

glossaries-extra.sty v1.39: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2019-03-22

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)	5
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	30
1.3	Modifications to Commands Provided by glossaries	44
1.3.1	Existence Checks	49
1.3.2	Document Definitions	58
1.3.3	Existing Glossary Style Modifications	64
1.3.4	Entry Formatting, Hyperlinks and Indexing	68
1.3.5	Entry Counting	106
1.3.6	Acronym Modifications	121
1.3.7	Indexing and Displaying Glossaries	124
1.4	Link Counting	162
1.5	Integration with glossaries-accsupp	164
1.6	Categories	177
1.7	Abbreviations	204
1.7.1	Abbreviation Styles Setup	224
1.7.2	Predefined Styles (Default Font)	227
1.7.3	Predefined Styles (Small Capitals)	244
1.7.4	Predefined Styles (Fake Small Capitals)	258
1.7.5	Predefined Styles (Emphasized)	272
1.7.6	Predefined Styles (User Parentheses Hook)	294
1.7.7	Predefined Styles (Hyphen)	303
1.7.8	Predefined Styles (No Short on First Use)	317
1.8	Using Entries in Headings	320
1.9	Multi-Lingual Support	339
1.10	glossaries-extra-bib2gls.sty	340
2	Style Adjustments (glossaries-extra-stylemods.sty)	385
2.1	Package Initialisation	385
2.2	List-Like Styles	386
2.3	Longtable Styles	389
2.4	Long Ragged Styles	391
2.5	Supertabular Styles	393
2.6	Super Ragged Styles	395
2.7	Inline Style	397
2.8	Tree Styles	397
2.9	Multicolumn Styles	415

3 bookindex style (glossary-bookindex.sty)	421
3.1 Package Initialisation and Options	421
4 longextra styles (glossary-longextra.sty)	428
4.1 Package Initialisation and Options	428
Glossary	451
Change History	452
Index	473

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2019/03/22 1.39 (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{\glxstr@doption}[1]{\setupglossaries{#1}}%
```

```
8 \let\@glxstr@declareoption\@gls@declareoption
```

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

```
11 \newcommand{\glxstr@doption}[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*{\@glxstr@declareoption}[2]{%
```

```
25 \DeclareOptionX{#1}{#2}%
```

```
26 \DeclareOption{#1}{#2}%
```

```
27 }
```

```
28 }
```

Declare package options.

`\glstrundefaction` 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*{\glstrundefaction}[2]{%
30   \@glstrundeftag\PackageError{glossaries-extra}{#1}{#2}%
31 }

```

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

```

32 \newcommand*{\glstr@warnonexistsordo}[1]{%

```

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

```

33 \newcommand*{\glstrundeftag}{??}
34 \newcommand*{\@glstrundeftag}{}

```

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 `\glstrundefaction` should behave if `undefaction=warn` is set.

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

47 \newcommand*{\@glstr@redef@for@gl@sentries}{}

```

`\f@for@gl@sentries`

```

48 \newcommand*{\@glstr@do@redef@for@gl@sentries}{%
49   \renewcommand*{\for@gl@sentries}[3][\gl@defaulttype]{%
50     \edef\@glo@list{\csname glolist@##1\endcsname}%
51     \ifdefstring{\@glo@list}{,}%
52     {%
53       \GlossariesExtraWarning{No entries defined in glossary '#1'}%
54     }%
55     {%
56       \@for##2:=\@glo@list\do

```

```

57      {%
58      \ifdefempty{##2}{-}{##3}%
59      }%
60      }%
61      }%
62 }%

```

undefaction

```

63 \define@choicekey{glossaries-extra.sty}{undefaction}%
64 [\glstr@undefaction@val\glstr@undefaction@nr]%
65 {warn,error}%
66 {%
67   \ifcase\glstr@undefaction@nr\relax
68     \let\glstrundefaction\glstr@warn@undefaction
69     \let\glstr@warnonexistssordo\glstr@warn@onexistssordo
70     \let\glstr@redef@for\glstr@do@redef@for
71   \or
72     \let\glstrundefaction\glstr@err@undefaction
73     \let\glstr@warnonexistssordo\glstr@gobble
74     \let\glstr@redef@for\glstr@relax
75   \fi
76 }

```

To assist bib2gls, v1.08 introduces the record option, which will write information to the aux file whenever an entry needs to be indexed.

\glstr@record Does nothing by default.

```
77 \newcommand*{\glstr@record}[3]{}

```

\glstr@recordsee Does nothing by default.

```
78 \newcommand*{\glstr@recordsee}[2]{}

```

\glstr@defaultnumberformat

```
79 \newcommand*{\glstr@defaultnumberformat}{\glstrnumberformat}%

```

\GlsXtrSetDefaultNumberFormat

```

80 \newcommand*{\GlsXtrSetDefaultNumberFormat}[1]{%
81   \renewcommand*{\glstr@defaultnumberformat}{#1}%
82 }%

```

The record option is somewhat problematic. On the first \LaTeX run the entries aren't defined. This isn't as straight-forward as commands like `\cite` since attributes associated with the entry's category may switch off the indexing or the entry's glossary type might require a particular counter. This kind of information can't be determined until the entry has been defined. So there are two different commands here. One that's used if the entry hasn't been defined, which tries to use sensible defaults, and one which is used when the entry has been defined.

`cord@wrglossary` The `record=only` option sets `\@@do@wrglossary` to this command, which means it's done within `\glsadd` and `\@gls@link`, and so is only done if the entry exists.

```

83 \newcommand*{\@glsxtr@do@record@wrglossary}[1]{%
84   \begingroup
85     \ifKV@gls@link@noindex
86     \else
87       \edef\@gls@label{\glsdetoklabel{#1}}%
88       \let\gls@label\@gls@label
89       \glswriteentry{#1}%
90       {%
91         \ifdefempty{\@glsxtr@thevalue}%
92         {%
93           \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
94           \else
95             \let\theHglentrycounter\@glsxtr@theHvalue
96             \fi
97             \glsxtr@saveentrycounter
98             \let\@@do@@wrglossary\@glsxtr@dorecord
99           }%
100          {%
101            \let\theHglentrycounter\@glsxtr@thevalue
102            \let\theHglentrycounter\@glsxtr@theHvalue
103            \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
104          }%
105          \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
106            \glsxtr@@do@wrglossary{#1}%
107          \else
108            \@glsxtrwrglossmark
109
110            Increment associated counter.
111            \glsxtr@inc@wrglossaryctr{#1}%
112            \@@do@@wrglossary
113          \fi
114        }%
115      \fi
116    \endgroup
117  }

```

`index@wrglossary` The `record=alsoindex` option needs to both record and index.

```

116 \newcommand*{\@glsxtr@do@alsoindex@wrglossary}[1]{%
117   \glsxtr@@do@wrglossary{#1}%
118   \@glsxtr@do@record@wrglossary{#1}%
119 }

```

`@@glsxtr@record` The `record=only` option sets `\@glsxtr@record` to this. This performs the recording if the entry *doesn't exist* and is done at the start of `\@gls@field@link` and commands like `\@gls@` (before the existence test). This means that it disregards the `wrgloss` key.

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. The second argument is the entry's

label. The third argument is the key family (glslink in most cases, glossadd for \glsadd).

```
120 \newcommand*{\@@glsxtr@record}[3]{%
121   \ifglsentryexists{#2}{}%
122   {%
123     \@@glsxtrwrglossmark
124     \begingroup
```

Save the label in case it's needed.

```
125     \edef\@gls@label{\glsdetoklabel{#2}}%
126     \let\glslabel\@gls@label
127     \let\@glsnumberformat\@glsxtr@defaultnumberformat
128     \def\@glsxtr@thevalue{%
129     \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
130     \let\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

Entry hasn't been defined, so we'll have to assume it's \glscounter by default.

```
131     \let\@gls@counter\glscounter
```

Unless the equations option is on and this is inside a numbered maths environment.

```
132     \ifglsxtr@equations
133     \@glsxtr@use@equation@counter
134     \fi
```

Check for default options (which may switch off indexing).

```
135     \@gls@setdefault@glslink@opts
```

Implement any pre-key settings.

```
136     \csuse{\@glsxtr@#3@prekeys}%
```

Assign keys.

```
137     \setkeys{#3}{#1}%
```

Implement any post-key settings. Is the auto-add on?

```
138     \glsxtr@do@autoadd{#3}%
```

Check post-key hook.

```
139     \csuse{\@glsxtr@#3@postkeys}%
```

Increment associated counter.

```
140     \glsxtr@inc@wrglossaryctr{#2}%
```

Check if noindex option has been used.

```
141     \ifKV@glslink@noindex
142     \else
143     \glswriteentry{#2}%
144     {%
```

Check if thevalue has been set.

```
145     \ifdefempty{\@glsxtr@thevalue}%
146     {%
```

Key thevalue hasn't been set, but check if theHvalue has been set. (Not particularly likely, but allow for it.)

```
147     \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

```

148         \else
149         \let\theHglentrycounter\@glxtr@theHvalue
150         \fi

```

Save the entry counter.

```

151         \glxtr@saveentrycounter
Temporarily redefine \@do@wrglossary for use with \glxtr@do@wrglossary.
152         \let\@do@wrglossary\@glxtr@dorecord
153         }%
154         {%

```

thevalue has been set, so there's no need to defer writing the location value. (If it's dependent on the page counter, the counter key should be set instead.)

```

155         \let\theglentrycounter\@glxtr@thevalue
156         \let\theHglentrycounter\@glxtr@theHvalue
157         \let\@do@wrglossary\@glxtr@dorecordnodefer
158         }%
159         \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
160         \glxtr@{#2}%
161         \else

```

No need to escape special characters.

```

162         \@do@wrglossary
163         \fi
164         }%
165     \fi
166 \endgroup
167 }%
168 }

```

glslink@prekeys

```

169 \newcommand{\@glxtr@glslink@prekeys}{\glslinkpresetkeys}

```

glslink@postkeys

```

170 \newcommand{\@glxtr@glslink@postkeys}{\glslinkpostsetkeys}

```

glossadd@prekeys

```

171 \newcommand{\@glxtr@glossadd@prekeys}{\glsaddpresetkeys}

```

glossadd@postkeys

```

172 \newcommand{\@glxtr@glossadd@postkeys}{\glsaddpostsetkeys}

```

glsxtr@dorecord If record=alsoindex is used, then \@glslocref may have been escaped, but this isn't appropriate here.

```

173 \newcommand*\@glxtr@dorecord{%
174     \global\let\@glsrecordlocref\theglentrycounter
175     \let\@glxtr@orgprefix\@glo@counterprefix
176     \ifx\theglentrycounter\theHglentrycounter
177         \def\@glo@counterprefix{}%
178     \else

```

Protect against non-expandable commands occurring in the location.

```

179 \protected@edef\@glxtr@theentrycounter{\theglentrycounter}%
180 \protected@edef\@glxtr@theHentrycounter{\theHglentrycounter}%
181 \@onelevel@sanitize\@glxtr@theentrycounter
182 \@onelevel@sanitize\@glxtr@theHentrycounter
183 \protected@edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
184   {\@glxtr@theentrycounter}{\@glxtr@theHentrycounter}}%
185 }%
186 \@do@gl@getcounterprefix
187 \fi

```

Don't protect the \@gl@recordlocref from premature expansion. If the counter isn't

page then it needs expanding. If the location includes \thepage then \protected@write will automatically deal with it.

