtlongpl: new | 182 |
| check | 113 | \glsfmtlongpl: new | 182 |
| \Glsxtr: new | 21 | \Glsxtrheadfull: new | 177 |

| | | | |
|---|-----|---|-----|
| \glxtrheadfull: new | 176 | \@GLSplural@: set abbreviation and regular format | 32 |
| \Glsxtrheadfullpl: new | 177 | \@GLSsymbol@: set regular format | 35 |
| \glxtrheadfullpl: new | 176 | \@GLSsymbolplural@: set regular format | 35 |
| \glxtrheadlong: new | 175 | \@GLStext@: set abbreviation and regular format | 30 |
| \glxtrheadlongpl: new | 176 | \@GLSuseri@: set regular format | 36 |
| \glxtrheadlongpl: new | 175 | \@GLSuserii@: set regular format | 36 |
| \Glsxtrtitlefull: new | 177 | \@GLSuseriii@: set regular format | 36 |
| \glxtrtitlefull: new | 176 | \@GLSuseriv@: set regular format | 37 |
| \Glsxtrtitlefullpl: new | 178 | \@GLSuserv@: set regular format | 37 |
| \glxtrtitlefullpl: new | 177 | \@GLSuservi@: set regular format | 37 |
| \Glsxtrtitlelong: new | 175 | \@Glsdesc@: set abbreviation and regular format | 34 |
| \glxtrtitlelong: new | 175 | \@Glsdescplural@: set abbreviation and regular format | 34 |
| \Glsxtrtitlelongpl: new | 176 | \@Glsfirst@: set abbreviation and regular format | 31 |
| \glxtrtitlelongpl: new | 175 | \@Glsfirstplural@: set abbreviation and regular format | 33 |
| \ifglxtrinsetinside: new | 137 | \@Glsname@: set abbreviation and regular format | 33 |
| postfootnote: added redef of \glxtrsetupfulldefs | 143 | \@Glsplural@: set abbreviation and regular format | 32 |
| stylemods: new | 13 | \@Glsymbol@: set regular format | 35 |
| 1.03 (2016-04-27) | | \@Glsymbolplural@: set regular format | 35 |
| \@GLSfirstplural@: bug fix: misspelt cs name | 33 | \@Glstext@: set abbreviation and regular format | 31 |
| \@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural | 32 | \@GLSuseri@: set regular format | 36 |
| \@Glsfirstplural@: bug fix: misspelt cs name | 33 | \@GLSuserii@: set regular format | 36 |
| \@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural | 32 | \@GLSuseriii@: set regular format | 36 |
| \@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural | 32 | \@GLSuseriv@: set regular format | 36 |
| \glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl .. | 175 | \@GLSuserv@: set regular format | 37 |
| \glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl | 170 | \@GLSuservi@: set regular format | 37 |
| 1.04 (2015-04-30) | | \@gls@preglossaryhook: added check for entry's existence | 112 |
| short-em-footnote: renamed from "footnote-em" | 162 | \@glsdesc@: set abbreviation and regular format | 34 |
| 1.04 (2016-05-02) | | \@glsdescplural@: set abbreviation and regular format | 34 |
| \@@glxtrpostloctag: new | 26 | \@glsfirst@: set abbreviation and regular format | 31 |
| \@GLSdesc@: set abbreviation and regular format | 34 | \@glsfirstplural@: set abbreviation and regular format | 32 |
| \@GLSdescplural@: set abbreviation and regular format | 34 | \@glsname@: set abbreviation and regular format | 33 |
| \@GLSfirst@: set abbreviation format .. | 31 | \@glsplural@: set abbreviation and regular format | 32 |
| \@GLSfirstplural@: set abbreviation and regular format | 33 | \@glsymbol@: set regular format | 35 |
| \@GLSname@: set abbreviation and regular format | 33 | | |

| | | | |
|--|-----|--|-----|
| \@glssymbolplural@: set regular format | 35 | short-em-nolong-desc: new | 160 |
| \@glstext@: set abbreviation and regular format | 30 | short-em-postfootnote: renamed from “postfootnote-em” | 162 |
| \@glstr@deprecated@abbrstyle: | | short-footnote: new | 142 |
| new | 135 | short-long-user: new | 165 |
| \@glstr@do@style: new | 13 | short-long-user-desc: new | 166 |
| \@glstr@doloctag: new | 27 | short-nolong: new | 146 |
| \@glstr@idx@entrynumberlist: | | short-nolong-desc: new | 147 |
| switched from \let to \newcommand | 77 | short-postfootnote: new | 144 |
| \@glstr@pagetag: new | 26 | short-sc-footnote: renamed from “footnote-sc” | 152 |
| \@glstr@pagetag: new | 26 | short-sc-nolong: new | 151 |
| \@glstr@preloctag: new | 26 | short-sc-nolong-desc: new | 151 |
| \@glstr@postloctag: new | 26 | short-sc-postfootnote: renamed from “postfootnote-sc” | 153 |
| \@glstr@preloctag: new | 26 | short-sm-footnote: renamed from “footnote-sm” | 156 |
| \glossentrydesc: added glossdescfont attribute check | 100 | short-sm-nolong: new | 155 |
| \Glossentryname: added glossnamefont attribute check | 104 | short-sm-nolong-desc: new | 155 |
| \glossentryname: added glossnamefont attribute check | 101 | short-sm-postfootnote: renamed from “postfootnote-sm” | 156 |
| moved post name hook inside condition | 104 | \letabbreviationstyle: new | 135 |
| \glsabbrvmfont: new | 157 | \newabbreviationstyle: bug fix: corrected test for existence | 134 |
| \glsabbrvuserfont: new | 163 | long-em-noshort-em: new | 161 |
| \glsfirstabbrvmfont: new | 157 | long-em-noshort-em-desc: new | 162 |
| \glsfirstabbrvuserfont: new | 163 | long-em-short-em: new | 157 |
| \glsfirstlongemfont: new | 157 | long-em-short-em-desc: new | 158 |
| \glsfirstlonguserfont: new | 164 | long-noshort: new | 149 |
| \glsifnotregularcategory: new | 96 | long-noshort-desc: new | 149 |
| \glslongdefaultfont: new | 121 | long-noshort-em: renamed from “long-em” | 161 |
| \glslongemfont: new | 157 | long-noshort-em-desc: renamed from “long-desc-em” | 161 |
| \glslongfont: new | 121 | long-noshort-sc: renamed from “long-sc” | 151 |
| \glslonguserfont: new | 163 | long-noshort-sc-desc: renamed from “long-desc-sc” | 152 |
| \glstrassignfieldfont: new | 30 | long-noshort-sm: renamed from “long-sm” | 155 |
| \GlsXtrEnablePreLocationTag: new | 25 | long-noshort-sm-desc: renamed from \long-desc-sm | 155 |
| \glstrfirstscfont: new | 149 | long-short-user: new | 164 |
| \glstrfirstsmfont: new | 153 | long-short-user-desc: new | 165 |
| \glstrlongshortdescsort: new | 138 | \renewabbreviationstyle: new style: new | 135 |
| \glstrpostnamehook: added category check | 105 | | 13 |
| \glstrregularfont: new | 27 | | |
| \glstruserfield: new | 163 | | |
| \glstruserparen: new | 163 | | |
| \glstrusersuffix: new | 164 | | |
| \GlsXtrWarnDeprecatedAbbrStyle: | | | |
| new | 136 | | |
| short-em-long-em: new | 159 | | |
| short-em-long-em-desc: new | 160 | | |
| short-em-nolong: new | 160 | | |
| | | 1.05 (2016-06-10) | |
| | | \eglssetwidest: new | 192 |
| | | \glsFindWidestAnyName: new | 194 |

| | | | |
|--|-----|---|-----|
| \glsFindWidestAnyNameLocation: new | 199 | \@GLSfirst@: added check for nohyperfirst attribute | 32 |
| \glsFindWidestAnyNameSymbol: new | 197 | \@GLSfirstplural@: added check for nohyperfirst attribute | 33 |
| \glsFindWidestAnyNameSymbolLocation: new | 198 | \@GLSxtrp: new | 52 |
| \glsFindWidestLevelTwo: new | 195 | \@Glsfirst@: added check for nohyperfirst attribute | 31 |
| \glsFindWidestUsedAnyName: new | 193 | \@Glsfirstplural@: added check for nohyperfirst attribute | 33 |
| \glsFindWidestUsedAnyNameLocation: new | 199 | \@Glsxtrp: new | 52 |
| \glsFindWidestUsedAnyNameSymbol: new | 196 | \@gls@preglossaryhook: added \glossxtrsetpopts | 112 |
| \glsFindWidestUsedAnyNameSymbolLocation: new | 197 | \@glsfirst@: added check for nohyperfirst attribute | 31 |
| \glsFindWidestUsedLevelTwo: new | 194 | \@glsfirstplural@: added check for nohyperfirst attribute | 32 |
| \glsFindWidestUsedTopLevelName: new | 193 | \@glsxtrinmark: new | 167 |
| \glsfirstlongfootnotefont: new | 140 | \@glsxtrnotinmark: new | 168 |
| \glsgetwidestname: new | 192 | \@glsxtrp: new | 51 |
| \glsgetwidestsubname: new | 193 | \@glsxtrp@opt: new | 51 |
| \glslongfootnotefont: new | 140 | \glossxtrsetpopts: new | 51 |
| \glsxtrAltTreeIndent: new | 192 | \glsps: new | 54 |
| \glsxtralttreeInit: new | 192 | \glspt: new | 54 |
| \glsxtrAltTreePar: new | 192 | \glsxtr@entry@p: new | 52 |
| \glsxtrAltTreeSetHangIndent: new | 200 | \glsxtrabbrvfootnote: new | 141 |
| \glsxtrAltTreeSetSubHangIndent: new | 200 | \glsxtrchecknohyperfirst: new | 31 |
| \glsxtralttreeSubSymbolDescLocation: new | 192 | \glsxtrfieldtitlecasecs: new | 99 |
| \glsxtralttreeSymbolDescLocation: new | 191 | \glsxtrifinmark: new | 167 |
| \glsxtrComputeTreeIndent: new | 200 | \GLSxtrp: new | 55 |
| \glsxtrComputeTreeSubIndent: new | 200 | \Glsxtrp: new | 54 |
| \glsxtrtreetopindent: new | 192 | \glsxtrp: new | 53 |
| short-em-long: fixed incorrect font used by long form | 158 | \glsxtrsetpopts: new | 51 |
| \xglsssetwidest: new | 192 | short-long-desc: added text key | 140 |
| 1.06 (2016-06-18) | | fixed misspelling of \glsabbrvfont in plural key | 140 |
| \@glsdoifexistssorwarn: new | 9 | long-short-desc: added missing text key | 138 |
| \@glsxtr@docdefval: new | 8 | fixed misspelling of \glsabbrvfont | 138 |
| \@glsxtr@useseesee: new | 17 | footnote: changed first forms to use \glsfirstlongfootnotefont | 141 |
| General: disabled docdef key at the start of the document | 14 | postfootnote: removed \footnote from first keys | 142 |
| docdef option changed to choice | 8 | switched from \glsfirstlongfont to \glsfirstlongfootnotefont | 144 |
| \glsxtr@useseesee: new | 17 | \RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse | 71 |
| \glsxtruseseesee: new | 17 | 1.08 (2016-12-13) | |
| \glsxtruseseeseeformat: new | 17 | \@@glsxtr@record: new | 6 |
| \if@glsxtrdocdefrestricted: new | 8 | | |
| 1.07 (2016-08-15) | | | |
| \@@glsxtrp: new | 51 | | |