```

188 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
189 \@glxtr@do@nameref@record
190   {\@gl@label}{\@gl@counterprefix}{\@gl@counter}{\@gl@numberformat}%
191   {\@gl@recordlocref}}%
192 \else
193   \protected@write\@auxout{}{\string\@glxtr@record
194     {\@gl@label}{\@gl@counterprefix}{\@gl@counter}{\@gl@numberformat}%
195     {\@gl@recordlocref}}}%
196 \fi
197 \@glxtr@counterrecordhook
198 \let\@gl@counterprefix\@glxtr@orgprefix
199 }

```

dorecordnodefer As above, but don't defer expansion of location. This uses \theglentrycounter directly for the location rather than \@gl@locref since there's no need to guard against premature expansion of the page counter.

```

200 \newcommand*\@glxtr@dorecordnodefer{%
201   \ifx\theglentrycounter\theHglentrycounter
202     \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
203       \@glxtr@do@nameref@record
204         {\@gl@label}{\@gl@counter}{\@gl@numberformat}%
205         {\theglentrycounter}}%
206   \else
207     \protected@write\@auxout{}{\string\@glxtr@record
208       {\@gl@label}{\@gl@counter}{\@gl@numberformat}%
209       {\theglentrycounter}}}%
210   \fi
211 \else
212   \edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
213     {\theglentrycounter}{\theHglentrycounter}}%
214   }%
215   \@do@gl@getcounterprefix
216   \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
217     \@glxtr@do@nameref@record

```

```

218         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}%
219         {\@glsnumberformat}{\theglsentrycounter}%
220     \else
221         \protected@write\@auxout{}\string\glsxtr@record
222         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
223         {\theglsentrycounter}}%
224     \fi
225 \fi
226 \@glsxtr@counterrecordhook
227 }

```

`\glsxtr@ifnum@mmode` Check if in a numbered maths environment. The `amsmath` package is automatically loaded by `datatool-base`, which is required by glossaries, so `\ifst@rred` and `\if@display` should both be defined.

```

228 \newcommand{\@glsxtr@ifnum@mmode}[2]{%
229   \ifmmode
230     \ifst@rred
231       #2%
232   \else

```

Non-`amsmath` environments and regular inline math mode isn't flagged as starred by `amsmath`, but we can't use `\mathchoice` in this case as it's not the current style that's relevant. Instead we can use `amsmath`'s `\if@display`. This may not work for environments that aren't provided by `amsmath`.

```

233     \if@display #1\else #2\fi
234   \fi
235 \else
236   #2%
237 \fi
238 }

```

`\glsnameref@record` With `record=nameref`, the current label information is included in the record, but this may not have been defined, so `\csuse` will prevent an undefined control sequence error and just leave the last two arguments blank if there's no information. In the event that a record is in `amsmath`'s `align` environment `\@currentHref` will be out. There may be other instances where `\@currentHref` is out, so this also saves `\theHglentrycounter`, which is useful if it can't be obtained by prefixing `\theglsentrycounter`.

```

239 \newcommand*{\@glsxtr@do@nameref@record}[5]{%
240   \gls@ifnotmeasuring
241   {%
242     \protected@write\@auxout{}\string\glsxtr@record@nameref
243     {#1}{#2}{#3}{#4}{#5}%
244     {\csuse{@currentlabelname}}{\csuse{@currentHref}}%
245     {\theHglentrycounter}}%
246   }%
247 }

```

`\glsxtr@recordcounter`

```

248 \newcommand*{\@glsxtr@recordcounter}{%
249   \glsxtr@noop@recordcounter
250 }

p@recordcounter
251 \newcommand*{\@glsxtr@noop@recordcounter}[1]{%
252   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
253     requires record=only or record=alsoindex package option}{}%
254 }

p@recordcounter
255 \newcommand*{\@glsxtr@op@recordcounter}[1]{%
256   \eappto\@glsxtr@counterrecordhook{\noexpand\@glsxtr@docounterrecord{#1}}%
257 }

lsxtr@recordsee Deal with \glssee in record mode. (This doesn't increment the associated counter.)
258 \newcommand*{\@glsxtr@recordsee}[2]{%
259   \@@glsxtrwrglossmark
260   \def\@gls@xref{#2}%
261   \@onelevel@sanitize\@gls@xref
262   \protected@write\@auxout{}\string\glsxtr@recordsee{#1}{\@gls@xref}}%
263 }

srtglossaryunit
264 \newcommand{\printunsrtglossaryunit}{%
265   \print@noop@unsrtglossaryunit
266 }

tr@setup@record Initialise.
267 \newcommand*{\glsxtr@setup@record}{\let\@do@wrglossary\glsxtr@do@wrglossary}

aveentrycounter Only store the entry counter information if the indexing is on.
268 \newcommand*{\glsxtr@indexonly@saveentrycounter}{%
269   \ifKV@glslink@noindex
270   \else
271     \glsxtr@saveentrycounter
272   \fi
273 }

addloclistfield
274 \newcommand*{\glsxtr@addloclistfield}{%
275   \key@ifundefined{glossentry}{loclist}%
276   {%
277     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
278     \appto\@gls@keymap{,{loclist}{loclist}}%
279     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
280     \appto\@newglossaryentryposthook{%
281       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%

```

```

282 }%
283 \glssetnoexpandfield{loclist}%
284 }%
285 {}%

```

The loclist field is just a comma-separated list. The location field is the formatted list.

```

286 \key@ifundefined{glossentry}{location}%
287 {%
288 \define@key{glossentry}{location}{\def\@glo@location{##1}}%
289 \appto\@gls@keymap{,{location}{location}}%
290 \appto\@newglossaryentryprehook{\def\@glo@location{}}%
291 \appto\@newglossaryentryposthook{%
292 \gls@assign@field{\@glo@label}{location}{\@glo@location}%
293 }%
294 \glssetnoexpandfield{location}%
295 }%
296 {}%

```

Add a key to store the group heading.

```

297 \key@ifundefined{glossentry}{group}%
298 {%
299 \define@key{glossentry}{group}{\def\@glo@group{##1}}%
300 \appto\@gls@keymap{,{group}{group}}%
301 \appto\@newglossaryentryprehook{\def\@glo@group{}}%
302 \appto\@newglossaryentryposthook{%
303 \gls@assign@field{\@glo@label}{group}{\@glo@group}%
304 }%
305 \glssetnoexpandfield{group}%
306 }%
307 {}%
308 }

```

@record@setting Keep track of the record package option.

```

309 \newcommand*{\@glsxtr@record@setting}{off}

```

etting@alsoindex

```

310 \newcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}

```

rd@setting@only

```

311 \newcommand*{\@glsxtr@record@setting@only}{only}

```

setting@nameref

```

312 \newcommand*{\@glsxtr@record@setting@nameref}{nameref}

```

@if@record@only

```

313 \newcommand*{\@glsxtr@if@record@only}[2]{%
314 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
315 #1%
316 \else

```

```

317 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
318   #1%
319 \else
320   #2%
321 \fi
322 \fi
323 }

```

ord@setting@off

```

324 \newcommand*{\@glsxtr@record@setting@off}{off}

```

cord@only@setup Initialisation code for record=only and record=nameref

```

325 \newcommand*{\@glsxtr@record@only@setup}{%
326 \def\glsxtr@setup@record{%
327   \@glsxtr@autoseeindexfalse
328   \let\@do@seeglossary\@glsxtr@recordsee
329   \let\@glsxtr@record\@glsxtr@record
330   \let\@do@wrglossary\@glsxtr@do@record@wrglossary
331   \let\@gls@saveentrycounter\relax
332   \let\glsxtrundefaction\@glsxtr@warn@undefaction
333   \let\glsxtr@warnonexistssordo\@glsxtr@warn@onexistssordo
334   \glsxtr@addloclistfield
335   \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
336   \let\@glsxtr@recordcounter\@glsxtr@op@recordcounter
337   \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

```

Switch off the index suppression for aliased entries. (bib2gls will deal with them.)

```

338 \def\glsxtrsetaliasnoindex{}%

```

\@gls@setupsort@none was only introduced to glossaries v4.30, so it may not be available. If it's defined, use it to remove the unnecessary overhead of escaping and sanitizing the sort value.

```

339 \ifdef\@gls@setupsort@none{\@gls@setupsort@none}{}%

```

Warn about using \printglossary:

```

340 \def\glsxtrNoGlossaryWarning{\@glsxtr@record@noglossarywarning}%

```

Load glossaries-extra-bib2gls:

```

341 \RequirePackage{glossaries-extra-bib2gls}%
342 }%
343 }

```

record Now define the record package option.

```

344 \define@choicekey{glossaries-extra.sty}{record}
345 [ \@glsxtr@record@setting\glsxtr@record@nr ]%
346 {off,only,alsoindex,nameref}%
347 [only]%
348 {%
349 \ifcase\glsxtr@record@nr\relax

```

Don't record.

```

350 \def\glxtr@setup@record{%
351 \renewcommand*{\@do@seeglossary}{\@glxtr@doseeglossary}%
352 \renewcommand*{\@glxtr@record}[3]{}%
353 \let\@do@wrglossary\glxtr@do@wrglossary
354 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
355 \let\glxtrundefaction\glxtr@err@undefaction
356 \let\glxtr@warnonexistsordo\@gobble
357 \let\@glxtr@recordcounter\@glxtr@noop@recordcounter
358 \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
359 \undef\glxtrsetaliasnoindex
360 }%
361 \or

```

Only record (don't index).

```

362 \@glxtr@record@only@setup
363 \or

```

Record and index. This option doesn't load glossaries-extra-bib2gls as the sorting is performed by xindy or makeindex.

```

364 \def\glxtr@setup@record{%
365 \renewcommand*{\@do@seeglossary}{\@glxtr@dosee@alsoindex@glossary}%
366 \let\@glxtr@record\@glxtr@record
367 \let\@do@wrglossary\glxtr@do@alsoindex@wrglossary
368 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
369 \let\glxtrundefaction\glxtr@warn@undefaction
370 \let\glxtr@warnonexistsordo\glxtr@warn@onexistsordo
371 \glxtr@addloclistfield
372 \let\@glxtr@recordcounter\@glxtr@op@recordcounter
373 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
374 \undef\glxtrsetaliasnoindex
375 }%
376 \or

```

Only record (don't index) but also include nameref information.

```

377 \@glxtr@record@only@setup
378 \ifundef\hyperlink
379 {\GlossariesExtraWarning{You have requested record=nameref but
380 the document doesn't support hyperlinks}}%
381 {}%
382 \fi
383 }

```

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

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

```

384 \newcommand*{\@glxtr@docdefval}{0}

```

Need to provide conditional commands that are backward compatible:

```

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

lsxtrdocdeftrue
386 \newcommand*{\@glxtrdocdeftrue}{\def\@glxtr@docdefval{1}}

lsxtrdocdeffalse
387 \newcommand*{\@glxtrdocdeffalse}{\def\@glxtr@docdefval{0}}

docdef 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.
388 \define@choicekey{glossaries-extra.sty}{docdef}
389 [ \@glxtr@docdefsetting\@glxtr@docdefval ]%
390 {false,true,restricted,atom}[true]%
391 {%
392   \ifnum\@glxtr@docdefval>1\relax
393     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
394   \else
395     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}%
396   \fi
397 }

ocdefrestricted
398 \newcommand*{\if@glxtrdocdefrestricted}{\ifnum\@glxtr@docdefval>1 }

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).
399 \newcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}

indexcrossrefs Automatically index cross references at the end of the document
400 \define@boolkey{glossaries-extra.sty}[\@glxtr]{indexcrossrefs}[true]{%
401   \if@glxtrindexcrossrefs
402   \else
403     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
404   \fi
405 }

Switch off since this can increase the build time.
406 \@glxtrindexcrossrefsfalse

But allow see key to switch it on automatically.

oindexcrossrefs
407 \newcommand*{\@glxtr@autoindexcrossrefs}{\@glxtrindexcrossrefstrue}

```

autoseeindex Provide a boolean option to allow the user to prevent the automatic indexing of the cross-referencing keys `see`, `seealso` and `alias`.

```

408 \define@boolkey{glossaries-extra.sty}[@glxtr@]{autoseeindex}[true]{%
409 }
410 \@glxtr@autoseeindextrue

```

equations Provide a boolean option to automatically switch to the equation counter when in a numbered maths environment.

```

411 \define@boolkey{glossaries-extra.sty}[@glxtr@]{equations}[true]{%
412 }
413 \@glxtr@equationsfalse

```

\glxtr@float

```

414 \let\glxtr@float\@float

```

glxtr@dblfloat

```

415 \let\glxtr@dblfloat\@dblfloat

```

floats Provide a boolean option to automatically switch to the the corresponding counter when in a float.

```

416 \define@boolkey{glossaries-extra.sty}[@glxtr@]{floats}[true]{%
417   \ifglxtr@floats
418     \renewcommand*{\@float}[1]{\renewcommand{\glscounter}{##1}\glxtr@float{##1}}%
419     \renewcommand*{\@dblfloat}[1]{\renewcommand{\glscounter}{##1}\glxtr@dblfloat{##1}}%
420   \else
421     \let\@float\glxtr@float
422     \let\@dblfloat\glxtr@dblfloat
423   \fi
424 }
425 \@glxtr@floatsfalse

```

iesExtraWarning Allow users to suppress warnings.

```

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

```

raWarningNoLine Allow users to suppress warnings.

```

427 \newcommand*{\GlossariesExtraWarningNoLine}[1]{%
428   \PackageWarningNoLine{glossaries-extra}{#1}}

429 \@glxtr@declareoption{nowarn}{%
430   \let\GlossariesExtraWarning\@gobble
431   \let\GlossariesExtraWarningNoLine\@gobble
432   \glxtr@doption{nowarn}%
433 }

```

xtr@defpostpunc Redefines `\glspostdescription`. The `postdot` and `nopostdot` options will have to redefine this.

```

434 \newcommand*{\@glxtr@defpostpunc}{%

```

postdot Shortcut for nopostdot=false

```
435 \@glsxtr@declareoption{postdot}{%
436   \glsxtr@doption{nopostdot=false}%
437   \renewcommand*{\@glsxtr@defpostpunc}{%
438     \renewcommand*{\glspostdescription}{%
439       \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
440   }%
441 }
```

nopostdot Needs to redefine \@glsxtr@defpostpunc

```
442 \define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
443   \glsxtr@doption{nopostdot=#1}%
444   \renewcommand*{\@glsxtr@defpostpunc}{%
445     \renewcommand*{\glspostdescription}{%
446       \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
447   }%
448 }
```

postpunc Set the post-description punctuation. This also sets the \ifglsnopostdot conditional, which now indicates if the post-description punctuation has been suppressed.

```
449 \define@key{glossaries-extra.sty}{postpunc}{%
450   \glsxtr@doption{nopostdot=false}%
451   \ifstrequal{#1}{dot}%
452   {%
453     \renewcommand*{\@glsxtr@defpostpunc}{%
454       \renewcommand*{\glspostdescription}{.\spacefactor\sfcode'\. }%
455     }%
456   }%
457   {%
458     \ifstrequal{#1}{comma}%
459     {%
460       \renewcommand*{\@glsxtr@defpostpunc}{%
461         \renewcommand*{\glspostdescription}{,}%
462       }%
463     }%
464     {%
465       \ifstrequal{#1}{none}%
466       {%
467         \glsxtr@doption{nopostdot=true}%
468         \renewcommand*{\@glsxtr@defpostpunc}{%
469           \renewcommand*{\glspostdescription}{}%
470         }%
471       }%
472     }%
473     \renewcommand*{\@glsxtr@defpostpunc}{%
474       \renewcommand*{\glspostdescription}{#1}%
475     }%
476   }%
477 }
```

```

478 }%
479 }

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

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

abbreviationsdef
482 \newcommand*{\@glsxtr@doabbreviationsdef}{%
483   \ifpackageloaded{babel}%
484   {\providecommand{\abbreviationsname}{\acronymname}}%
485   {\providecommand{\abbreviationsname}{Abbreviations}}%
486   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
487   \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
488   \newcommand*{\printabbreviations}[1][{}]{%
489     \printglossary[type=glsxtrabbrvtype,##1]%
490   }%
491   \disable@keys{glossaries-extra.sty}{abbreviations}%

   If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.
492   \ifglsacronym
493   \else
494     \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
495   \fi
496 }%

abbreviations If abbreviations, create a new glossary type for abbreviations.
497 \@glsxtr@declareoption{abbreviations}{%
498   \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
499 }

AbbreviationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided
by glossaries, this uses \newcommand instead of \let as a safety feature (except for \newabbr
which is also provided with \GlsXtrDefineAcShortcuts).
500 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
501   \newcommand*{\ab}{\cgl}s}%
502   \newcommand*{\abp}{\cgl spl}%
503   \newcommand*{\as}{\glsxtrshort}%
504   \newcommand*{\asp}{\glsxtrshortpl}%
505   \newcommand*{\al}{\glsxtrlong}%
506   \newcommand*{\alp}{\glsxtrlongpl}%
507   \newcommand*{\af}{\glsxtrfull}%
508   \newcommand*{\afp}{\glsxtrfullpl}%
509   \newcommand*{\Ab}{\cGls}%
510   \newcommand*{\Abp}{\cGls pl}%
511   \newcommand*{\As}{\Glsxtrshort}%
512   \newcommand*{\Asp}{\Glsxtrshortpl}%

```

```

513 \newcommand*{\Al}{\Glsxtrlong}%
514 \newcommand*{\Alp}{\Glsxtrlongpl}%
515 \newcommand*{\Af}{\Glsxtrfull}%
516 \newcommand*{\Afp}{\Glsxtrfullpl}%
517 \newcommand*{\AB}{\cGLS}%
518 \newcommand*{\ABP}{\cGLSpl}%
519 \newcommand*{\AS}{\GLSxtrshort}%
520 \newcommand*{\ASP}{\GLSxtrshortpl}%
521 \newcommand*{\AL}{\GLSxtrlong}%
522 \newcommand*{\ALP}{\GLSxtrlongpl}%
523 \newcommand*{\AF}{\GLSxtrfull}%
524 \newcommand*{\AFP}{\GLSxtrfullpl}%

525 \providecommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

526 \let\GlsXtrDefineAbbreviationShortcuts\relax
527 }

```

fineAcShortcuts Enable shortcut commands for the abbreviations, but uses the analogous commands provided by glossaries.

```

528 \newcommand*{\GlsXtrDefineAcShortcuts}{%
529 \newcommand*{\ac}{\cglS}%
530 \newcommand*{\acp}{\cglSpl}%
531 \newcommand*{\acs}{\glSxtrshort}%
532 \newcommand*{\acsp}{\glSxtrshortpl}%
533 \newcommand*{\acl}{\glSxtrlong}%
534 \newcommand*{\aclp}{\glSxtrlongpl}%
535 \newcommand*{\acf}{\glSxtrfull}%
536 \newcommand*{\acfp}{\glSxtrfullpl}%
537 \newcommand*{\Ac}{\cGls}%
538 \newcommand*{\Acp}{\cGlspl}%
539 \newcommand*{\Acs}{\Glsxtrshort}%
540 \newcommand*{\Acsp}{\Glsxtrshortpl}%
541 \newcommand*{\Acl}{\Glsxtrlong}%
542 \newcommand*{\Aclp}{\Glsxtrlongpl}%
543 \newcommand*{\Acf}{\Glsxtrfull}%
544 \newcommand*{\Acfp}{\Glsxtrfullpl}%
545 \newcommand*{\AC}{\cGLS}%
546 \newcommand*{\ACP}{\cGLSpl}%
547 \newcommand*{\ACS}{\GLSxtrshort}%
548 \newcommand*{\ACSP}{\GLSxtrshortpl}%
549 \newcommand*{\ACL}{\GLSxtrlong}%
550 \newcommand*{\ACLP}{\GLSxtrlongpl}%
551 \newcommand*{\ACF}{\GLSxtrfull}%
552 \newcommand*{\ACFP}{\GLSxtrfullpl}%

553 \providecommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

554 \let\GlsXtrDefineAcShortcuts\relax
555 }

```

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

```

556 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
557   \newcommand*{\newentry}{\newglossaryentry}%
558   \ifdef\printsymbols
559   {%
560     \newcommand*{\newsym}{\glstrnewsymbol}%
561   }{}%
562   \ifdef\printnumbers
563   {%
564     \newcommand*{\newnum}{\glstrnewnumber}%
565   }{}%
566   \let\GlsXtrDefineOtherShortcuts\relax
567 }

```

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.

```

568 \newcommand*{\@glstr@setupshortcuts}{}

```

tr@shortcutsval Store the value of the shortcuts option. (Needed by bib2gls.)

```

569 \newcommand*{\@glstr@shortcutsval}{\ifglstrshortcuts acro\else none\fi}%

```

shortcuts 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. New to v1.17: shortcuts=ac which implements \GlsXtrDefineAcShortcuts (not included in shortcuts=all as it conflicts with other shortcuts).

```

570 \define@choicekey{glossaries-extra.sty}{shortcuts}%
571 [ \@glstr@shortcutsval \@glstr@shortcutsnr ]%
572 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
573   \ifcase \@glstr@shortcutsnr\relax % acronyms
574     \renewcommand*{\@glstr@setupshortcuts}{%
575       \glstrshortcuts true
576       \DefineAcronymSynonyms
577     }%
578   \or % acro
579     \renewcommand*{\@glstr@setupshortcuts}{%
580       \glstrshortcuts true
581       \DefineAcronymSynonyms
582     }%
583   \or % abbreviations
584     \renewcommand*{\@glstr@setupshortcuts}{%
585       \GlsXtrDefineAbbreviationShortcuts

```

```

586     }%
587 \or % abbr
588     \renewcommand*{\@glxtr@setupshortcuts}{%
589         \GlsXtrDefineAbbreviationShortcuts
590     }%
591 \or % other
592     \renewcommand*{\@glxtr@setupshortcuts}{%
593         \GlsXtrDefineOtherShortcuts
594     }%
595 \or % all
596     \renewcommand*{\@glxtr@setupshortcuts}{%
597         \glsacrshortcutstrue
598         \GlsXtrDefineAcShortcuts
599         \GlsXtrDefineAbbreviationShortcuts
600         \GlsXtrDefineOtherShortcuts
601     }%
602 \or % true
603     \renewcommand*{\@glxtr@setupshortcuts}{%
604         \glsacrshortcutstrue
605         \GlsXtrDefineAcShortcuts
606         \GlsXtrDefineAbbreviationShortcuts
607         \GlsXtrDefineOtherShortcuts
608     }%
609 \or % ac
610     \renewcommand*{\@glxtr@setupshortcuts}{%
611         \glsacrshortcutstrue
612         \GlsXtrDefineAcShortcuts
613     }%

```

Leave none and false as last option.

```

614 \else % none, false
615     \renewcommand*{\@glxtr@setupshortcuts}{}%
616 \fi
617 }

```

lsxtr@doaccsupp

```

618 \newcommand*{\@glxtr@doaccsupp}{%

```

accsupp If accsupp, load glossaries-accsupp package.