| | | | |
|--------------------------------------|----|---------------------------------------|----|
| \@GLS@: added \@glsxtr@record | 29 | \glsxtr@addloclistfield: new | 7 |
| \@GLSpl@: added \@glsxtr@record | 29 | \glsxtr@indexonly@saveentrycounter: | |
| \@Gls@: added \@glsxtr@record | 28 | new | 7 |
| \@Glspl@: added \@glsxtr@record | 29 | \glsxtr@record: new | 6 |
| \@gls@: added \@glsxtr@record | 28 | \glsxtr@resource: new | 82 |
| \@gls@@link@: added | | \glsxtr@saveentrycounter: new | 14 |
| \@glsxtr@record | 29 | \glsxtr@setup@record: new | 7 |
| \@gls@field@link: added | | \glsxtrassignfieldfont: added check | |
| \@glsxtr@record | 28 | for existence | 30 |
| \@gls@saveentrycounter: new | 14 | \glsxtrresourcefile: new | 82 |
| \@glsdispl: added \@glsxtr@record | 29 | \printunsrtglossaries: new | 82 |
| \@glspl@: added \@glsxtr@record | 28 | \printunsrtglossary: new | 82 |
| \@glsxtr@dorecord: new | 6 | 1.09 (2016-12-16) | |
| \@glsxtr@err@undefaction: new | 5 | \@glsxtr@gettype: new | 76 |
| \@glsxtr@record: new | 6 | \@glsxtr@mixed@assign@sortkey: | |
| \@glsxtr@warn@onexistsordo: new | 5 | new | 76 |
| \@glsxtr@warn@undefaction: new | 5 | \@printglossary: redefined to save | |
| \@print@unsrt@glossary: new | 83 | options | 75 |
| General: added record package option | 7 | \glsxtr@makeglossaries: new | 76 |
| \glsadd: added \@glsxtr@record | 29 | 1.10 (2016-12-17) | |
| \glsdoifexists: now defines | | \@GLSpl@: fixed bug caused by typo in | |
| \glslabel | 15 | command name | 29 |
| \glsxtr@@do@wrglossary: new | 14 | | |

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 | |
|-------------------------------------|--------------|
| \. | 114, 191 |
| \@@cGLS@ | 59, 67 |
| \@@cGLSpl@ | 59, 67 |
| \@@cGLspl@ | 59, 67 |
| \@@cglS@ | 59, 67 |
| \@@cglSpl@ | 59, 60, 67 |
| \@@do@wrglossary | 6, 73 |
| \@@do@wrglossary | 7, 8, 14, 30 |
| \@@glo@assign@sortkey | 76 |
| \@@glo@type | 82 |
| \@@glslocalreset | 57 |
| \@@glslocalunset | 56 |
| \@@glsreset | 56 |
| \@@glsunset | 56 |
| \@@glsxtr@autoindex@escspch | 108, 109 |
| \@@glsxtr@checkspch | 107–110 |
| \@@glsxtr@disabledflycommand | 23 |
| \@@glsxtr@record | 7, 8 |
| \@@glsxtrp | 51, 52 |
| \@@glsxtrpostloctag | 25 |
| \@@glsxtrpreloctag | 25, 26 |
| \@@newglossaryentry@defcounters | 58 |
| \@@newglossaryentry@defunitcounters | 65 |
| \@@par | 192 |
| \@ACRlong | 48 |
| \@ACRlongpl | 48 |
| \@ACRshort | 48 |
| \@ACRshortpl | 48 |
| \@Acrlong | 48 |
| \@Acrlongpl | 48 |
| \@Acrshort | 48 |
| \@Acrshortpl | 48 |
| \@GLS@ | 47, 61, 62 |
| \@GLSdesc@ | 34 |
| \@GLSpl@ | 47, 61, 62 |
| \@GLSplural@ | 48 |
| \@GLSsymbol@ | 35 |
| \@GLStext@ | 48 |
| \@GLSxtr@full | 123 |
| \@GLSxtr@fullpl | 124 |
| \@GLSxtr@p@acrlong@ | 48 |
| \@GLSxtr@p@acrlongpl@ | 48 |
| \@GLSxtr@p@acrshort@ | 48 |
| \@GLSxtr@p@acrshortpl@ | 48 |
| \@GLSxtr@p@long@ | 48 |
| \@GLSxtr@p@longpl@ | 48 |
| \@GLSxtr@p@plural@ | 47 |
| \@GLSxtr@p@short@ | 48 |
| \@GLSxtr@p@shortpl@ | 48 |
| \@GLSxtr@p@text@ | 47 |
| \@GLSxtrlong | 48, 127 |
| \@GLSxtrlongpl | 48, 131 |
| \@GLSxtrp | 55, 56 |
| \@GLSxtrshort | 48, 126 |
| \@GLSxtrshortpl | 48, 129 |
| \@Gls@ | 47, 61, 62 |
| \@Gls@acentryname | 69 |
| \@Gls@entry@field | 42, 54, 55 |
| \@Gls@entryname | 69 |
| \@GlsXtrEnableOnTheFly | 20 |
| \@Glspl@ | 47, 61, 62 |
| \@Glsplural@ | 48 |
| \@Glstext@ | 48 |
| \@Glsxtr | 21, 22 |
| \@Glsxtr@full | 122 |
| \@Glsxtr@fullpl | 124 |
| \@Glsxtr@p@acrlong@ | 48 |
| \@Glsxtr@p@acrlongpl@ | 48 |
| \@Glsxtr@p@acrshort@ | 48 |
| \@Glsxtr@p@acrshortpl@ | 48 |
| \@Glsxtr@p@long@ | 48 |
| \@Glsxtr@p@longpl@ | 48 |
| \@Glsxtr@p@plural@ | 47 |

| | | | |
|-------------------------|--|-------------------------------------|---------------------------|
| \@Glsxtr@p@short@ | 48 | \@glo@sorttype | 75 |
| \@Glsxtr@p@shortpl@ | 48 | \@glo@thisvalue | 163 |
| \@Glsxtr@p@text@ | 47 | \@glo@tmp | 42 |
| \@Glsxtrlong | 48, 127 | \@glo@type | 18, 69, 72, 76, 78, 81–83 |
| \@Glsxtrlongpl | 48, 130 | \@glo@types | 97, 193–199 |
| \@Glsxtrp | 54, 55 | \@glossary@default@style | 23, 202 |
| \@Glsxtrpl | 22 | \@gls@ | 47, 60, 61 |
| \@Glsxtrshort | 48, 125 | \@gls@@link | 29 |
| \@Glsxtrshortpl | 48, 128, 129 | \@gls@actualchar | 107 |
| \@acrlong | 48 | \@gls@adjustmode | 29 |
| \@acrlongpl | 48 | \@gls@alt@hyp@opt | 46 |
| \@acrshort | 48 | \@gls@alt@hyp@opt@char | 46 |
| \@acrshortpl | 48 | \@gls@alt@hyp@opt@keys | 46 |
| \@alt@gls@hyp@opt | 46 | \@gls@automake | 75 |
| \@auxout | 6, 26, 59, 68, 72, 73, 78, 82 | \@gls@checkedmkidx | 107–110 |
| \@cGLS | 62 | \@gls@checkmkidxchars | 106 |
| \@cGLS@ | 59, 62, 67 | \@gls@codepage | 78 |
| \@cGLSpl | 62 | \@gls@counter | 6, 30 |
| \@cGLSpl@ | 59, 62, 67 | \@gls@currentlettergroup | 83 |
| \@cGLspl@ | 59, 67 | \@gls@declareoption | 4 |
| \@cgls@ | 59, 61, 67 | \@gls@doautomake | 75 |
| \@cglspl@ | 59, 67 | \@gls@encapchar | 107 |
| \@disable@onlypremakeg | 73 | \@gls@entry@count | 59 |
| \@do@auxoutstuff | 78 | \@gls@entry@field | 42, 52–56, 58 |
| \@do@newglossaryentry | 70, 119 | \@gls@entry@unitcount | 67, 68 |
| \@do@seeglossary | 73 | \@gls@field@font | 30–37 |
| \@empty | 30, 38–41, 107, 122–131 | \@gls@field@link | 30–37, 42, 43 |
| \@end@glxstr@addunused | 18 | \@gls@hyp@opt | 42, 43, 46, 62, 122–131 |
| \@end@glxstr@gettype | 75, 76 | \@gls@hyp@opt@cs | 46 |
| \@end@glxstr@usesee | 17 | \@gls@increment@currcount | 58 |
| \@endfortrue | 134 | \@gls@increment@currunitcount | 66 |
| \@firstofone | 30, 100, 101, 106, 111 | \@gls@keymap | 7, 17, 42 |
| \@firstofthree | 30, 38–41, 46, 122, 123, 125, 127, 128, 130 | \@gls@label | 6, 45, 46, 73, 134 |
| \@firstoftwo | 31–35, 39, 41, 44, 46, 71, 115, 116, 122–124, 128–131, 168 | \@gls@levelchar | 107 |
| \@for | 13, 18, 57, 69, 72, 75, 83, 99, 111 | \@gls@link | 28, 29, 38–42, 122–131 |
| \@glo@assign@sortkey | 75 | \@gls@link@checkfirsthyper | 71 |
| \@glo@category | 63 | \@gls@link@nocheckfirsthyper | 28, 38–41, 122–131 |
| \@glo@counterprefix | 6 | \@gls@local@increment@currcount | 58 |
| \@glo@countunit | 63 | \@gls@local@increment@currunitcount | 66 |
| \@glo@default@sorttype | 75 | \@gls@loclist | 76, 77 |
| \@glo@label | 7, 17, 18, 42, 193–200 | \@gls@longpl | 117–119 |
| \@glo@loclist | 7 | \@gls@noidx@do | 83 |
| \@glo@name | 106, 107 | \@gls@noidx@nosanitizesort | 75 |
| \@glo@no@assign@sortkey | 76 | \@gls@noidx@sanitizesort | 74 |
| \@glo@parent | 194–196 | \@gls@noidxloclist@finalsep | 76 |
| \@glo@see | 17, 18 | \@gls@noidxloclist@prev | 76 |
| \@glo@sort | 106, 107 | \@gls@noidxloclist@sep | 76 |
| | | \@gls@noref@warn | 74 |