```

619 \@glxtr@declareoption{accsupp}{%
620 \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

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

```

621 \newcommand{\glxtrNoGlossaryWarning}[1]{%
622 \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
623 \@glxtr@defaultnoglossarywarning{#1}%
624 }

```

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

```

625 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
626 [ \@glxtr@nomissingglstextval \@glxtr@nomissingglstextnr ]%
627 {true,false}[true]{%
628   \ifcase \@glxtr@nomissingglstextnr \relax % true
629   \renewcommand{\glxtrNoGlossaryWarning}[1]{\null}%
630 \else % false
631   \renewcommand{\glxtrNoGlossaryWarning}[1]{%
632     \@glxtr@defaultnoglossarywarning{#1}%
633   }%
634 \fi
635 }

```

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

xtr@redefstyles

```

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

```

stylemods

```

637 \define@key{glossaries-extra.sty}{stylemods}[default]{%
638   \ifstrequal{#1}{default}%
639   {%
640     \renewcommand*{\@glxtr@redefstyles}{%
641       \RequirePackage{glossaries-extra-stylemods}}%
642   }%
643   {%
644     \ifstrequal{#1}{all}%
645     {%
646       \renewcommand*{\@glxtr@redefstyles}{%
647         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
648         \RequirePackage{glossaries-extra-stylemods}%
649       }%
650     }%
651     {%
652       \renewcommand*{\@glxtr@redefstyles}{}%
653       \@for \@glxtr@tmp:=#1\do{%
654         \IfFileExists{glossary-\@glxtr@tmp.sty}%
655         {%
656           \eappto \@glxtr@redefstyles{%
657             \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
658         }%
659         {%
660           \PackageError{glossaries-extra}%
661             {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
662              doesn’t exist (did you mean to use the ‘style’ key?)}%
663             {The list of values (#1) in the ‘stylemods’ key should
664              match the glossary-xxx.sty files provided with
665              glossaries.sty}%
666         }%
667       }%
668     }%
669   }%

```

```

667     }%
668     \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}}%
669   }
670   }%
671 }

```

`glsxtr@do@style`

```

672 \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.

```

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

```

Defer actual style change:

```

674 \renewcommand*{\@glsxtr@do@style}{%

```

Set this as the default style:

```

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

```

Set this style:

```

676 \setglossarystyle{#1}%

```

```

677 }%

```

```

678 }

```

`c@wrglossaryctr` Increments the associated counter if enabled. Does nothing by default. The optional argument is the entry label in case it's required, but the `wrglossary` counter is globally used by all entries.

```

679 \newcommand*{\glsxtr@inc@wrglossaryctr}[1]{%

```

`ocationHyperlink`

```

\glsxtrinternallocationhyperlink{<counter>}{<prefix>}{<location>}

```

The first two arguments are always control sequences.

```

680 \newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%

```

```

681 \glsxtrhyperlink{#1#2#3}{#3}%

```

```

682 }

```

`cationhyperlink`

```

683 \newcommand*{\@glsxtr@wrglossary@locationhyperlink}[3]{%

```

```

684 \pageref{wrglossary.#3}%

```

```

685 }

```

`indexcounter` Define the `wrglossary` counter that's incremented every time an entry is indexed, except for cross-references. This is designed for use with `bib2gls v1.4+`. It can work with the other indexing methods but it will interfere with the number list collation. This option automatically implements `counter=wrglossary`.

Since glossaries automatically loads amsmath, there may be a problem if the indexing occurs in the equation environment, because only one `\label` is allowed in each instance of that environment. It's best to change the counter when in maths mode.

```

686 \@glsxtr@declareoption{indexcounter}{%
687   \glsxtr@doption{counter=wrglossary}%
688   \ifundef\c@wrglossary
689   {%
690     \newcounter{wrglossary}%
691     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
692   }%
693   {}%
694   \renewcommand*\@glsxtr@inc@wrglossaryctr}[1]{%

```

Only increment if the current counter is wrglossary.

```

695   \ifdefstring\@gls@counter{wrglossary}%
696   {%
697     \refstepcounter{wrglossary}%
698     \label{wrglossary.\thewrglossary}%
699   }%
700   {}%
701 }%
702 \renewcommand*\@GlsXtrInternalLocationHyperlink[3]{%
703   \ifdefstring\glsentrycounter{wrglossary}%
704   {%
705     \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
706   }%
707   {\@glsxtrhyperlink{##1##2##3}{##3}}%
708 }%
709 }

```

`sxtrwrglossmark` Marks the place where indexing occurs. Does nothing by default.

```

710 \newcommand*\@glsxtrwrglossmark{}

```

`sxtrwrglossmark` Since `\glsadd` can be used in the preamble, this action needs to be disabled until the start of the document.

```

711 \newcommand*\@@glsxtrwrglossmark{}
712 \AtBeginDocument{\renewcommand*\@glsxtrwrglossmark{\@@glsxtrwrglossmark}}

```

`sxtrwrglossmark` Does nothing by default.

```

713 \newcommand*\@glsxtrwrglossmark{\ensuremath{\cdot}}

```

`debug` Provide extra debug options.

```

714 \define@choicekey{glossaries-extra.sty}{debug}
715 [\@glsxtr@debugval\@glsxtr@debugnr]%
716 {true,false,showtargets,showwrgloss,all}[true]{%
717   \ifcase\@glsxtr@debugnr\relax % true
718   \glsxtr@doption{debug=true}%
719   \renewcommand*\@glsxtrwrglossmark{}%

```

```

720 \or % false
721 \glstr@dooption{debug=false}%
722 \renewcommand*{\@glstrwrglossmark}{}%
723 \or % showtargets
724 \glstr@dooption{debug=showtargets}%
725 \or % showwrgloss
726 \glstr@dooption{debug=true}%
727 \renewcommand*{\@glstrwrglossmark}{\glstrwrglossmark}%
728 \or % all
729 \glstr@dooption{debug=showtargets}%
730 \renewcommand*{\@glstrwrglossmark}{\glstrwrglossmark}%
731 \fi
732 }

```

Pass all other options to glossaries.

```

733 \DeclareOptionX*{%
734 \expandafter\glstr@dooption\expandafter{\CurrentOption}}

```

Process options.

```
735 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
736 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
737 \@glstr@doaccsupp
```

Redefine \glspostdescription if required.

```
738 \@glstr@defpostpunc
```

`\glsshowtarget` This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using `\def`.

```

739 \def\glsshowtarget#1{%
740 \glstrtitleorpdforheading
741 {%
742 \ifmmode
743 \texttt{\small [#1]}%
744 \else
745 \ifinner
746 \texttt{\small [#1]}%
747 \else
748 \marginpar{\texttt{\small #1}}%
749 \fi
750 \fi
751 }%
752 {[#1]}%
753 {\texttt{\small [#1]}}%
754 }

```

`g@doseeglossary` Save original definition of `\@do@seeglossary`

```
755 \let\@glstr@org@doseeglossary\@do@seeglossary
```

`r@doseeglossary` This doesn't increment the associated counter.

```

756 \newcommand*{\@glxtr@doseeglossary}[2]{%
757   \glsoifexists{#1}%
758   {%
759     \@glxtrwrglossmark
760     \@glxtr@org@doseeglossary{#1}{#2}%
761   }%
762 }

```

`oindex@glossary`

```

763 \newcommand*{\@glxtr@dosee@alsoindex@glossary}[2]{%
764   \@glxtr@recordsee{#1}{#2}%
765   \@glxtr@doseeglossary{#1}{#2}%
766 }

```

`@org@gloautosee` Save and restore original definition of `\@glo@autosee`. (That command may not be defined as it was only introduced to glossaries v4.30, in which case the synonym won't be defined either.)

```

767 \let\@glxtr@org@gloautosee\@glo@autosee

```

Check if user tried `autoseeindex=false` when it can't be supported.

```

768 \if@glxtr@autoseeindex
769 \else
770   \ifdef\@glxtr@org@gloautosee
771     {}%
772     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
773       option requires at least v4.30 of glossaries.sty}%
774       {You need to update the glossaries.sty package}%
775     }
776 \fi

```

`\@glo@autosee` If `\@glo@autosee` has been defined (glossaries v4.30 onwards), redefine it to test the `autoseeindex` option.

```

777 \ifdef\@glo@autosee
778 {%
779   \renewcommand*{\@glo@autosee}{%
780     \if@glxtr@autoseeindex\@glxtr@org@gloautosee\fi}%
781 }%
782 {}

```

`checkseeallowed` Don't prohibit the use of the `see` key before the indexing files have been opened if the automatic `see` indexing has been disabled, since it's no longer an issue.

```

783 \renewcommand*{\gls@checkseeallowed}{%
784   \if@glxtr@autoseeindex\@gls@see@noindex\fi
785 }

```

Define abbreviations glossaries if required.

```

786 \@glxtr@abbreviationsdef
787 \let\@glxtr@abbreviationsdef\relax

```

Setup shortcuts if required.

```
788 \@glxtr@setupshortcuts
```

Redefine \@glxtr@redef@forlgsentries if required.

```
789 \@glxtr@redef@forlgsentries
```

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

```
790 \renewcommand{\glxtr@doption}[1]{\setupglossaries{#1}}%
```

Now define the user command:

```
791 \newcommand*{\glossariesextrasetup}[1]{%
792   \let\glxtr@setup@record\relax
793   \let\@glxtr@setupshortcuts\relax
794   \let\@glxtr@redef@forlgsentries\relax
795   \setkeys{glossaries-extra.sty}{#1}%
796   \@glxtr@abbreviationsdef
797   \let\@glxtr@abbreviationsdef\relax
798   \@glxtr@setupshortcuts
799   \glxtr@setup@record
800   \@glxtr@redef@forlgsentries
801 }
```

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

```
802 \let\glxtr@org@@do@wrglossary\@@do@wrglossary
```

@@do@wrglossary The new version adds code that can show a marker for debugging and increments the associated counter if enabled.

```
803 \newcommand*{\glxtr@@do@wrglossary}[1]{%
804   \@@glxtrwrglossmark
805   \glxtr@inc@wrglossaryctr{#1}%
806   \glxtr@org@@do@wrglossary{#1}%
807 }
```

aveentrycounter Save original definition of \@gls@saveentrycounter.

```
808 \let\glxtr@saveentrycounter\@gls@saveentrycounter
```

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

```
809 \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter
```

etcounterprefix This command is provided by the base glossaries package, but is redefined here. The standard indexing methods don't directly store the hypertarget but instead need to split it into the counter, prefix and location parts, which can be reconstituted in the location list. Unfortunately, not all targets are in this form, so the links fail. With record=nameref, the complete target name can be saved, so this modification adjusts the warning.

```
810 \renewcommand*\@gls@getcounterprefix[2]{%
811   \protected@edef\@gls@thisloc{#1}\protected@edef\@gls@thisHloc{#2}%
```

```

812 \ifx\@gls@thisloc\@gls@thisHloc
813   \def\@glo@counterprefix{}%
814 \else
815   \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
816     \def\@glo@tmp{##2}%
817     \ifx\@glo@tmp\@empty
818       \def\@glo@counterprefix{}%
819     \else
820       \def\@glo@counterprefix{##1}%
821     \fi
822   }%
823   \@gls@get@counterprefix#2.#1\end@getprefix
Warn if no prefix can be formed, unless record=nameref.
824   \ifx\@glo@counterprefix\@empty
825     \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
826     \else
827       \GlossariesExtraWarning{Hyper target ‘#2’ can’t be formed by
828         prefixing^^Jlocation ‘#1’. You need to modify the
829         definition of \string\theH\@gls@counter^^Jotherwise you
830         will get the warning: “‘name{\@gls@counter.#1}’ has been^^J
831         referenced but does not exist”%
832       \ifx\@glxtr@record@setting\@glxtr@record@setting@only
833         . You may want to consider using record=nameref instead%
834       \fi}%
835     \fi
836   \fi
837 \fi
838 }

```

Provide script dialect hook (does nothing unless redefined by glossaries-extra-bib2gls).

sxtrdialecthook

```

839 \newcommand*{\@glxtrdialecthook}{}

```

Set up record option if required.

```

840 \glxtr@setup@record

```

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

```

841 \AtBeginDocument{%
842   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
843   \def\@glxtrundeftag{\glxtrundeftag}%
844 }

```

1.2 Extra Utilities

unusedOrUndefined

```

\GlsXtrIfUnusedOrUndefined{<label>}{<true>}{<false>}

```

Does *<true>* if the entry given by *<label>* is either undefined or hasn't been used (or has had the first use flag reset).