| | | | |
|---|---------------------|---|-----------------------------|
| <code>\@gls@org@glsnoidxdisplayloc</code> | 77 | <code>\@glsxtr@autoindex@level</code> | 107, 108 |
| <code>\@gls@org@glsseeformat</code> | 77 | <code>\@glsxtr@autoindex@setname</code> | 106 |
| <code>\@gls@preglossaryhook</code> | 111 | <code>\@glsxtr@autoindexcrossrefs</code> | 9, 17 |
| <code>\@gls@prevlevel</code> | 200, 201 | <code>\@glsxtr@cat</code> | 57, 69, 111 |
| <code>\@gls@quotechar</code> | 107 | <code>\@glsxtr@csname</code> | 64–67 |
| <code>\@gls@reference</code> | 19, 72, 73 | <code>\@glsxtr@current@style</code> | 23, 202 |
| <code>\@gls@saveentrycounter</code> | 7, 8, 14, 30 | <code>\@glsxtr@currentunitcount</code> | 64–67 |
| <code>\@gls@setdefault@glslink@opts</code> | 45 | <code>\@glsxtr@currunitcount</code> | 66, 68 |
| <code>\@gls@short</code> | 118 | <code>\@glsxtr@declareoption</code> | 4, 9, 10, 12 |
| <code>\@gls@shortpl</code> | 117–119 | <code>\@glsxtr@defaultnoglossarywarning</code> | 12 |
| <code>\@gls@tmpb</code> | 109, 110 | <code>\@glsxtr@deprecated@abbrstyle</code> | 152, 153, 155, 156, 161–163 |
| <code>\@gls@type</code> | 73–75, 134, 193–199 | <code>\@glsxtr@disabledflycommand</code> | 22 |
| <code>\@gls@write@entrycounts</code> | 59 | <code>\@glsxtr@do@@wrindex</code> | 45, 46 |
| <code>\@gls@write@entryunitcounts</code> | 67 | <code>\@glsxtr@do@glsdisablehyperinlist</code> | 44 |
| <code>\@gls@write@entryunitcounts@do</code> | 68 | <code>\@glsxtr@do@style</code> | 13, 184 |
| <code>\@glsabbrv@current@abbreviation</code> | 118, 131 | <code>\@glsxtr@do@titlecaps@warn</code> | 100–103, 111, 112 |
| <code>\@glsacronymlists</code> | 69 | <code>\@glsxtr@doabbreviationsdef</code> | 10 |
| <code>\@glsdisp</code> | 29 | <code>\@glsxtr@doaccsupp</code> | 12, 14 |
| <code>\@glsdoifexistsorwarn</code> | 8, 101, 103, 104 | <code>\@glsxtr@docdefval</code> | 8, 19 |
| <code>\@glsentry</code> | 59, 68 | <code>\@glsxtr@doloctag</code> | 25, 26 |
| <code>\@glslink</code> | 47 | <code>\@glsxtr@dorecord</code> | 6 |
| <code>\@glslocref</code> | 6 | <code>\@glsxtr@dostylewarn</code> | 134 |
| <code>\@glsnumberformat</code> | 6, 30, 105, 106 | <code>\@glsxtr@enabletagging</code> | 110 |
| <code>\@glsorder</code> | 72 | <code>\@glsxtr@end@</code> | 20 |
| <code>\@glspl@</code> | 47, 60, 62 | <code>\@glsxtr@endescspch</code> | 107–110 |
| <code>\@glsplural@</code> | 48 | <code>\@glsxtr@entrycount@org@localreset</code> | 58 |
| <code>\@gls punc@token</code> | 115, 116 | <code>\@glsxtr@entrycount@org@localunset</code> | 58 |
| <code>\@glsstyle@alttree</code> | 191 | <code>\@glsxtr@entrycount@org@reset</code> | 58 |
| <code>\@glsstyle@inline</code> | 191 | <code>\@glsxtr@entrycount@org@unset</code> | 58 |
| <code>\@glsstyle@listdotted</code> | 186 | <code>\@glsxtr@entryunitcount@org@localreset</code> | 67 |
| <code>\@gls target</code> | 47 | <code>\@glsxtr@entryunitcount@org@localunset</code> | 66 |
| <code>\@gls text@</code> | 48 | <code>\@glsxtr@entryunitcount@org@reset</code> | 66 |
| <code>\@gls widestname</code> | 192, 193, 200 | <code>\@glsxtr@entryunitcount@org@unset</code> | 66 |
| <code>\@glsxtr</code> | 21, 22 | <code>\@glsxtr@err@undefaction</code> | 5, 7 |
| <code>\@glsxtr@@do@@wrglossary</code> | 73 | <code>\@glsxtr@field@linkdefs</code> | 28 |
| <code>\@glsxtr@abbreviationsdef</code> | 10, 14 | <code>\@glsxtr@format@overridefalse</code> | 105 |
| <code>\@glsxtr@activate@initialtagging</code> | 111, 112 | <code>\@glsxtr@format@overridetrue</code> | 106 |
| <code>\@glsxtr@addunitcounter</code> | 63 | <code>\@glsxtr@foundinlist</code> | 116 |
| <code>\@glsxtr@addunusedxrefs</code> | 18 | <code>\@glsxtr@full</code> | 122 |
| <code>\@glsxtr@attrval</code> | 100–106 | <code>\@glsxtr@fullpl</code> | 123 |
| <code>\@glsxtr@autoindex@at</code> | 107, 108 | <code>\@glsxtr@gettype</code> | 75 |
| <code>\@glsxtr@autoindex@doextra@esc</code> | 107 | <code>\@glsxtr@glossdescfont</code> | 100, 101 |
| <code>\@glsxtr@autoindex@encap</code> | 106–108 | <code>\@glsxtr@glossnamefont</code> | 102–105 |
| <code>\@glsxtr@autoindex@esc</code> | 107, 109, 110 | <code>\@glsxtr@gobbleto@endescspch</code> | 109 |
| <code>\@glsxtr@autoindex@escat</code> | 107, 108 | <code>\@glsxtr@idx@displaynumberlist</code> | 74 |
| <code>\@glsxtr@autoindex@escencap</code> | 107, 108 | | |
| <code>\@glsxtr@autoindex@esclevel</code> | 107, 108 | | |
| <code>\@glsxtr@autoindex@escquote</code> | 107, 109 | | |

| | | | |
|--|----------|--|--------------|
| <code>\@glxstr@idx@entrynumberlist</code> | 74 | <code>\@glxstr@org@glxstrtitletext</code> | 168, 169 |
| <code>\@glxstr@ifcsstart</code> | 20 | <code>\@glxstr@org@makeglossaries</code> | 72 |
| <code>\@glxstr@ifpunctoken</code> | 116 | <code>\@glxstr@org@markboth</code> | 167 |
| <code>\@glxstr@ifunitcounter</code> | 63 | <code>\@glxstr@org@markright</code> | 167 |
| <code>\@glxstr@insert@dots</code> | 117 | <code>\@glxstr@org@newacronymstyle</code> | 71 |
| <code>\@glxstr@insert@dots@next</code> | 118 | <code>\@glxstr@org@postdescription</code> | 112 |
| <code>\@glxstr@insertdots</code> | 118 | <code>\@glxstr@org@setacronymstyle</code> | 70, 71 |
| <code>\@glxstr@label</code> | 18, 99 | <code>\@glxstr@org@printglossary</code> | 22, 75 |
| <code>\@glxstr@loadstyles</code> | 185 | <code>\@glxstr@org@warndep</code> | 117 |
| <code>\@glxstr@mixed@assign@sortkey</code> | 75 | <code>\@glxstr@p@acrlong@</code> | 48 |
| <code>\@glxstr@noidx@displaynumberlist</code> | 74 | <code>\@glxstr@p@acrlongpl@</code> | 48 |
| <code>\@glxstr@noidx@entrynumberlist</code> | 74 | <code>\@glxstr@p@acrshort@</code> | 48 |
| <code>\@glxstr@noidx@numberlistloop</code> | 74 | <code>\@glxstr@p@acrshortpl@</code> | 48 |
| <code>\@glxstr@notfoundinlist</code> | 116 | <code>\@glxstr@p@long@</code> | 48 |
| <code>\@glxstr@optlist</code> | 22 | <code>\@glxstr@p@longpl@</code> | 48 |
| <code>\@glxstr@org@GLS@</code> | 29 | <code>\@glxstr@p@plural@</code> | 47 |
| <code>\@glxstr@org@GLSpl@</code> | 29 | <code>\@glxstr@p@short@</code> | 47 |
| <code>\@glxstr@org@GLs@</code> | 28 | <code>\@glxstr@p@shortpl@</code> | 48 |
| <code>\@glxstr@org@GLspl@</code> | 29 | <code>\@glxstr@p@text@</code> | 47 |
| <code>\@glxstr@org@glxstrtitlefirst</code> | 168, 169 | <code>\@glxstr@pagetag</code> | 25, 26 |
| <code>\@glxstr@org@glxstrtitlefirstplural</code> | 168, 169 | <code>\@glxstr@pagetag</code> | 25, 26 |
| <code>\@glxstr@org@glxstrtitlefull</code> | 168, 169 | <code>\@glxstr@prevunitcount</code> | 66 |
| <code>\@glxstr@org@glxstrtitlefullpl</code> | 168, 169 | <code>\@glxstr@printglossopts</code> | 22, 75 |
| <code>\@glxstr@org@glxstrtitlelong</code> | 168, 169 | <code>\@glxstr@record</code> | 7, 8, 28, 29 |
| <code>\@glxstr@org@glxstrtitlelongpl</code> | 168, 169 | <code>\@glxstr@redefstyles</code> | 13, 184 |
| <code>\@glxstr@org@glxstrtitleplural</code> | 168, 169 | <code>\@glxstr@reg@glosslist</code> | 72–76 |
| <code>\@glxstr@org@glxstrtitleshort</code> | 168, 169 | <code>\@glxstr@savepreloctag</code> | 25, 26 |
| <code>\@glxstr@org@glxstrtitleshortpl</code> | 168, 169 | <code>\@glxstr@setentrycountunsetattr</code> | 57 |
| <code>\@glxstr@org@glxstrtitleshortpl</code> | 168, 169 | <code>\@glxstr@setentryunitcountunsetattr</code> | 69 |
| <code>\@glxstr@org@glxstrtitletext</code> | 168, 169 | <code>\@glxstr@setupshortcuts</code> | 11, 12, 14 |
| <code>\@glxstr@org@MakeUppercase</code> | 168, 169 | <code>\@glxstr@swaptwo</code> | 116 |
| <code>\@glxstr@org@checkfirsthyper</code> | 44, 71 | <code>\@glxstr@tag</code> | 111 |
| <code>\@glxstr@org@delimN</code> | 26 | <code>\@glxstr@taggingcs</code> | 111 |
| <code>\@glxstr@org@delimR</code> | 26 | <code>\@glxstr@thisloctag</code> | 26 |
| <code>\@glxstr@org@doseeglossary</code> | 73 | <code>\@glxstr@tmp</code> | 13 |
| <code>\@glxstr@org@glS@</code> | 28 | <code>\@glxstr@type</code> | 99 |
| <code>\@glxstr@org@glSdisp</code> | 29 | <code>\@glxstr@unitcountlist</code> | 63, 64 |
| <code>\@glxstr@org@glSignore</code> | 26 | <code>\@glxstr@usesee</code> | 17 |
| <code>\@glxstr@org@glSpl@</code> | 28 | <code>\@glxstr@warn@onexistssordo</code> | 5, 7, 8 |
| <code>\@glxstr@org@glxstrtitlefirst</code> | 168, 169 | <code>\@glxstr@warn@undefaction</code> | 5, 7, 8 |
| <code>\@glxstr@org@glxstrtitlefirstplural</code> | 168, 169 | <code>\@glxstr@docdeffalse</code> | 19 |
| <code>\@glxstr@org@glxstrtitlefull</code> | 168, 169 | <code>\@glxstr@indexcrossrefsfalse</code> | 9 |
| <code>\@glxstr@org@glxstrtitlefullpl</code> | 168, 169 | <code>\@glxstr@indexcrossrefstrue</code> | 9 |
| <code>\@glxstr@org@glxstrtitlelong</code> | 168, 169 | <code>\@glxstr@inmark</code> | 167 |
| <code>\@glxstr@org@glxstrtitlelongpl</code> | 168, 169 | <code>\@glxstr@long</code> | 48, 126 |
| <code>\@glxstr@org@glxstrtitleplural</code> | 168, 169 | <code>\@glxstr@longpl</code> | 48, 130 |
| <code>\@glxstr@org@glxstrtitleshort</code> | 168, 169 | <code>\@glxstr@notinmark</code> | 167 |
| <code>\@glxstr@org@glxstrtitleshortpl</code> | 168, 169 | <code>\@glxstr@p</code> | 53 |
| | | <code>\@glxstr@p@opt</code> | 51 |