```
845 \newcommand*{\GlsXtrIfUnusedOrUndefined}[3]{%
846   \ifglssentryexists{#1}%
847   {\ifbool{glo@\glstdetoklabel{#1}@flag}{#3}{#2}}%
848   {#2}}%
849 }
```

xtrifemptyglossary

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

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

```
850 \newcommand{\glsxtrifemptyglossary}[3]{%
851   \ifcsdef{glolist@#1}%
852   {%
853     \ifcsstring{glolist@#1}{,}{#2}{#3}%
854   }%
855   {%
856     \glsxtrundefaction{Glossary type '#1' doesn't exist}{}%
857     #2%
858   }%
859 }
```

xtrifkeydefined

Tests if the key given in the first argument has been defined.

```
860 \newcommand*{\glsxtrifkeydefined}[3]{%
861   \key@ifundefined{glossentry}{#1}{#3}{#2}%
862 }
```

providestoragekey

Like `\glssaddstoragekey` but does nothing if the key has already been defined.

```
863 \newcommand*{\glsxtrprovidestoragekey}{%
864   \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
865 }
```

vide@storagekey

Unstarred version.

```
866 \newcommand*{\@glsxtr@provide@storagekey}[3]{%
867   \key@ifundefined{glossentry}{#1}%
868   {%
869     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
870     \appto\@gls@keymap{,{#1}{#1}}%
871     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
872     \appto\@newglossaryentryposthook{%
873       \letcs{\@glo@tmp}{@glo@#1}%
874     }%
875   }%
876 }
```

```

874 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
875 }%

```

Allow the user to omit the user level command if they only intended fetching the value with `\glsxtrusefield`

```

876 \ifblank{#3}
877 {}%
878 {%
879 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
880 }%
881 }%
882 {%

```

Provide the no-link command if not already defined.

```

883 \ifblank{#3}
884 {}%
885 {%
886 \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
887 }%
888 }%
889 }

```

`\provide@storagekey` Starred version.

```

890 \newcommand*\s@glstr@provide@storagekey}[1]{%
891 \key@ifundefined{glossentry}{#1}%
892 {%
893 \expandafter\newcommand\expandafter*\expandafter
894 {\csname gls@assign@#1@field\endcsname}[2]{%
895 \@gls@expand@field{##1}{#1}{##2}%
896 }%
897 }%
898 {}%
899 \@glstr@provide@addstoragekey{#1}%
900 }

```

The name of a text-block control sequence can be stored in a field (given by `\GlsXtrFmtField`). This command can then be used with `\glsxtrfmt[⟨options⟩]{⟨label⟩}{⟨text⟩}` which effectively does `\glslink[⟨options⟩]{⟨label⟩}{⟨cs⟩{⟨text⟩}}` If the field hasn't been set for that entry just `⟨text⟩` is done.

`\GlsXtrFmtField`

```

901 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

902 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

`\glsxtrfmt` The post-link hook isn't done. This now has a starred form that checks for a final optional argument.

```

903 \newrobustcmd*\glsxtrfmt{\@ifstar\s@glsxtrfmt\@glsxtrfmt}

```

`\@glsxtrfmt` Unstarred form.

```
904 \newcommand*{\@glsxtrfmt}[3] [] {\@glsxtrfmt{#1}{#2}{#3}{}}
```

`\s@glsxtrfmt` Starred form.

```
905 \newcommand*{\s@glsxtrfmt}[3] [] {%
906   \new@ifnextchar[{\s@@glsxtrfmt{#1}{#2}{#3}}{%
907     {\@glsxtrfmt{#1}{#2}{#3}{}}%
908   }
```

`\s@@glsxtrfmt` Pick up final optional argument.

```
909 \def\s@@glsxtrfmt#1#2#3[#4] {\@glsxtrfmt{#1}{#2}{#3}{#4}}
```

`\@@glsxtrfmt` Actual inner working.

```
910 \newcommand*{\@@glsxtrfmt}[4] {%
```

Since there's no post-link hook to worry about, grouping can be added to provide some protection against nesting (but in general nested link text should be avoided).

```
911   \begingroup
912   \def\glslabel{#2}%
913   \glsdoifexistsordo{#2}%
914   {%
915     \ifglshasfield{\GlsXtrFmtField}{#2}%
916     {%
917       \let\do@glsl@link@checkfirsthyper\relax
918       \expandafter\@glsl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
919       {\glsxtrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
920     }%
921     {\glsxtrfmtdisplay{@firstofone}{#3}{#4}}%
922   }%
923   {%
```

Has the default `noindex` been counteracted? If so, this needs `\glsadd` in case `bib2gls` needs to pick up the record.

```
924   \begingroup
925   \@glsl@setdefault@glslink@opts
926   \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
927   \ifKV@glslink@noindex\else\glsadd{#2}\fi
928   \endgroup
929   \glsxtrfmtdisplay{@firstofone}{#3}{#4}%
930   }%
931 \endgroup
932 }
```

`\glsxtrfmtdisplay` The command used internally by `\glsxtrfmt` to do the actual formatting. The first argument is the control sequence name, the second is the control sequence's argument, the third is the inserted material (if starred form used).

```
933 \newcommand{\glsxtrfmtdisplay}[3] {\csuse{#1}{#2}#3}
```

`\glxxtreentryfmt` No link or indexing.

```

934 \ifdef\teorpdfstring
935 {
936   \newcommand*\glxxtreentryfmt}[2]{%
937     \teorpdfstring{\@glxxtreentryfmt{#1}{#2}}{#2}%
938   }
939 }
940 {
941   \newcommand*\glxxtreentryfmt{\@glxxtreentryfmt}
942 }

```

`@glxxtreentryfmt`

```

943 \newrobustcmd*\@glxxtreentryfmt}[2]{%
944   \glsoifexistsordo{#1}%
945   {%
946     \ifglshasfield{\GlsXtrFmtField}{#1}%
947     {%
948       \csuse{\glscurrentfieldvalue}{#2}%
949     }%
950     {#2}%
951   }%
952   {#2}%
953 }

```

`xtrfieldlistadd` If a field stores an etoolbox internal list (e.g. `loclist`) then this macro provides a convenient way of adding to the list via etoolbox's `\listcsadd`. The first argument is the entry's label, the second is the field label and the third is the element to add to the list.

```

954 \newcommand*\glxxttrfieldlistadd}[3]{%
955   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
956 }

```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```

957 \newcommand*\glxxttrfieldlistgadd}[3]{%
958   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
959 }

```

`trfieldlistead` Similarly but uses `\listcseadd`.

```

960 \newcommand*\glxxttrfieldlistead}[3]{%
961   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
962 }

```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```

963 \newcommand*\glxxttrfieldlistxadd}[3]{%
964   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
965 }

```

Now provide commands to iterate over these lists.

fielddolistloop

```
966 \newcommand*\glxtrfielddolistloop}[2]{%
967   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
968 }
```

fieldforlistloop

```
969 \newcommand*\glxtrfieldforlistloop}[3]{%
970   \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
971 }
```

List element tests:

trfieldifinlist First argument label, second argument field, third argument item, fourth true part and fifth false part.

```
972 \newcommand*\glxtrfieldifinlist}[5]{%
973   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
974 }
```

rfieldxifinlist Expands item.

```
975 \newcommand*\glxtrfieldxifinlist}[5]{%
976   \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
977 }
```

lsxtrforcsvfield `\glxtrforcsvfield{<label>}{<field>}{<cs handler>}`

```
978 \newcommand*\glxtrforcsvfield}[3]{%
979   \@glxtrifhasfield{#2}{#1}%
980   {%
981     \let\glxxtrendfor\endfortrue
982     \@for\@glxtr@label:=\glscurrentfieldvalue\do
983     {\expandafter#3\expandafter{\@glxtr@label}}}%
984   }%
985 }
```

lsxtrifhasfield A simpler alternative to `\ifglshasfield` that doesn't complain if the entry or the field doesn't exist. (No mapping is used.) Grouping is added to the unstarred version allow for nested use.

```
986 \newrobustcmd{\glxtrifhasfield}{%
987   \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
988 }
```

lsxtrifhasfield Unstarred version adds grouping.

```
989 \newcommand{\@glxtrifhasfield}[4]{%
990   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
991 }
```

lsxtrifhasfield Starred version omits grouping.

```

992 \newcommand{\s@glxtrifhasfield}[4]{%
993   \letcs{\glscurrentfieldvalue}{glo@glsetoklabel{#2}@#1}%
994   \ifundef\glscurrentfieldvalue
995     {#4}%
996     {%
997       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
998     }%
999 }
```

rIfFieldNonZero Designed for numeric fields.

```

1000 \newcommand{\GlsXtrIfFieldNonZero}{%
1001   \@ifstar\s@GlsXtrIfFieldNonZero\@GlsXtrIfFieldNonZero
1002 }
```

rIfFieldNonZero

```

1003 \newcommand{\@GlsXtrIfFieldNonZero}[4]{%
1004   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
1005 }
```

sXtrIfFieldEqNum

`\GlsXtrIfFieldEqNum{<field>}{<label>}{<value>}{<true>}{<false>}`

Designed for numeric fields.

```

1006 \newcommand{\GlsXtrIfFieldEqNum}{%
1007   \@ifstar\s@GlsXtrIfFieldEqNum\@GlsXtrIfFieldEqNum
1008 }
```

XtrIfFieldEqNum

```

1009 \newcommand{\@GlsXtrIfFieldEqNum}[5]{%
1010   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1011 }
```

XtrIfFieldEqNum

```

1012 \newcommand{\s@GlsXtrIfFieldEqNum}[5]{%
1013   \s@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1014 }
```

XtrIfFieldCmpNum

`\GlsXtrIfFieldCmpNum{<field>}{<label>}{<comparison>}{<value>}{<true>}{<false>}`

Designed for numeric fields.

```

1015 \newcommand{\GlsXtrIfFieldCmpNum}{%
1016   \@ifstar\s@GlsXtrIfFieldCmpNum\@GlsXtrIfFieldCmpNum
1017 }
```

trIfFieldCmpNum

```

1018 \newcommand{\@GlsXtrIfFieldCmpNum}[6]{%
1019   {%
1020     \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
1021     \ifundef\glscurrentfieldvalue
1022       {\def\glscurrentfieldvalue{0}}%
1023     {%
1024       \ifdefempty\glscurrentfieldvalue
1025         {\def\glscurrentfieldvalue{0}}%
1026       {%
1027         }%
1028       \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1029     }%
1030 }

```

trIfFieldCmpNum

```

1031 \newcommand{\s@GlsXtrIfFieldCmpNum}[6]{%
1032   \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
1033   \ifundef\glscurrentfieldvalue
1034     {\def\glscurrentfieldvalue{0}}%
1035   {%
1036     \ifdefempty\glscurrentfieldvalue
1037       {\def\glscurrentfieldvalue{0}}%
1038     {%
1039       }%
1040     \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1041 }

```

sXtrIfFieldUndef

`\GlsXtrIfFieldUndef{<field>}{<label>}{<true>}{<false>}`

Just uses \ifcsundef.

```

1042 \newcommand{\GlsXtrIfFieldUndef}[2]{%
1043   \ifcsundef{glo@\glsetoklabel{#2}@#1}%
1044 }

```

\glsxtrusefield

Provide a user-level alternative to \@gls@entry@field. The first argument is the entry label. The second argument is the field label.

```

1045 \newcommand*\glsxtrusefield}[2]{%
1046   \@gls@entry@field{#1}{#2}%
1047 }