| | | | |
|---|---|-----------------------------------|---|
| <code>\@glxstrpl</code> | 21, 22 | <code>short</code> | 146 |
| <code>\@glxstrpostloctag</code> | 25, 27 | <code>\abbreviationsname</code> | 9 |
| <code>\@glxstrpreloctag</code> | 25, 27 | <code>\abbrvpluralsuffix</code> | 118, 119, 137, 139, 141, 143, 145, 146, 148, 150–156, 164, 165 |
| <code>\@glxstrshort</code> | 47, 125 | <code>\ABP</code> | 10 |
| <code>\@glxstrshortpl</code> | 48, 128 | <code>\Abp</code> | 10 |
| <code>\@glxstrundeftag</code> | 5, 14 | <code>\abp</code> | 10 |
| <code>\@gobble</code> | 5, 7, 9, 30, 118 | <code>\ACRfullfmt</code> | 70 |
| <code>\@gobbletwo</code> | 117 | <code>\Acrfullfmt</code> | 70 |
| <code>\@ifnextchar</code> | 46 | <code>\acrfullfmt</code> | 70 |
| <code>\@ifpackageloaded</code> | 4, 9, 83, 100, 101, 104, 105, 184 | <code>\ACRfullplfmt</code> | 70 |
| <code>\@ifstar</code> | 20, 46, 110 | <code>\Acrfullplfmt</code> | 70 |
| <code>\@ifundefined</code> | 184 | <code>\acrfullplfmt</code> | 70 |
| <code>\@input@</code> | 78 | <code>\acronymentry</code> | 70 |
| <code>\@istfilename</code> | 72 | <code>\acronymfont</code> | 38–41, 50, 71 |
| <code>\@makeglossary</code> | 72, 73 | <code>\acronymname</code> | 9 |
| <code>\@mfu@domakefirsttuc</code> | 111, 112 | <code>\acronymsort</code> | 70 |
| <code>\@mfu@nocaplist</code> | 112 | <code>\acronymtype</code> | 10, 69–71 |
| <code>\@ne</code> | 59, 68 | <code>\acrpluralsuffix</code> | 70 |
| <code>\@newglossaryentry@defcounters</code> | 58, 65 | <code>\actualchar</code> | 109 |
| <code>\@newglossaryentryposthook</code> | 7, 42 | <code>\addtolength</code> | 201 |
| <code>\@newglossaryentryprehook</code> | 7, 42 | <code>\advance</code> | 59, 68 |
| <code>\@nnil</code> | 107, 109, 110, 116–118 | <code>\AF</code> | 10 |
| <code>\@no@makeglossaries</code> | 82 | <code>\Af</code> | 10 |
| <code>\@onelevel@sanitize</code> | 22 | <code>\af</code> | 10 |
| <code>\@onlypreamble</code> | 23, 26, 68, 82, 106, 108–110 | <code>\AFP</code> | 10 |
| <code>\@print@unsrt@glossary</code> | 82 | <code>\Afp</code> | 10 |
| <code>\@printgloss@setsort</code> | 75 | <code>\afp</code> | 10 |
| <code>\@printglossary</code> | 22, 75, 82 | <code>\AL</code> | 10 |
| <code>\@sGlsXtrEnableOnTheFly</code> | 20 | <code>\Al</code> | 10 |
| <code>\@secondofthree</code> | 31– 33, 38–41, 43, 122, 124, 125, 127, 129, 130 | <code>\al</code> | 10 |
| <code>\@secondoftwo</code> | 30, 33–41, 44, 47, 71, 116, 122, 123, 125–131, 143, 168, 169 | <code>\ALP</code> | 10 |
| <code>\@thirdofthree</code> | 31– 33, 38–41, 43, 123, 124, 126, 128, 129, 131 | <code>\Alp</code> | 10 |
| <code>\@thirdoftwo</code> | 33–37 | <code>\alp</code> | 10 |
| <code>\@warn@nomakeglossaries</code> | 78 | <code>\AnyTrackedLanguages</code> | 184 |
| <code>\@xdy@main@language</code> | 78 | <code>\appto</code> | 7, 13, 17, 42, 45, 46, 58, 65, 106, 115, 118 |
| <code>\@xdy@language</code> | 78 | <code>\AS</code> | 10 |
| <code>_</code> | 80 | <code>\As</code> | 10 |
| A | | <code>\as</code> | 10 |
| <code>\AB</code> | 10 | <code>\ASP</code> | 10 |
| <code>\Ab</code> | 10 | <code>\Asp</code> | 10 |
| <code>\ab</code> | 10 | <code>\asp</code> | 10 |
| abbreviation styles: | | <code>\AtBeginDocument</code> | 14, 24 |
| long-noshort | 161 | <code>\AtEndDocument</code> | 17, 59, 67, 78 |
| | | B | |
| | | babel package | 106, 108, 115 |
| | | <code>\begin</code> | 61, 80, 83, 186–190, 200 |
| | | <code>\begingroup</code> | 6 |

| C | |
|------------------------------|--|
| category attributes: | |
| discardperiod | 114 |
| entrycount | 56, 57, 59, 68, 69 |
| firsttuc | 103 |
| glossdesc | 100 |
| glossdescfont | 100 |
| glossname | 101 |
| glossnamefont | 101, 104 |
| headuc | 169 |
| indexname | 106 |
| indexonlyfirst | 45 |
| insertdots | 118, 119 |
| nohyper | 44 |
| nohyperfirst | 31–33 |
| regular | 27, 62, 136–141, 143, 145, 147, 148, 158, 159, 164, 165 |
| \cGLS | 10, 57, 69 |
| \cGls | 10, 57, 69 |
| \cglS | 10, 57, 69 |
| \cGLSformat | 61 |
| \cGlsformat | 60 |
| \cglSformat | 60, 62 |
| \cGLSpl | 10, 57, 69 |
| \cGlspl | 10, 57, 69 |
| \cglSpl | 10, 57, 69 |
| \cGLSplformat | 61 |
| \cGlsplformat | 61 |
| \cglSplformat | 60, 62 |
| \columnwidth | 24 |
| \count@ | 59, 68 |
| \cs | 61, 200 |
| \csdef | 42, 43, 58, 64, 65, 67, 94, 134–136, 143, 186 |
| \csedef | 65 |
| \csgdef | 19, 25, 58, 59, 64, 66, 67 |
| \csletcs | 135, 136 |
| \csname | 6, 23, 27, 30, 38–43, 51, 64, 65, 73, 78, 81–83, 99, 117, 122–131, 136, 200 |
| \csuse | 25, 42, 43, 53, 54, 63–67, 94, 105, 113, 134, 136, 193–196 |
| \csxdef | 17, 64, 67 |
| \CurrentOption | 13, 185 |
| \CurrentTrackedTag | 184 |
| \CustomAbbreviationFields | 119, 137–141, 143, 144, 146, 147, 149, 157, 159, 161, 164, 165 |
| D | |
| \DeclareAcronymList | 69 |
| \DeclareOption | 4, 185 |
| \DeclareOptionX | 4, 13 |
| \def | 6–8, 14, 17, 18, 20–22, 24, 27–41, 46–51, 60–62, 69, 73, 75, 76, 83, 105–112, 115–119, 122–131, 134, 200, 201 |
| \define@boolkey | 9, 44 |
| \define@choicekey | 5, 7, 8, 11, 12 |
| \define@key | 7, 13, 42, 117 |
| \DefineAcronymSynonyms | 11, 12 |
| \delimN | 26 |
| \delimR | 26 |
| \detokenize | 20 |
| \dimen@ | 72, 193–200 |
| \dimen@i | 194–196 |
| \dimen@ii | 194–196 |
| \dimexpr | 24, 192 |
| \disable@keys | 10, 14, 19 |
| \do | 13, 18, 57, 69, 72, 75, 83, 99, 111 |
| \do@glS@link@checkfirsthyper | 28, 29, 38–41, 122–131 |
| \do@glSdisablehyperinlist | 45 |
| doc package | 109 |
| \DTLifinlist | 73, 74, 76 |
| E | |
| \eappto | 13, 106, 185 |
| \edef | 6, 15, 30, 44, 63–67, 72–74, 76, 78, 100, 101, 103–105, 107, 109, 110, 117, 194–196 |
| \eglssetwidest | 193–200 |
| \else | 6, 7, 9, 10, 12, 19, 20, 25, 27, 45, 60, 72, 75, 79–81, 106, 107, 109, 110, 116, 118, 125–131, 137–148, 164–166, 187–191, 201, 202 |
| \emph | 157 |
| \encapchar | 109 |
| \end | 80, 83, 186–190, 200 |
| \endcsname | 6, 23, 27, 30, 38–43, 51, 64, 65, 73, 78, 81–83, 99, 117, 122–131, 136, 200 |
| \endgroup | 6 |
| entry categories: | |
| abbreviation | 131 |
| general | 94, 96 |
| index | 98 |
| \epreto | 107 |
| \equal | 81 |
| etoolbox package | 4 |
| \expandafter | 13, 17, 18, 20–22, 42, 43, 46, 51, 62, 63, 73–76, 83, 99, 102, 104, 106, 109, 116, 118, 119 |
| \expandonce | 70, 107 |

| | |
|--|---|
| F | |
| <code>\fi</code> | 5–10, 12, 17, 19, 20, 23–25, 27, 45, 59, 60, 67, 68, 72, 75, 78–82, 106–108, 110, 116, 118, 125– 131, 137–148, 164–166, 187–191, 193–202 |
| first use | 203 |
| flag | 203 |
| text | 203 |
| <code>\firstacronymfont</code> | 71, 72 |
| <code>\footnote</code> | 141 |
| <code>\forall glossaries</code> ... | 18, 82, 97, 99, 193–199 |
| <code>\forall glsentries</code> | 59, 68 |
| <code>\ForEachTrackedDialect</code> | 184 |
| <code>\forall glsentries</code> | 18, 97, 99, 193–199 |
| <code>\forall listcsloop</code> | 68 |
| <code>\forall listloop</code> | 76, 77, 112 |
| <code>\futurelet</code> | 115 |
| G | |
| <code>\gdef</code> | 26, 108, 109 |
| <code>\Genacrfullformat</code> | 70 |
| <code>\genacrfullformat</code> | 70 |
| <code>\GenericAcronymFields</code> | 70 |
| <code>\Genplacrfullformat</code> | 70 |
| <code>\genplacrfullformat</code> | 70 |
| <code>\glo@name</code> | 102–104 |
| glossaries package | 185 |
| glossaries-accsupp package | 12, 14, 83 |
| glossaries-extra package | 2 |
| glossaries-extra-stylemods package . | 12, 113, 184 |
| <code>\GlossariesExtraWarning</code> | 5, 9, 20, 22, 71, 74, 80, 82, 83, 100–105, 111, 136 |
| <code>\GlossariesExtraWarningNoLine</code> . | 9, 59, 68 |
| <code>\GlossariesWarning</code> | 25, 74, 77, 134 |
| <code>\GlossariesWarningNoLine</code> | 73, 78 |
| glossary styles: | |
| alttree | 191, 192, 200 |
| inline | 191 |
| listdotted | 186 |
| listdottedstyle | 186 |
| sublistdotted | 186 |
| glossary-long package | 187 |
| glossary-longbooktabs package | 187 |
| glossary-mcols package | 185 |
| glossary-tree package | 185 |
| <code>\glossaryentrynumbers</code> | 27 |
| <code>\glossaryheader</code> | 83, 186–190, 201 |
| <code>\glossarypostamble</code> | 83 |
| <code>\glossarypreamble</code> | 83 |
| <code>\glossarysection</code> | 81, 83 |
| <code>\glossarytitle</code> | 81, 83 |
| <code>\glossarytoctitle</code> | 81, 83 |
| <code>\glossentry</code> | 186–190, 201 |
| <code>\glossentrydesc</code> | 186–192 |
| <code>\glossentryname</code> | 186–190, 201 |
| <code>\glossentrysymbol</code> | 187–192 |
| <code>\glossxtrsetpopts</code> | 112 |
| <code>\GLS</code> | 57, 69 |
| <code>\Gls</code> | 21, 57, 69 |
| <code>\gls</code> | 21, 23, 57, 69, 74, 79 |
| <code>\gls@assign@field</code> | 7, 42 |
| <code>\gls@checkseeallowed</code> | 19, 73 |
| <code>\gls@codepage</code> | 78 |
| <code>\gls@defdocnewglossaryentry</code> | 58, 65 |
| <code>\gls@defglossaryentry</code> | 21, 22 |
| <code>\gls@noidxglossary</code> | 73 |
| <code>\gls@save@numberlist</code> | 24, 25, 27 |
| <code>\gls@tmplen</code> | 193–199, 201 |
| <code>\gls@type</code> | 73 |
| <code>\glsabbrvdefaultfont</code> | 121, 137, 139, 141, 143, 145, 146, 148 |
| <code>\glsabbrvemfont</code> | 157–163 |
| <code>\glsabbrvfont</code> | 49, 71, 121, 125, 126, 128, 129, 132, 137–141, 143–146, 148–166 |
| <code>\glsabbrvuserfont</code> | 163, 164, 166 |
| <code>\glsabrvfont</code> | 137, 139, 141, 143, 158, 159, 164, 165 |
| <code>\GLSaccessdesc</code> | 34 |
| <code>\Glsaccessdesc</code> | 34, 100, 110 |
| <code>\glsaccessdesc</code> | 34, 100, 114 |
| <code>\GLSaccessdescplural</code> | 34 |
| <code>\Glsaccessdescplural</code> | 34 |
| <code>\glsaccessdescplural</code> | 34 |
| <code>\GLSaccessfirst</code> | 32 |
| <code>\Glsaccessfirst</code> | 31 |
| <code>\glsaccessfirst</code> | 31 |
| <code>\GLSaccessfirstplural</code> | 33 |
| <code>\Glsaccessfirstplural</code> | 33 |
| <code>\glsaccessfirstplural</code> | 32 |
| <code>\Glsaccesslong</code> 40, 120, 127, 138, 145, 148, 165 | |
| <code>\glsaccesslong</code> | 40, 120, 127, 128, 137, 139, 141, 142, 144–148, 164, 166 |
| <code>\Glsaccesslongpl</code> | 41, 120, 130, 138, 145, 148, 165 |
| <code>\glsaccesslongpl</code> | 41, 120, 130, 131, 137, 139, 140, 142, 144–148, 164, 166 |
| <code>\GLSaccessname</code> | 33 |
| <code>\Glsaccessname</code> | 33 |

| | | | |
|--|-------------------------|--|------------------------------|
| <code>\glsaccessname</code> | 33 | <code>\glsenableentrycount</code> | 57, 59, 67 |
| <code>\GLSaccessplural</code> | 32 | <code>\glsenableentryunitcount</code> | 59, 68 |
| <code>\Glsaccessplural</code> | 32 | <code>\glsentrycurrcount</code> | 58, 59, 66 |
| <code>\glsaccessplural</code> | 32 | <code>\Glsentrydesc</code> | 88, 92, 101 |
| <code>\Glsaccessshort</code> | | <code>\glsentrydesc</code> | 87, 88, 92, 101 |
| 38, 126, 132, 139, 142, 144, 147, 166 | | <code>\Glsentrydescplural</code> | 88, 93 |
| <code>\glsaccessshort</code> | 38, 120, | <code>\glsentrydescplural</code> | 88, 92, 93 |
| 125, 126, 132, 137–139, 141–148, 164–166 | | <code>\Glsentryfirst</code> | 63, 85, 91 |
| <code>\Glsaccessshortpl</code> | | <code>\glsentryfirst</code> | 63, 85, 86, 91, 180 |
| 39, 129, 132, 140, 142, 144, 147, 166 | | <code>\Glsentryfirstplural</code> | 63, 86, 92 |
| <code>\glsaccessshortpl</code> | 39, 120, | <code>\glsentryfirstplural</code> ... | 63, 86, 91, 92, 181 |
| 128, 129, 132, 137–139, 142–148, 164–166 | | <code>\Glsentryfull</code> | 70 |
| <code>\GLSaccesssymbol</code> | 35 | <code>\glsentryfull</code> | 70 |
| <code>\Glsaccesssymbol</code> | 35, 110 | <code>\Glsentryfullpl</code> | 70 |
| <code>\glsaccesssymbol</code> | 35, 110, 114 | <code>\glsentryfullpl</code> | 70 |
| <code>\GLSaccesssymbolplural</code> | 35 | <code>\glsentryitem</code> | 186–190, 201 |
| <code>\Glsaccesssymbolplural</code> | 35 | <code>\Glsentrylong</code> | 50, 63, 90, 93 |
| <code>\glsaccesssymbolplural</code> | 35 | <code>\glsentrylong</code> 49, 50, 63, 90, 93, 143, 181, 182 | |
| <code>\GLSaccessesstext</code> | 31 | <code>\Glsentrylongpl</code> | 50, 51, 63, 90, 93 |
| <code>\Glsaccessesstext</code> | 31 | <code>\glsentrylongpl</code> ... | 50, 51, 63, 90, 93, 94, 182 |
| <code>\glsaccessesstext</code> | 30 | <code>\Glsentryname</code> | 84, 91, 102–105 |
| <code>\glsacrshortcutstrue</code> | 11, 12 | <code>\glsentryname</code> | 83, 84, 91, 106, 193–200 |
| <code>\glsacspacemax</code> | 72 | <code>\glsentrynumberlist</code> | 74, 77, 198–200 |
| <code>\glsadd</code> | 18, 79 | <code>\Glsentryplural</code> | 85, 91 |
| <code>\glsaddstoragekey</code> | 94 | <code>\glsentryplural</code> | 85, 91, 180 |
| <code>\glsbackslash</code> | 20 | <code>\glsentryprevcount</code> | 58, 60, 66 |
| <code>\glscapscase</code> | 30–41, 43, 122–133 | <code>\glsentryprevmaxcount</code> | 66 |
| <code>\glscategory</code> .. 27, 30, 44, 49, 95–97, 100, | | <code>\glsentryprevtotalcount</code> | 66 |
| 101, 103–105, 110, 113, 122–126, 128, 129 | | <code>\Glsentryshort</code> | 49, 50, 89, 93 |
| <code>\glscategorylabel</code> | 44, 117–119, 143 | <code>\glsentryshort</code> | 49, 50, 72, 89, 93, 178 |
| <code>\glsclsebrace</code> | 80, 81 | <code>\Glsentryshortpl</code> | 49, 50, 89, 93 |
| <code>\glscurrententrylabel</code> | | <code>\glsentryshortpl</code> . | 49, 50, 89, 90, 93, 178, 179 |
| 25, 26, 83, 105, 112, 113 | | <code>\Glsentrysymbol</code> | 86, 92 |
| <code>\glscurrentfieldvalue</code> | 163 | <code>\glsentrysymbol</code> | 86, 87, 92, 197, 198 |
| <code>\glscustomtext</code> | 28, 38–41, 122–131, 133 | <code>\Glsentrysymbolplural</code> | 87, 92 |
| <code>\glsdefaulttype</code> | 9, 75, 79, 82 | <code>\glsentrysymbolplural</code> | 87, 92 |
| <code>\glsdescriptionaccessdisplay</code> . | 87, 88, 100 | <code>\Glsentrytext</code> | 84, 91 |
| <code>\glsdescriptionpluralaccessdisplay</code> . | 88 | <code>\glsentrytext</code> | 84, 91, 179 |
| <code>\glsdescwidth</code> | 186, 188–190 | <code>\Glsentryuseri</code> | 36 |
| <code>\glsdetoklabel</code> | 15–20, 30, 58, | <code>\glsentryuseri</code> | 36 |
| 59, 64–68, 73, 76, 77, 99, 102–104, 194–196 | | <code>\Glsentryuserii</code> | 36 |
| <code>\glsdisplaynumberlist</code> | 74, 76 | <code>\glsentryuseriii</code> | 36 |
| <code>\glsdohyperlink</code> | 47 | <code>\glsentryuseriii</code> | 36 |
| <code>\glsdoifexists</code> | | <code>\glsentryuseriii</code> | 36 |
| 8, 17, 28, 29, 38–41, 76, 77, 122–131 | | <code>\Glsentryuseriv</code> | 37 |
| <code>\glsdoifexistsordo</code> | 29 | <code>\glsentryuseriv</code> | 37 |
| <code>\glsdoifexistsorwarn</code> | 9, 100, 101, 110 | <code>\Glsentryuserv</code> | 37 |
| <code>\glsdonohyperlink</code> | 47 | <code>\glsentryuserv</code> | 37 |
| <code>\glsdosanitizesort</code> | 74 | <code>\Glsentryuservi</code> | 37 |