```

\Glsxtrusefield

Provide a user-level alternative to \@Gls@entry@field.

```

1048 \ifdef\texorpdfstring
1049 {
1050   \newcommand*\Glsxtrusefield}[2]{%
1051     \texorpdfstring

```

```

1052     {\@Gls@entry@field{#1}{#2}}
1053     {\@gls@entry@field{#1}{#2}}%
1054   }
1055 }
1056 {
1057   \newcommand*{\Glsxtrusefield}[2]{%
1058     \@Gls@entry@field{#1}{#2}%
1059   }
1060 }

```

`\Glsxtrusefield` As above but convert to all caps.

```

1061 \ifdef\teorpdfstring
1062 {
1063   \newcommand*{\Glsxtrusefield}[2]{%
1064     \teorpdfstring
1065       {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}}%
1066       {\@gls@entry@field{#1}{#2}}%
1067   }
1068 }
1069 {
1070   \newcommand*{\Glsxtrusefield}[2]{%
1071     \glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}%
1072   }
1073 }

```

`entryparentname`

```

1074 \newcommand*{\glsxtrentryparentname}[1]{%
1075   \ifcsdef{glo@glstdetoklabel{#1}@parent}%
1076     {\csuse{glo@csuse{glo@glstdetoklabel{#1}@parent}@name}}%
1077     {}%
1078 }

```

`\glsxtrdeffield` Just use `\csdef` to provide a field value for the given entry.

```

1079 \newcommand*{\glsxtrdeffield}[2]{\csdef{glo@glstdetoklabel{#1}@#2}}

```

`glsxtredeffield` Just use `\csdef` to provide a field value for the given entry.

```

1080 \newcommand*{\glsxtredeffield}[2]{\protected@csdef{glo@glstdetoklabel{#1}@#2}}

```

`etfieldifexists`

```

1081 \newcommand*{\glsxtrsetfieldifexists}[3]{\glsdoifexists{#1}{#3}}

```

`\GlsXtrSetField` Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```

1082 \newrobustcmd*{\GlsXtrSetField}[3]{%
1083   \glsxtrsetfieldifexists{#1}{#2}%
1084   {\csdef{glo@glstdetoklabel{#1}@#2}{#3}}%
1085 }

```

`\GlsXtrLetField` Uses `\cslet` instead. Third argument should be a macro.

```

1086 \newrobustcmd*{\GlsXtrLetField}[3]{%
1087   \glstrsetfieldifexists{#1}{#2}%
1088   {\cslet{glo@glstoklabel{#1}@#2}{#3}}%
1089 }

```

`sGlsXtrLetField` Uses `\csletcs` instead. Third argument should be a control sequence name.

```

1090 \newrobustcmd*{\csGlsXtrLetField}[3]{%
1091   \glstrsetfieldifexists{#1}{#2}%
1092   {\csletcs{glo@glstoklabel{#1}@#2}{#3}}%
1093 }

```

`LetFieldToField` Sets the field for one entry to the field for another entry. Third argument should be the other entry and the fourth argument that other field label.

```

1094 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
1095   \glstrsetfieldifexists{#1}{#2}%
1096   {\csletcs{glo@glstoklabel{#1}@#2}{glo@glstoklabel{#3}@#4}}%
1097 }

```

`gGlsXtrSetField` Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```

1098 \newrobustcmd*{\gGlsXtrSetField}[3]{%
1099   \glstrsetfieldifexists{#1}{#2}%
1100   {\csgdef{glo@glstoklabel{#1}@#2}{#3}}%
1101 }

```

`xGlsXtrSetField`

```

1102 \newrobustcmd*{\xGlsXtrSetField}[3]{%
1103   \glstrsetfieldifexists{#1}{#2}%
1104   {\protected@csxdef{glo@glstoklabel{#1}@#2}{#3}}%
1105 }

```

`eGlsXtrSetField`

```

1106 \newrobustcmd*{\eGlsXtrSetField}[3]{%
1107   \glstrsetfieldifexists{#1}{#2}%
1108   {\protected@csedef{glo@glstoklabel{#1}@#2}{#3}}%
1109 }

```

`XtrIfFieldEqStr` Starred version uses starred version of `\glstrifhasfield` (that is, no grouping).

```

1110 \newcommand*{\GlsXtrIfFieldEqStr}{%
1111   \@ifstar\sGlsXtrIfFieldEqStr\@GlsXtrIfFieldEqStr
1112 }

```

`XtrIfFieldEqStr`

```

1113 \newrobustcmd*{\@GlsXtrIfFieldEqStr}[5]{%
1114   \@glstrifhasfield{#1}{#2}%
1115   {%
1116     \ifdefstring{glscurrentfieldvalue}{#3}{#4}{#5}%

```

```

1117 }%
1118 {#5}%
1119 }

```

XtrIfFieldEqStr

```

1120 \newrobustcmd*{\s@GlsXtrIfFieldEqStr}[5]{%
1121   \s@glstrifhasfield{#1}{#2}%
1122   {%
1123     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1124   }%
1125   {#5}%
1126 }

```

rIfFieldEqXpStr Like the above but first expands the string. Starred version uses starred version of \glstrifhasfield (that is, no grouping).

```

1127 \newcommand*{\GlsXtrIfFieldEqXpStr}{%
1128   \@ifstar\s@GlsXtrIfFieldEqXpStr\@GlsXtrIfFieldEqXpStr
1129 }

```

rIfFieldEqXpStr

```

1130 \newrobustcmd*{\@GlsXtrIfFieldEqXpStr}[5]{%
1131   \@glstrifhasfield{#1}{#2}%
1132   {%
1133     \protected@edef\@glstmp{#3}%
1134     \ifdefequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1135   }%
1136   {#5}%
1137 }

```

rIfFieldEqXpStr

```

1138 \newrobustcmd*{\s@GlsXtrIfFieldEqXpStr}[5]{%
1139   \s@glstrifhasfield{#1}{#2}%
1140   {%
1141     \protected@edef\@glstmp{#3}%
1142     \ifdefequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1143   }%
1144   {#5}%
1145 }

```

fXpFieldEqXpStr Like the above but also expands the field value. Starred version uses starred version of \glstrifhasfield (that is, no grouping).

```

1146 \newcommand*{\GlsXtrIfXpFieldEqXpStr}{%
1147   \@ifstar\s@GlsXtrIfXpFieldEqXpStr\@GlsXtrIfXpFieldEqXpStr
1148 }

```

fXpFieldEqXpStr

```

1149 \newrobustcmd*{\@GlsXtrIfXpFieldEqXpStr}[5]{%
1150   \@glstrifhasfield{#1}{#2}%

```

```

1151 {%
1152   \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
1153   \let\glscurrentfieldvalue\@gls@tmp
1154   \protected@edef\@gls@tmp{#3}%
1155   \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1156 }%
1157 {#5}%
1158 }

```

fXpFieldEqXpStr

```

1159 \newrobustcmd*{\s@GlsXtrIfXpFieldEqXpStr}[5]{%
1160   \s@glstrifhasfield{#1}{#2}%
1161   {%
1162     \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
1163     \let\glscurrentfieldvalue\@gls@tmp
1164     \protected@edef\@gls@tmp{#3}%
1165     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1166   }%
1167   {#5}%
1168 }

```

lsXtrForeignText

<code>\GlsXtrForeignText{<entry label>}{<text>}</code>
--

If a field is used to store a language tag (such as en-GB or de-CH-1996) then this command uses tracklang's interface to encapsulate *<text>*. The field identifying the locale is given by `\GlsXtrForeignTextField`.

```

1169 \ifdef\foreignlanguage
1170 {
1171   \ifdef\GetTrackedDialectFromLanguageTag
1172   {
1173     \newcommand{\GlsXtrForeignText}[2]{%

```

In case this is used inside the argument of `\glstrifhasfield`, save and restore `\glscurrentfieldvalue`.

```

1174     \let\@glstr@org@currentfieldvalue\glscurrentfieldvalue
1175     \glstrifhasfield{\GlsXtrForeignTextField}{#1}%
1176     {%
1177       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
1178       {\glscurrentfieldvalue}{\@glstr@dialect}%
1179       \let\@glstr@locale\glscurrentfieldvalue
1180       \let\glscurrentfieldvalue\@glstr@org@currentfieldvalue
1181       \ifdefempty\@glstr@dialect
1182       {%

```

An exact match hasn't been found. A partial match can only be obtained with at least tracklang v1.3.6.

```

1183         \ifundef\TrackedDialectClosestSubMatch
1184         {%

```

```

1185         \GlossariesExtraWarning{Can't obtain dialect label
1186         (tracklang v1.3.6+ required)}%
1187     }%
1188     {\let\@glxtr@dialect\TrackedDialectClosestSubMatch}%
1189 }%
1190 {}%
1191 \ifdefempty\@glxtr@dialect
1192 {%
    No tracked dialect found for the root language.
1193 }%
1194 {%
    Check if there's a caption hook for the given dialect label.
1195     \ifcsundef{captions\@glxtr@dialect}{}%
1196     {%
        Dialect label not recognised. Check if there's a known mapping.
1197         \IfTrackedDialectHasMapping{\@glxtr@dialect}%
1198         {%
1199             \edef\@glxtr@dialect{%
1200                 \GetTrackedDialectToMapping{\@glxtr@dialect}}%
        Does a caption hook exist for this?
1201         \ifcsundef{captions\@glxtr@dialect}{}%
1202         {%
            No mapping. Try root language label instead.
1203             \ifcsundef{captions\@tracklang@lang}{}%
1204             {%
1205                 \let\@glxtr@dialect\@tracklang@lang
1206             }%
1207         }%
1208     }%
1209     {%
        No mapping. Try root language label instead.
1210         \ifcsundef{captions\@tracklang@lang}{}%
1211         {%
1212             \let\@glxtr@dialect\@tracklang@lang
1213         }%
1214     }%
1215 }%
1216 }%
1217 \ifdefempty\@glxtr@dialect
1218 {%
1219     \GlsXtrUnknownDialectWarning{\@glxtr@locale}{\@tracklang@lang}%
1220     #2%
1221 }%
1222 {\foreignlanguage{\@glxtr@dialect}{#2}}%
1223 }%
1224 {#2}% key not set

```

```

1225     }
1226   }
1227   {
1228     \newcommand{\GlsXtrForeignText}[2]{%
1229       \GlossariesExtraWarning{Can't encapsulate foreign text:
1230         tracklang v1.3.6+ required}%
1231       #2%
1232     }
1233   }
1234 }
1235 {
  \foreignlanguage isn't defined so just do <text>.
1236 \newcommand{\GlsXtrForeignText}[2]{#2}
1237 }

```

`\foreignTextField` This is the `user2` field by default but may be redefined as required.

```

1238 \newcommand*{\GlsXtrForeignTextField}{userii}

```

`\nDialectWarning`

```

1239 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
1240   \GlossariesExtraWarning{Can't determine valid dialect label
1241     for locale '#1' (root language: #2)}%
1242 }

```

`\glstrpageref` Like `\glsrefentry` but references the page number instead (if entry counting is on). The base glossaries package only introduced `\GlsEntryCounterLabelPrefix` in version 4.38, so it may not be defined.

```

1243 \ifdef\GlsEntryCounterLabelPrefix
1244 {%
1245   \newcommand*{\glstrpageref}[1]{%
1246     \ifglentrycounter
1247       \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1248     \else
1249       \ifglssubentrycounter
1250         \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1251       \else
1252         \gls{#1}%
1253       \fi
1254     \fi
1255   }
1256 }%
1257 {%
1258   \newcommand*{\glstrpageref}[1]{%
1259     \ifglentrycounter
1260       \pageref{glentry-\glsdetoklabel{#1}}%
1261     \else
1262       \ifglssubentrycounter
1263         \pageref{glentry-\glsdetoklabel{#1}}%

```

```

1264     \else
1265         \gls{#1}%
1266     \fi
1267 \fi
1268 }
1269 }%

```

lossarypreamble

```

1270 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1271   \ifcsdef{glolist@#1}%
1272   {%
1273     \ifcsundef{@glossarypreamble@#1}%
1274     {\csdef{@glossarypreamble@#1}{}}%
1275   }%
1276   \csappto{@glossarypreamble@#1}{#2}%
1277 }%
1278 {%
1279   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1280 }%
1281 }

```

lossarypreamble

```

1282 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1283   \ifcsdef{glolist@#1}%
1284   {%
1285     \ifcsundef{@glossarypreamble@#1}%
1286     {\csdef{@glossarypreamble@#1}{}}%
1287   }%
1288   \cspreto{@glossarypreamble@#1}{#2}%
1289 }%
1290 {%
1291   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1292 }%
1293 }

```

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.