| | | | |
|---|---|--|--|
| <code>\glsentryuservi</code> | 37 | <code>\glskeylisttok</code> | 70, 118, 119 |
| <code>\glsfieldxdef</code> | 99 | <code>\glslabel</code> 15, 27, 44, 71, 113, 114, 131–133, 143 | |
| <code>\glsfindwidesttoplevelname</code> | 193 | <code>\glslabeltok</code> | 70, 118, 119, 137–141, 143, 145–147, 149, 157–159, 161, 164, 165 |
| <code>\GLSfirst</code> | 173 | <code>\glsletentryfield</code> | 106 |
| <code>\Glsfirst</code> | 173 | <code>\glslink</code> | 70 |
| <code>\glsfirst</code> | 173 | <code>\glslink options</code> | |
| <code>\glsfirstabbrvdefaultfont</code> | | format | 105 |
| 121, 137, 139, 141, 143, 145, 146, 148 | | hyper | 167 |
| <code>\glsfirstabbrvemfont</code> | 158–163 | noindex | 6, 44, 167 |
| <code>\glsfirstabbrvfont</code> | | <code>\glslinkcheckfirsthyperhook</code> | 44 |
| 71, 120, 137–148, 150–156, 158–166 | | <code>\glslinkvar</code> | 46 |
| <code>\glsfirstabbrvuserfont</code> | 164, 166 | <code>\glslistdottedwidth</code> | 186 |
| <code>\glsfirstaccessdisplay</code> | 85 | <code>\glslongaccessdisplay</code> | 90 |
| <code>\glsfirstlongdefaultfont</code> | | <code>\glslongdefaultfont</code> | |
| 137, 139, 145, 146, 148 | | 121, 137, 139, 140, 145, 146, 148 | |
| <code>\glsfirstlongemfont</code> | 158–162 | <code>\glslongemfont</code> | 157–162 |
| <code>\glsfirstlongfont</code> | 120, 137–141, 143, 145–149, 157–162, 164–166 | <code>\glslongfont</code> | |
| <code>\glsfirstlongfootnotefont</code> | 141–144 | .. 50, 121, 127, 128, 130, 131, 137, 139, 141, 143, 145, 146, 148, 158–162, 164, 166 | |
| <code>\glsfirstlonguserfont</code> | 164, 166 | <code>\glslongfootnotefont</code> | 140, 141, 143 |
| <code>\GLSfirstplural</code> | 174 | <code>\glslongpltok</code> | 119, 137–141, 147, 149, 158, 159, 161, 164, 165 |
| <code>\Glsfirstplural</code> | 174 | <code>\glslongpluralaccessdisplay</code> | 90 |
| <code>\glsfirstplural</code> | 174 | <code>\glslongtok</code> | 70, 118, 119, 137–141, 143, 145–147, 149, 157–159, 161, 164, 165 |
| <code>\glsfirstpluralaccessdisplay</code> | 86 | <code>\glslonguserfont</code> | 164–166 |
| <code>\glsforeachincategory</code> | 134 | <code>\glsnameaccessdisplay</code> | 83, 84, 102, 104 |
| <code>\glsgenentryfmt</code> | 27 | <code>\glsnamefont</code> | 102–105 |
| <code>\glsgetattribute</code> 60, 64–66, 100, 101, 103–106 | | <code>\glsnoidxdisplayloc</code> | 77 |
| <code>\glsgetcategoryattribute</code> | 95 | <code>\glsnoidxdisplaylocclsthandler</code> | 76 |
| <code>\glsgetwidestname</code> | 192 | <code>\glsnoidxloclist</code> | 77 |
| <code>\glsgroupheading</code> | 186–190, 201 | <code>\glsnoidxnumberlistloophandler</code> | 77 |
| <code>\glsgroupskip</code> | 187–191, 202 | <code>\glsnonumberlistfalse</code> | 25 |
| <code>\glschasattribute</code> 59, 60, 64–68, 100, 101, 103–106, 137–141, 143, 158, 159, 164, 165 | | <code>\glsnonumberlisttrue</code> | 25 |
| <code>\glschascategoryattribute</code> | 95 | <code>\glsnumberlistloop</code> | 74 |
| <code>\glshypernumber</code> | 106 | <code>\glsnumlistlastsep</code> | 76 |
| <code>\glsifattribute</code> | 31, 44, 45, 52, 97, 100–103, 112, 114, 115, 169–177 | <code>\glsnumlistsep</code> | 76 |
| <code>\glsifcategory</code> | 97 | <code>\glsopenbrace</code> | 80, 81 |
| <code>\glsifcategoryattribute</code> .. | 44, 96, 118, 119 | <code>\glsorder</code> | 72 |
| <code>\glsifnotregular</code> | 30 | <code>\glspagelistwidth</code> | 186, 188–190 |
| <code>\glsifnotregularcategory</code> | 97 | <code>\GLSpl</code> | 57, 69 |
| <code>\glsifplural</code> | | <code>\Glspl</code> | 22, 57, 69 |
| 30, 32–35, 38–41, 114, 115, 122–132 | | <code>\glspl</code> | 22, 57, 69 |
| <code>\glsifregular</code> | 27, 30, 63 | <code>\GLSplural</code> | 172 |
| <code>\glsifregularcategory</code> | 96 | <code>\Glsplural</code> | 172, 173 |
| <code>\glsignore</code> | 26 | <code>\glsplural</code> | 172 |
| <code>\glsinlinedescformat</code> | 191 | <code>\glspluralaccessdisplay</code> | 84, 85 |
| <code>\glsinlinesubdescformat</code> | 191 | | |
| <code>\glsinsert</code> | 30, 38–41, 122–133 | | |

| | | | |
|---|--|--|---|
| <code>\glspluralsuffix</code> | 118, 121, 137, 139, 141, 143, 145, 146, 148, 149, 153, 164 | <code>\glsxtr@indexonly@saveentrycounter</code> | 7, 8, 14 |
| <code>\glspostdescription</code> | 112, 186–192 | <code>\glsxtr@keylist</code> | 21, 22 |
| <code>\glspostinline</code> | 191 | <code>\glsxtr@makeglossaries</code> | 72 |
| <code>\glspostlinkhook</code> . | 28, 29, 38–42, 51, 122–131 | <code>\glsxtr@next</code> | 116 |
| <code>\glsprestandardsort</code> | 74 | <code>\glsxtr@orgmakenoidxglossaries</code> .. | 18, 19 |
| <code>\glsresetentrylist</code> | 83 | <code>\glsxtr@punclist</code> | 115, 116 |
| <code>\glsseeformat</code> | 17, 73, 77 | <code>\glsxtr@record</code> | 6 |
| <code>\glssetabbrvfmt</code> | 27, 30, 49, 100, 101, 103–105, 110, 122–126, 128, 129 | <code>\glsxtr@resource</code> | 82 |
| <code>\glssetattribute</code> | 137–141, 143, 145–147, 149, 158, 159, 161, 164, 165 | <code>\glsxtr@saveentrycounter</code> | 6, 7 |
| <code>\glssetcategoryattribute</code> | 57, 69, 71, 95, 96, 98, 99, 111 | <code>\glsxtr@setup@record</code> | 7, 8, 14 |
| <code>\glsshortaccessdisplay</code> | 89 | <code>\glsxtr@usesee</code> | 17 |
| <code>\glsshortpltok</code> | 119, 137–141, 143–146, 158, 159, 164, 165 | <code>\glsxtr@warnonexistsordo</code> | 5, 7, 8, 16 |
| <code>\glsshortpluralaccessdisplay</code> | 89, 90 | <code>\glsxtrabbrvfootnote</code> | 141–143 |
| <code>\glsshorttok</code> | 70, 118, 119, 137–141, 143, 144, 146, 149, 157–159, 161, 164, 165 | <code>\glsxtrabbrvtype</code> | 9, 10, 119 |
| <code>\glssubentryitem</code> | 186–191, 201 | <code>\glsxtraddallcrossrefs</code> | 17 |
| <code>\glsymbolaccessdisplay</code> | 86, 87 | <code>\glsxtrAltTreeIndent</code> | 192 |
| <code>\glsymbolpluralaccessdisplay</code> | 87 | <code>\glsxtrAlttreeInit</code> | 200 |
| <code>\glstarget</code> | 186–191, 201 | <code>\glsxtrAltTreePar</code> | 191 |
| <code>\GLstext</code> | 171, 172 | <code>\glsxtrAltTreeSetHangIndent</code> ... | 192, 201 |
| <code>\Glstext</code> | 172 | <code>\glsxtrAltTreeSetSubHangIndent</code> | 201 |
| <code>\glstext</code> | 171 | <code>\glsxtrAlttreeSubSymbolDescLocation</code> | 201 |
| <code>\glstextaccessdisplay</code> | 84 | <code>\glsxtrAlttreeSymbolDescLocation</code> .. | 192, 201 |
| <code>\glstextformat</code> | 29 | <code>\glsxtrassignfieldfont</code> | 30–37 |
| <code>\glstextup</code> | 149 | <code>\glsxtrcat</code> | 21, 22 |
| <code>\glstreeindent</code> | 200, 201 | <code>\glsxtrchecknohyperfirst</code> | 31–33 |
| <code>\glstreenamebox</code> | 201 | <code>\glsxtrComputeTreeIndent</code> | 201 |
| <code>\glstreenamefmt</code> | 192–201 | <code>\glsxtrComputeTreeSubIndent</code> | 201 |
| <code>\glstype</code> | 38–42, 122–131 | <code>\GlsXtrDefineAbbreviationShortcuts</code> | 11, 12 |
| <code>\glsunset</code> | 18, 60, 61 | <code>\GlsXtrDefineOtherShortcuts</code> | 11, 12 |
| <code>\glswrite</code> | 72 | <code>\glsxtrdiscardperiod</code> | 113 |
| <code>\glswriteentry</code> | 6 | <code>\glsxtrdoautoindexname</code> | 45, 46, 105 |
| <code>\Glsxtr</code> | 22 | <code>\glsxtrdopostpunc</code> | 143 |
| <code>\glsxtr</code> | 22 | <code>\glsxtrdowrglossaryhook</code> | 45, 46 |
| <code>\glsxtr@do@wrglossary</code> | 6–8 | <code>\GlsXtrEnableEntryCounting</code> | 69 |
| <code>\glsxtr@addloclistfield</code> | 8 | <code>\GlsXtrEnableEntryUnitCounting</code> | 57 |
| <code>\glsxtr@addunused</code> | 18 | <code>\GlsXtrEnableOnTheFly</code> | 20, 23 |
| <code>\glsxtr@applyabbrvfmt</code> | 131 | <code>\glsxtrfieldtitlecase</code> | 100–103 |
| <code>\glsxtr@applyabbrvstyle</code> | 117, 118, 134 | <code>\glsxtrfieldtitlecasecs</code> | 99 |
| <code>\glsxtr@doooption</code> | 4, 9, 13, 14 | <code>\glsxtrfirstscfont</code> | 150–153 |
| <code>\glsxtr@headentry@p</code> | 52, 53 | <code>\glsxtrfirstsmfont</code> | 153–156 |
| <code>\glsxtr@ifnextpunc</code> | 115 | <code>\GlsXtrFormatLocationList</code> | 25, 27, 198–200 |
| <code>\glsxtr@ifpunctoken</code> | 116 | <code>\GLSxtrfull</code> | 10, 176, 177 |
| | | <code>\Glsxtrfull</code> | 10, 177 |
| | | <code>\glsxtrfull</code> | 10, 176 |
| | | <code>\Glsxtrfullformat</code> | 120, 133, 135, 138, 139, 142, 144, 145, 147, 148, 165, 166 |