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

In the event that undefined entries should trigger a warning rather than an error, `\ifglsused` needs to be modified to check for existence. If the boolean variable is undefined, then its

state is indeterminate and is neither true nor false, so neither *<true part>* nor *<false>* part will be performed if *<label>* is undefined.

```
1294 \renewcommand*{\ifglsused}[3]{%
1295   \glsdoifexists{#1}{\ifbool{glo@glstdetoklabel{#1}@flag}{#2}{#3}}%
1296 }
```

Provide a starred version of `\longnewglossaryentry` that doesn't automatically insert `\leavevmode\unskip\nopostdesc` at the end of the description. The unstarred version is modified to use `\glstrpostlongdescription` instead.

`ewglossaryentry`

```
1297 \renewcommand*{\longnewglossaryentry}{%
1298   \@ifstar{\glstr@s@longnewglossaryentry\glstr@longnewglossaryentry
1299 }
```

`ewglossaryentry` Starred version.

```
1300 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1301   \glsdoifnoexists{#1}%
1302   {%
1303     \bgroup
1304     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1305     \long\def\@newglossaryentryprehook{%
1306       \long\def\@glo@desc{#3}%
1307       \@org@newglossaryentryprehook
1308     }%
1309     \renewcommand*{\gls@assign@desc}[1]{%
1310       \global\cslet{glo@glstdetoklabel{#1}@desc}{\@glo@desc}%
1311       \global\cslet{glo@glstdetoklabel{#1}@descplural}{\@glo@descplural}%
1312     }
1313     \gls@defglossaryentry{#1}{#2}%
1314   \egroup
1315   }%
1316 }
```

`ewglossaryentry` Unstarred version.

```
1317 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1318   \glsdoifnoexists{#1}%
1319   {%
1320     \bgroup
1321     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1322     \long\def\@newglossaryentryprehook{%
1323       \long\def\@glo@desc{#3\glstrpostlongdescription}%
1324       \@org@newglossaryentryprehook
1325     }%
1326     \renewcommand*{\gls@assign@desc}[1]{%
1327       \global\cslet{glo@glstdetoklabel{#1}@desc}{\@glo@desc}%
1328       \global\cslet{glo@glstdetoklabel{#1}@descplural}{\@glo@descplural}%
1329     }
```

The following is different from the base glossaries.sty:

```
1328   \global\cslet{glo@glstdetoklabel{#1}@descplural}{\@glo@descplural}%
1329 }
```

```

1329     }
1330     \gls@defglossaryentry{#1}{#2}%
1331 \egroup
1332 }%
1333 }

```

`\longdescription` Hook at the end of the description when using the unstarred `\longnewglossaryentry`.

```

1334 \newcommand*{\glsxtrpostlongdescription}{\leavevmode\unskip\nopostdesc}

```

Provide a starred version of `\newignoredglossary` that doesn't add the glossary to the `nohyperlist` list.

`\ignoredglossary` Redefine to check for star.

```

1335 \renewcommand{\newignoredglossary}{%
1336 \ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
1337 }

```

`\ignoredglossary` The original definition is patched to check for existence.

```

1338 \newcommand*{\glsxtr@org@newignoredglossary}[1]{%
1339 \ifcsdef{glolist@#1}
1340 {%
1341 \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
1342 }%
1343 {%
1344 \ifdefempty\@ignored@glossaries
1345 {%
1346 \edef\@ignored@glossaries{#1}%
1347 }%
1348 {%
1349 \eappto\@ignored@glossaries{,#1}%
1350 }%
1351 \csgdef{glolist@#1}{,%}%
1352 \ifcsundef{gls@#1@entryfmt}%
1353 {%
1354 \defglsentryfmt[#1]{\glsentryfmt}%
1355 }%
1356 {}%
1357 \ifdefempty\@gls@nohyperlist
1358 {%
1359 \renewcommand*{\@gls@nohyperlist}{#1}%
1360 }%
1361 {%
1362 \eappto\@gls@nohyperlist{,#1}%
1363 }%
1364 }%
1365 }

```

`\ignoredglossary` Starred form.

```

1366 \newcommand*{\glsxtr@s@newignoredglossary}[1]{%

```

```

1367 \ifcsdef{glolist@#1}
1368 {%
1369   \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1370 }%
1371 {%
1372   \ifdefempty\@ignored@glossaries
1373   {%
1374     \edef\@ignored@glossaries{#1}%
1375   }%
1376   {%
1377     \eappto\@ignored@glossaries{,#1}%
1378   }%
1379   \csgdef{glolist@#1}{,}%
1380   \ifcsundef{gls@#1@entryfmt}%
1381   {%
1382     \defglsentryfmt[#1]{\glsentryfmt}%
1383   }%
1384   {}%
1385 }%
1386 }

```

`\glsettoctitle` Ignored glossaries don't have an associated title, so modify `\glsettoctitle` to check for it to prevent an undefined command written to the toc file.

```

1387 \glsifusetranslator
1388 {%
1389   \renewcommand*{\glsettoctitle}[1]{%
1390     \ifcsdef{gls@tr@set@#1@toctitle}%
1391     {%
1392       \csuse{gls@tr@set@#1@toctitle}%
1393     }%
1394     {%
1395       \ifcsdef{@glotype@#1@title}%
1396       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1397       {\def\glossarytoctitle{\glossarytitle}}%
1398     }%
1399   }%
1400 }
1401 {
1402   \renewcommand*{\glsettoctitle}[1]{%
1403     \ifcsdef{@glotype@#1@title}%
1404     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1405     {\def\glossarytoctitle{\glossarytitle}}%
1406   }
1407 }

```

`ignoredglossary` As above but won't do anything if the glossary already exists.

```

1408 \newcommand{\provideignoredglossary}{%
1409   \@ifstar\glstr@s@provideignoredglossary\glstr@provideignoredglossary
1410 }

```

ignoredglossary Unstarred version.

```
1411 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1412   \ifcsdef{glolist@#1}
1413   {}%
1414   {%
1415     \ifdefempty\@ignored@glossaries
1416     {%
1417       \edef\@ignored@glossaries{#1}%
1418     }%
1419     {%
1420       \eappto\@ignored@glossaries{,#1}%
1421     }%
1422     \csgdef{glolist@#1}{,}%
1423     \ifcsundef{gls@#1@entryfmt}%
1424     {%
1425       \defglsentryfmt[#1]{\glsentryfmt}%
1426     }%
1427     {}%
1428     \ifdefempty\@gls@nohyperlist
1429     {%
1430       \renewcommand*{\@gls@nohyperlist}{#1}%
1431     }%
1432     {%
1433       \eappto\@gls@nohyperlist{,#1}%
1434     }%
1435   }%
1436 }
```

ignoredglossary Starred form.

```
1437 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1438   \ifcsdef{glolist@#1}
1439   {}%
1440   {%
1441     \ifdefempty\@ignored@glossaries
1442     {%
1443       \edef\@ignored@glossaries{#1}%
1444     }%
1445     {%
1446       \eappto\@ignored@glossaries{,#1}%
1447     }%
1448     \csgdef{glolist@#1}{,}%
1449     \ifcsundef{gls@#1@entryfmt}%
1450     {%
1451       \defglsentryfmt[#1]{\glsentryfmt}%
1452     }%
1453     {}%
1454   }%
1455 }
```

`\trcopytoglossary` Adds an entry label to another glossary list. First argument is entry label. Second argument is glossary label.

```

1456 \newcommand*{\glxtrcopytoglossary}[2]{%
1457   \glsdoidexists{#1}%
1458   {%
1459     \ifcsdef{glolist@#2}
1460     {%
1461       \cseappto{glolist@#2}{#1,}%
1462     }%
1463     {%
1464       \glxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1465     }%
1466   }%
1467 }

```

1.3.1 Existence Checks

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

```

1468 \renewcommand{\glsdoidexists}[2]{%
1469   \ifglstryexists{#1}{#2}%
1470   {%

```

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

```

1471   \edef\glslabel{\glsdetoklabel{#1}}%
1472   \glxtrundefaction{Glossary entry ‘\glslabel’
1473     has not been defined}{You need to define a glossary entry before
1474     you can reference it.}%
1475   }%
1476 }

```

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

```

1477 \renewcommand{\glsdoidnoexists}[2]{%
1478   \ifglstryexists{#1}{%
1479     \glxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1480       has already been defined}{}}{#2}%
1481 }

```

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

```

1482 \ifdef\glsdoidexistsordo
1483 {%
1484   \renewcommand{\glsdoidexistsordo}[3]{%
1485     \ifglstryexists{#1}{#2}%
1486     {%
1487       \glxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1488         has not been defined}{You need to define a glossary entry
1489         before you can use it.}%

```

```

1490      #3%
1491    }%
1492  }%
1493 }
1494 {%
1495   \glstr@warnonexistsordo\glsoifexistsordo
1496   \newcommand{\glsoifexistsordo}[3]{%
1497     \ifglstryexists{#1}{#2}%
1498     {%
1499       \glstrundefaction{Glossary entry ‘\glstoklabel{#1}’
1500       has not been defined}{You need to define a glossary entry
1501       before you can use it.}%
1502     #3%
1503     }%
1504   }%
1505 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

1506 \ifdef\doifglossarynoexistsordo
1507 {%
1508   \renewcommand{\doifglossarynoexistsordo}[3]{%
1509     \ifglossaryexists{#1}%
1510     {%
1511       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1512     #3%
1513     }%
1514     {#2}%
1515   }%
1516 }
1517 {%
1518   \glstr@warnonexistsordo\doifglossarynoexistsordo
1519   \newcommand{\doifglossarynoexistsordo}[3]{%
1520     \ifglossaryexists{#1}%
1521     {%
1522       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1523     #3%
1524     }%
1525     {#2}%
1526   }%
1527 }
1528

```

There are now three types of cross-references: the see key (as original), the alias key (from glossaries-extra v1.12) and the seealso key (from glossaries-extra v1.16). The original see key needs to have a corresponding field (which it doesn't with the base glossaries package).