| | |
|---|--|
| <code>\glxstrfullformat</code> | 120, 133, 135, 137–141, 143, 145, 147, 148, 164, 166 |
| <code>\GLSxtrfullpl</code> | 10, 177 |
| <code>\Glsxtrfullpl</code> | 10, 177, 178 |
| <code>\glxstrfullpl</code> | 10, 177 |
| <code>\Glsxtrfullplformat</code> | 121, 133, 135, 138, 140, 142, 144, 146–148, 165, 166 |
| <code>\glxstrfullplformat</code> | 132, 133, 135, 137, 139, 141, 143, 145, 147, 148, 164, 166 |
| <code>\glxstrfullsep</code> | 120, 137–140, 142, 144–148, 157–159, 163 |
| <code>\glxstrgenabbrvfmt</code> | 27 |
| <code>\Glsxtrheadfirst</code> | 168 |
| <code>\glxtrheadfirst</code> | 168 |
| <code>\Glsxtrheadfirstplural</code> | 168 |
| <code>\glxtrheadfirstplural</code> | 168 |
| <code>\Glsxtrheadfull</code> | 169 |
| <code>\glxtrheadfull</code> | 169 |
| <code>\Glsxtrheadfullpl</code> | 169 |
| <code>\glxtrheadfullpl</code> | 169 |
| <code>\Glsxtrheadlong</code> | 169 |
| <code>\glxtrheadlong</code> | 169 |
| <code>\Glsxtrheadlongpl</code> | 169 |
| <code>\glxtrheadlongpl</code> | 169 |
| <code>\Glsxtrheadplural</code> | 168 |
| <code>\glxtrheadplural</code> | 168 |
| <code>\Glsxtrheadshort</code> | 168 |
| <code>\glxtrheadshort</code> | 168 |
| <code>\Glsxtrheadshortpl</code> | 168 |
| <code>\glxtrheadshortpl</code> | 168 |
| <code>\Glsxtrheadtext</code> | 168 |
| <code>\glxtrheadtext</code> | 168 |
| <code>\glxstrifcounttrigger</code> | 60, 61 |
| <code>\glxstrifemptyglossary</code> | 81 |
| <code>\glxstrifindexing</code> | 45 |
| <code>\glxstrifinmark</code> | 53–55, 168, 169 |
| <code>\glxstrifnextpunc</code> | 115, 116 |
| <code>\glxstrifperiod</code> | 114, 115 |
| <code>\glxstrifwasfirstuse</code> | 30– 33, 38–41, 44, 71, 114, 122, 125–131, 143 |
| <code>\Glsxtrinlinefullformat</code> | 121, 123, 135, 142, 144, 145, 147, 148, 183 |
| <code>\glxtrinlinefullformat</code> | 121–123, 135, 142, 144–146, 148, 182 |
| <code>\Glsxtrinlinefullplformat</code> | 121, 124, 135, 142, 144, 145, 147, 148, 183 |
| <code>\glxtrinlinefullplformat</code> | 120, 121, 123, 124, 135, 142, 144–146, 148, 183 |
| <code>\glxtrinsertinsidfalse</code> | 137 |
| <code>\GLSxtrlong</code> | 10, 174, 175 |
| <code>\Glsxtrlong</code> | 10, 175 |
| <code>\glxtrlong</code> | 10, 174, 175 |
| <code>\GLSxtrlongpl</code> | 10, 175, 176 |
| <code>\Glsxtrlongpl</code> | 10, 176 |
| <code>\glxtrlongpl</code> | 10, 175 |
| <code>\glxtrlongshortdescsort</code> | 138 |
| <code>\glxtrmarkhook</code> | 167 |
| <code>\glxtrnewabbrevpresetkeyhook</code> | 119 |
| <code>\glxtrnewnumber</code> | 11 |
| <code>\glxtrnewsymbol</code> | 11 |
| <code>\glxtrNoGlossaryWarning</code> | 12, 78 |
| <code>\GlsXtrNoGlsWarningAutoMake</code> | 81 |
| <code>\GlsXtrNoGlsWarningBuildInfo</code> | 82 |
| <code>\GlsXtrNoGlsWarningCheckFile</code> | 81 |
| <code>\GlsXtrNoGlsWarningEmptyMain</code> | 81 |
| <code>\GlsXtrNoGlsWarningEmptyNotMain</code> | 81 |
| <code>\GlsXtrNoGlsWarningEmptyStart</code> | 81 |
| <code>\GlsXtrNoGlsWarningHead</code> | 81 |
| <code>\GlsXtrNoGlsWarningMisMatch</code> | 82 |
| <code>\GlsXtrNoGlsWarningNoOut</code> | 82 |
| <code>\GlsXtrNoGlsWarningTail</code> | 82 |
| <code>\glxstrorg@ifKV@glslink@hyper</code> | 28 |
| <code>\GLSxtrp</code> | 52 |
| <code>\Glsxtrp</code> | 52 |
| <code>\glxstrp</code> | 51, 54 |
| <code>\Glsxtrpl</code> | 22 |
| <code>\glxstrpl</code> | 22 |
| <code>\glxstrpostdescription</code> | 98, 112, 191 |
| <code>\glxstrpostlink</code> | 113 |
| <code>\glxstrpostlinkendsentence</code> | 113 |
| <code>\glxstrpostlinkhook</code> | 113 |
| <code>\glxstrpostlocalreset</code> | 57, 58, 67 |
| <code>\glxstrpostlocalunset</code> | 56, 58, 66 |
| <code>\glxstrpostnamehook</code> | 103–105 |
| <code>\GlsXtrPostNewAbbreviation</code> | 119, 135, 137–141, 143, 145–147, 149, 158, 159, 161, 164, 165 |
| <code>\glxstrpostreset</code> | 56, 58, 66 |
| <code>\glxstrpostunset</code> | 56, 58, 66 |
| <code>\glxstrprotectlinks</code> | 47 |
| <code>\glxstrregularfont</code> | 27, 30 |
| <code>\glxstrrestoremarkhook</code> | 167 |
| <code>\glxstrscfont</code> | 149–153 |
| <code>\glxstrscsuffix</code> | 150–153 |
| <code>\GlsXtrSetActualChar</code> | 109 |
| <code>\GlsXtrSetEncapChar</code> | 109 |
| <code>\GlsXtrSetEscChar</code> | 109 |
| <code>\GlsXtrSetLevelChar</code> | 109 |

| | | | |
|---|---|--|---|
| <code>\ifKV@glslink@hyper</code> | 28 | <code>\makeglossary</code> | 73 |
| <code>\ifKV@glslink@noindex</code> | 6, 7, 45 | <code>makeindex</code> | 203 |
| <code>\ifnum</code> | 8, 59, 60, 67, 68, 201 | <code>makeindex</code> | 72 |
| <code>\ifthenelse</code> | 81 | <code>\makenoidxglossaries</code> | 80 |
| <code>\IfTrackedLanguageFileExists</code> | 184 | <code>\MakeTextUppercase</code> | 168 |
| <code>\ifundef</code> | 47, 72, 111 | <code>\MakeUppercase</code> | 168, 169 |
| <code>\ifx</code> | 23, 25, 107, 109, 116, 118, 202 | <code>\marg</code> | 200 |
| <code>\immediate</code> | 59, 68, 78 | <code>\markboth</code> | 167 |
| <code>\index</code> | 106 | <code>\markright</code> | 167 |
| <code>\indexspace</code> | 202 | <code>\maxdimen</code> | 24 |
| <code>\input</code> | 184 | <code>\mbox</code> | 201 |
| <code>\InputIfFileExists</code> | 82 | <code>\medskip</code> | 81 |
| <code>\istfilename</code> | 72 | <code>\MessageBreak</code> | 19, 23, 59, 68, 75, 134 |
| <code>\item</code> | 80, 186 | <code>mfirstuc package</code> | 111 |
| J | | | |
| <code>\jobname</code> | 78–82 | <code>\mfirstucMakeUppercase</code> | 31–41, 43, 49–52, 55, 56, 62, 70, 84–94, 102, 103, 123, 124, 126, 128, 129, 131–133 |
| K | | | |
| <code>\key@ifundefined</code> | 7, 42 | <code>\mfu@checkword@arg</code> | 111, 112 |
| <code>\KV@glslink@hyperfalse</code> | 31, 44, 47 | <code>\mfu@checkword@do</code> | 112 |
| <code>\KV@glslink@noindexfalse</code> | 44, 45 | N | |
| <code>\KV@glslink@noindextrue</code> | 47 | <code>\NeedsTeXFormat</code> | 4, 185 |
| L | | | |
| <code>\LaTeX</code> | 79, 80 | <code>\new@glossaryentry</code> | 19, 75 |
| <code>\leaders</code> | 186 | <code>\new@ifnextchar</code> | 42, 43, 62, 115, 122–131 |
| <code>\let</code> | 4–11, 14, 18, 19, 23, 25, 26, 28–41, 43, 44, 46–48, 51, 57–59, 66, 67, 69–73, 75–77, 100–108, 111, 112, 116–118, 122–131, 143, 167–169, 191, 193 | <code>\newabbr</code> | 10 |
| <code>\letabbreviationstyle</code> | 142, 144, 146, 147, 149, 151, 155, 160 | <code>\newabbreviation</code> | 10, 71 |
| <code>\letcs</code> | 17, 18, 42, 76, 77, 100–105 | <code>\newabbreviationhook</code> | 119 |
| <code>\levelchar</code> | 109 | <code>\newabbreviationstyle</code> | 137–142, 144, 146, 147, 149–162, 164–166 |
| <code>\listadd</code> | 63 | <code>\newacronym</code> | 69, 71 |
| <code>\listbreak</code> | 111 | <code>\newacronymhook</code> | 70 |
| <code>\listcseadd</code> | 65 | <code>\newacronymstyle</code> | 71 |
| <code>\listcsgadd</code> | 19 | <code>\newcommand</code> | 4–18, 20–23, 25–27, 30, 31, 42, 43, 45–47, 51, 53–58, 60, 62–66, 68, 69, 71, 72, 76, 77, 79–99, 105–131, 133–136, 138, 140, 141, 149, 153, 157, 163, 164, 167–185, 191–193, 200 |
| <code>\listcsxadd</code> | 64 | <code>\newcount</code> | 8 |
| <code>\loadglsentries</code> | 19, 79 | <code>\newentry</code> | 11 |
| M | | | |
| <code>\MakeAcronymsAbbreviations</code> | 71 | <code>\newglossary</code> | 9, 73 |
| <code>\makeatletter</code> | 78, 108 | <code>\newglossaryentry</code> 11, 19, 58, 65, 70, 98, 99, 119 | |
| <code>\makeatother</code> | 108 | <code>\newglossaryentry options</code> | |
| <code>\makebox</code> | 186, 201 | <code>desc</code> | 87, 88, 92 |
| <code>\makefirstuc</code> | 112 | <code>descplural</code> | 88, 92, 93 |
| <code>makeglossaries</code> | 76 | <code>first</code> | 47, 85, 91, 136, 173, 174, 180, 203 |
| <code>\makeglossaries</code> | 72, 78, 80–82 | <code>firstplural</code> . | 86, 91, 92, 136, 173, 174, 181, 203 |
| | | <code>long</code> | 90, 93, 181 |
| | | <code>longplural</code> | 90, 94, 182 |
| | | <code>name</code> | 83, 84, 91, 106 |

| | | | |
|---------------------------|--|------------------------|--|
| plural | 84, 85, 91, 136, 172, 173, 179 | package options: | |
| see | 9, 17, 19, 73 | abbreviations | 9, 10 |
| short | 89, 93, 117 | accsupp | 12, 83 |
| shortplural | 89, 93, 117 | acronym | 10 |
| symbol | 86, 87, 92 | automake | 75, 79 |
| symbolplural | 87, 92 | docdef | 8, 18, 19, 58, 65 |
| text | 47, 84, 91, 136, 138, 171, 172, 179 | false | 19 |
| \newif | 105, 137 | restricted | 9 |
| \newlength | 192 | true | 19 |
| \newnum | 11 | nonumberlist | 24 |
| \newrobustcmd | 42, 43, 51, 52, 62, 111, 112, 122–131, 167, 168, 170–178, 193–199 | numbers | 10 |
| \newsym | 11 | record | 6, 7, 28, 82, 212 |
| \newterm | 98 | shortcuts | 11 |
| \newtoks | 117 | all | 11 |
| \newwrite | 72 | false | 11 |
| \NoCaseChange | 53–55, 169–177 | none | 11 |
| \noexpand | 13, 70, 78, 107, 119, 185 | true | 11 |
| \nofiles | 81 | style | 13 |
| \noindent | 81 | stylemods | 13 |
| \nopostdesc | 21, 22, 98 | symbols | 10, 98 |
| \nr | 5, 7, 8, 11, 12 | undefaction | 15, 16 |
| \ns@GLSxtrfull | 123 | error | 5 |
| \ns@Glsxtrfull | 122 | warn | 5 |
| \ns@glxtrfull | 122 | \PackageError | |
| \ns@GLSxtrfullpl | 124 | | 5, 13, 19, 22, 23, 42, 43, 51, 52, 57–59, 65, 67, 69, 71, 73–75, 133–136, 185 |
| \ns@Glsxtrfullpl | 124 | \PackageWarning | 9 |
| \ns@glxtrfullpl | 123 | \PackageWarningNoLine | 9 |
| \ns@GLSxtrlong | 127 | \par | 81, 82, 191, 192, 201 |
| \ns@Glsxtrlong | 127 | \parindent | 192, 201 |
| \ns@glxtrlong | 126 | \PassOptionsToPackage | 4 |
| \ns@GLSxtrlongpl | 131 | \preto | 45 |
| \ns@Glsxtrlongpl | 130 | \printabbreviations | 9 |
| \ns@glxtrlongpl | 130 | \printglossaries | 73, 80 |
| \ns@GLSxtrshort | 126 | \printglossary | 9, 73, 80 |
| \ns@Glsxtrshort | 125 | \printglossary options | |
| \ns@glxtrshort | 125 | nonumberlist | 27 |
| \ns@GLSxtrshortpl | 129 | type | 75 |
| \ns@Glsxtrshortpl | 128 | \printnoidxglossaries | 80 |
| \ns@glxtrshortpl | 128 | \printnoidxglossary | 74, 80 |
| \null | 12 | \printnumbers | 11, 98, 99 |
| \number | 64–67 | \printsymbols | 11, 98 |
| \numexpr | 64, 65, 67 | \printunsrtglossary | 82 |
| O | | \ProcessOptions | 185 |
| \or | 5, 7, 8, 11, 12, 19 | \ProcessOptionsX | 13 |
| \org@glossaryentrynumbers | 24, 25 | \protect | 53–56, 91– 94, 120, 137–159, 161, 164, 165, 169–177 |
| P | | \protected@csedef | 192 |
| \p@gl@s@hyp@opt | 46 | \protected@csxdef | 192 |

| | | | |
|--|--|---|--|
| <code>\protected@edef</code> | 23, 70, 106, 119 | <code>\spacefactor</code> | 114, 118, 191 |
| <code>\protected@write</code> | 6, 26, 72, 73, 82 | <code>\string</code> .. | 5, 6, 19, 20, 23, 26, 42, 43, 51, 52, 57–59, 65, 67–69, 71–75, 78–82, 102–106 |
| <code>\providecommand</code> . | 9, 26, 43, 59, 68, 72, 78, 185 | <code>\strut</code> | 186–191 |
| <code>\ProvidesFile</code> | 184 | <code>\subglossentry</code> | 186–191, 201 |
| <code>\ProvidesPackage</code> | 4, 185 | | |
| Q | | T | |
| <code>\quotechar</code> | 109 | <code>\tablehead</code> | 189, 190 |
| R | | <code>\tabletail</code> | 189, 190 |
| <code>\raggedright</code> | 188, 190 | <code>\tabularnewline</code> | 186–191 |
| <code>\relax</code> | 5, 7, 8, 10–12, 14, 19, 23, 24, 26, 29, 46, 51, 59, 60, 67, 68, 73, 77, 107–109, 112, 114, 118, 193–202 | <code>\TeX</code> | 79 |
| <code>relese package</code> | 153 | <code>\texorpdfstring</code> | 53–55, 178–183 |
| <code>\renewcommand</code> ... | 7–16, 19, 20, 22, 23, 25– 29, 42, 44, 45, 47, 51, 56–59, 63, 65–67, 69–75, 78, 98, 100, 101, 103, 104, 110– 113, 121, 135, 137–167, 186–191, 201, 202 | <code>textcase package</code> | 167 |
| <code>\renewenvironment</code> | 186–190, 200 | <code>\textsc</code> | 149 |
| <code>\renewglossarystyle</code> | 186–190, 200 | <code>\textsmaller</code> | 153 |
| <code>\renewrobustcmd</code> | 29 | <code>\texttt</code> | 79–81 |
| <code>\RequireGlossariesExtraLang</code> | 184 | <code>\the</code> | 70, 109, 110, 119, 137– 141, 143–147, 149, 157–159, 161, 164, 165 |
| <code>\RequirePackage</code> | 4, 12–14, 185 | <code>\theindex</code> | 106 |
| <code>\reserved@a</code> | 115, 116 | <code>\this@dialect</code> | 184 |
| <code>\reserved@b</code> | 115, 116 | <code>\toks@</code> | 109, 110 |
| <code>\reserved@d</code> | 116 | U | |
| <code>\RestoreAcronyms</code> | 71 | <code>\undef</code> | 111 |
| <code>\romannumeral</code> | 192, 193, 200 | <code>\underline</code> | 112 |
| S | | <code>\unskip</code> | 18, 186 |
| <code>\s@glshyp@opt</code> | 46 | <code>\usepackage</code> | 80, 81 |
| <code>\s@glstr@enabletagging</code> | 110, 111 | V | |
| <code>\seename</code> | 17 | <code>\val</code> | 5, 7, 8, 11, 12 |
| <code>\setabbreviationstyle</code> | 71, 138, 146 | W | |
| <code>\setacronymstyle</code> | 70, 71 | <code>\warn@nomakeglossaries</code> | 73 |
| <code>\SetGenericNewAcronym</code> | 71 | <code>\warn@noprintglossary</code> | 73 |
| <code>\setglossarystyle</code> | 13, 186, 202 | <code>\write</code> | 59, 68, 72, 78 |
| <code>\setkeys</code> | 6, 13, 14, 30, 45, 69, 118, 119 | X | |
| <code>\setlength</code> | 24, 192, 201 | <code>\xcapitalisewords</code> | 99 |
| <code>\settowidth</code> | 72, 192–200 | <code>\xifinlist</code> | 64 |
| <code>\setupglossaries</code> | 4, 14 | <code>xindy</code> | 203 |
| <code>\sfcode</code> | 114, 191 | <code>xindy</code> | 72 |
| <code>\space</code> | 5, 20, 23, 57–59, 65, 67–69, 71–74, 78, 81, 114, 120, 138, 191, 192, 200 | <code>xkeyval package</code> | 4 |
| | | <code>\XKV@checkchoice</code> | 27 |
| | | <code>\XKV@plfalse</code> | 27 |
| | | <code>\XKV@resa</code> | 27 |
| | | <code>\XKV@sttrue</code> | 27 |