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

```

1529 \appto\newglossaryentryposthook{%
1530   \ifdefvoid\glo@see

```

```

1531   {\csxdef{glo@\@glo@label @see}{}}%
1532   {%
1533     \csxdef{glo@\@glo@label @see}{\@glo@see}%
1534     \if@glxtr@autoseeindex
1535       \@glxtr@autoindexcrossrefs
1536     \fi
1537   }%
1538 }
1539 \appto\@gls@keymap{,{see}{see}}

```

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

```

1540 \newcommand*{\glxtrusesee}[1]{%
1541   \glsdoifexists{#1}%
1542   {%
1543     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1544     \ifdefempty\@glo@see
1545     {}%
1546     {%
1547       \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1548     }%
1549   }%
1550 }

```

`\glxtr@usesee`

```

1551 \newcommand*{\glxtr@usesee}[1][\seename]{%
1552   \@glxtr@usesee[#1]%
1553 }

```

`\@glxtr@usesee`

```

1554 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1555   \glxtruseseeformat{#1}{#2}%
1556 }

```

`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.

```

1557 \newcommand*{\glxtruseseeformat}[2]{%
1558   \glsseeformat{#1}{#2}{}%
1559 }

```

`lsseeitemformat` glossaries originally defined `\glsseeitemformat` to use `\glsentryname` but in v3.0 this was switched to use `\glsentrytext` due to problems occurring with the name field being sanitized. Since this is no longer a problem, glossaries-extra restores the original definition as it makes more sense to use the name in the cross-reference list. This still uses `\glsaccesstext` for abbreviations.

```

1560 \renewcommand*{\glsseeitemformat}[1]{%
1561   \ifglshasshort{#1}{\glsaccesstext{#1}}{\glsaccessname{#1}}%
1562 }

```

`\glxtrhiername`

`\glxtrhiername{<label>}`

Displays the hierarchical name for the given entry. The cross-reference format `\glsseeitemformat` may be redefined to use this command to show the hierarchy, if required.

```
1563 \newcommand*{\glxtrhiername}[1]{%
1564   \glsdoifexists{#1}%
1565   {%
1566     \glxtrifhasfield{parent}{#1}%
1567     {\glxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1568     {}%
1569     \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
1570   }%
1571 }
```

`\Glsxtrhiername`

`\Glsxtrhiername{<label>}`

As above but displays the top-level name with an initial capital.

```
1572 \newcommand*{\Glsxtrhiername}[1]{%
1573   \glsdoifexists{#1}%
1574   {%
1575     \glxtrifhasfield{parent}{#1}%
1576     {%
1577       \Glsxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1578       \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
1579     }%
1580     {\ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}}%
1581   }%
1582 }
```

`\GlsXtrhiername`

`\GlsXtrhiername{<label>}`

As above but converts the first letter of each name to a capital.

```
1583 \newcommand*{\GlsXtrhiername}[1]{%
1584   \glsdoifexists{#1}%
1585   {%
1586     \glxtrifhasfield{parent}{#1}%
1587     {\GlsXtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1588     {}%
1589     \ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}%
1590   }%
1591 }
```

\GLSxtrhiername

\GLSxtrhiername{<label>}

As above but displays the top-level name in all-caps.

```
1592 \newcommand*{\GLSxtrhiername}[1]{%
1593   \glsdoifexists{#1}%
1594   {%
1595     \glsxtrifhasfield{parent}{#1}%
1596     {%
1597       \GLSxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
1598       \ifglshasshort{#1}{\glsaccesssshort{#1}}{\glsaccessname{#1}}%
1599     }%
1600     {\ifglshasshort{#1}{\GLSaccesssshort{#1}}{\GLSaccessname{#1}}}%
1601   }%
1602 }
```

\GLSXTRhiername

\GLSXTRhiername{<label>}

As above but displays all names in all-caps.

```
1603 \newcommand*{\GLSXTRhiername}[1]{%
1604   \glsdoifexists{#1}%
1605   {%
1606     \glsxtrifhasfield{parent}{#1}%
1607     {\GLSXTRhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
1608     {}
1609     \ifglshasshort{#1}{\GLSaccesssshort{#1}}{\GLSaccessname{#1}}%
1610   }%
1611 }
```

\glsxtrhiernamesep Separator used in \glsxtrhiername and variants.

```
1612 \newcommand*{\glsxtrhiernamesep}{\,\small$\triangleright$}\,}
```

\glsxtruseseealso Apply \glsseeformat to the seealso key if not empty. There's no optional tag to worry about here.

```
1613 \newcommand*{\glsxtruseseealso}[1]{%
1614   \glsdoifexists{#1}%
1615   {%
1616     \letcs{\@glo@see}{glo\glsdetoklabel{#1}@seealso}%
1617     \ifdefempty\@glo@see
1618     {}%
1619     {%
1620       \expandafter\glsxtruseseealsoformat\expandafter{\@glo@see}%
1621     }%
1622   }%
1623 }
```

`\seeseealsoformat` The format used by `\glxtruseeseealso`. The argument is the comma-separated list of cross-referenced labels.

```
1624 \newcommand*{\glxtruseeseealsoformat}[1]{%
1625   \glssseeformat[\seealsoformat]{#1}{}%
1626 }
```

`\glxtrseelist` Fully expands argument before passing to `\glsseelist`. (The argument to `\glsseelist` must be a comma-separated list of entry labels.)

```
1627 \newrobustcmd{\glxtrseelist}[1]{%
1628   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1629 }
```

`\seealsoformat` In case this command hasn't been defined. (Should be provided by language packages.)

```
1630 \providecommand{\seealsoformat}{see also}
```

`\glxtrindexseealso` If `\@xdycrossrefhook` is defined, provide a `seealso` crossref class. Otherwise this just does `\glsssee` with `\seealsoformat` as the tag. The hook is only defined if both `xindy` and `glossaries v4.30+` are being used.

```
1631 \ifdef\@xdycrossrefhook
1632 {
```

Add the cross-reference class definition to the hook.

```
1633   \appto\@xdycrossrefhook{%
1634     \write\glswrite{(define-crossref-class \string"seealso\string"
1635       :unverified )}%
1636     \write\glswrite{(markup-crossref-list
1637       :class \string"seealso\string"^^J\space\space\space
1638       :open \string"\string\glxtruseeseealsoformat\glsoopenbrace\string"
1639       :close \string"\glsclosebrace\string")}%
1640   }
```

Append to class list.

```
1641   \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

This essentially works like `\@do@seeglossary` but uses the `seealso` class. This doesn't increment the associated counter.

```
1642   \newrobustcmd*{\glxtrindexseealso}[2]{%
1643     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1644       \@glxtr@recordsee{#1}{#2}%
1645     \fi
1646     \glsoifexists{#1}%
1647     {%
1648       \@@glxtrwrglossmark
1649       \def\@glsoxref{#2}%
1650       \@onelevel@sanitize\@glsoxref
1651       \@glsocheckmkidxchars\@glsoxref
1652       \glso@glossary{\csname glo@#1@type\endcsname}{%
1653         (indexentry
1654           :tkey (\csname glo@#1@index\endcsname)
```

```

1655         :xref (\string"\@gls@xref\string")
1656         :attr \string"seealso\string"
1657     )
1658 }%
1659 }%
1660 }
1661 }
1662 {
    xindy not in use or glossaries version too old to support this.
1663 \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
1664 }

```

The alias key should be set to the label of the synonymous entry. The seealso key essentially behaves like `see=[\seealsoname]{\langle xr-list \rangle}`. Neither of these new keys has the optional tag part allowed with `see`.

If `\gls@set@xr@key` has been defined (glossaries v4.30), use that, otherwise just use `\glsaddstoragekey`.

```

1665 \ifdef\gls@set@xr@key
1666 {

```

We have at least glossaries v4.30. This means the new keys can be governed by the same settings as the `see` key.

```

1667 \define@key{glossentry}{alias}{%
1668 \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1669 }
1670 \define@key{glossentry}{seealso}{%
1671 \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1672 }

```

Add to the key mappings.

```

1673 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1674 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1675 \appto\@newglossaryentryposthook{%
1676 \ifdefvoid\@glo@seealso
1677 {\csxdef{glo@\@glo@label @seealso}{}}%
1678 {%
1679 \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1680 \if@glxtr@autoseeindex
1681 \@glxtr@autoindexcrossrefs
1682 \fi
1683 }%

```

The alias field doesn't trigger the automatic cross-reference indexing performed at the end of the document.

```

1684 \ifdefvoid\@glo@alias

```

```

1685     {\csxdef{glo@\@glo@label @alias}{}}}%
1686     {%
1687     \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1688     }%
1689 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

1690 \newcommand*{\glxtralias}[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1691 \newcommand*{\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

```

Add to the `\@glo@autosee` hook.

```

1692 \appto\@glo@autoseehook{%
1693   \ifdefvoid\@glo@alias
1694   {%
1695     \ifdefvoid\@glo@seealso
1696     }%
1697     {%
1698       \edef\@do@glsssee{\noexpand\glxtrindexseealso
1699         {\@glo@label}{\@glo@seealso}}%
1700       \@do@glsssee
1701     }%
1702   }%
1703   {%

```

Add cross-reference if see key hasn't been used.

```

1704   \ifdefvoid\@glo@see
1705   {%
1706     \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1707     \@do@glsssee
1708   }%
1709   }%
1710 }%
1711 }%
1712 }
1713 {

```

We have an older version of glossaries, so just use `\glsaddstoragekey`.

`\glxtralias`

```

1714 \glsaddstoragekey*{alias}{\glxtralias}

```

`trseealsolabels`

```

1715 \glsaddstoragekey*{seealso}{\glxtrseealsolabels}

```

If `\gls@set@xr@key` isn't defined, then `\@glo@autosee` won't be either, so use the post entry definition hook.

ryentryposthook Append to the hook to check for the alias and seealso keys.

```

1716 \appto\@newglossaryentryposthook{%
1717   \ifcsvoid{glo@\@glo@label @alias}%
1718   {%
1719     \ifcsvoid{glo@\@glo@label @seealso}%
1720     {}%
1721     {%
1722       \edef\@do@glsee{\noexpand\glstrindexseealso
1723         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1724       \@do@glsee
1725     }%
1726   }%
1727   {%

```

Add cross-reference if see key hasn't been used.

```

1728   \ifdefvoid\@glo@see
1729   {%
1730     \edef\@do@glsee{\noexpand\glsee
1731       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1732     \@do@glsee
1733   }%
1734   {}%
1735   }%
1736 }
1737 }

```

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

```

1738 \AtEndDocument{\if@glstrindexcrossrefs\glstraddallcrossrefs\fi}

```

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

```

1739 \newcommand*\@glstraddallcrossrefs{%
1740   \forallglossaries{\@glo@type}%
1741   {%
1742     \forallglsentries[\@glo@type]{\@glo@label}%
1743     {%
1744       \ifglused{\@glo@label}%
1745       {\expandafter\@glstr@addunusedxrefs\expandafter{\@glo@label}}{}%
1746     }%
1747   }%
1748 }

```

@addunusedxrefs If the given entry has a see or seealso field add all unused cross-references. (The alias field isn't checked.)

```

1749 \newcommand*\@glstr@addunusedxrefs[1]{%
1750   \letcs{\@glo@see}{glo\@glsetoklabel{#1}@see}%
1751   \ifdefvoid\@glo@see
1752   {%

```

```

1753 {%
1754   \expandafter\glsxtr@addunused\@glo@see\@end\glsxtr@addunused
1755 }%
1756 \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
1757 \ifdefvoid\@glo@see
1758 {}%
1759 {%
1760   \expandafter\glsxtr@addunused\@glo@see\@end\glsxtr@addunused
1761 }%
1762 }

```

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

```

1763 \newcommand*{\glsxtr@addunused}[1][]{%
1764   \@glsxtr@addunused
1765 }

```

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

```

1766 \def\@glsxtr@addunused#1\@end\glsxtr@addunused{%
1767   \@for\@glsxtr@label:=#1\do
1768   {%
1769     \ifglsused{\@glsxtr@label}{}%
1770     {%
1771       \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1772       \glsunset{\@glsxtr@label}%
1773       \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1774     }%
1775   }%
1776 }

```

`\glsxtrunusedformat`

```

1777 \newcommand*{\glsxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

`\gls@begindocdefs` This command was only introduced to glossaries v4.37, so it may not be defined. If it has been defined, redefine it to check `\@glsxtr@docdefval` so that it only inputs the `.glsdefs` file if `docdef=true`.

```

1778 \ifdef\gls@begindocdefs
1779 {%
1780   \renewcommand*{\gls@begindocdefs}{%
1781     \ifnum\@glsxtr@docdefval=1\relax
1782       \@gls@enablesavenonumberlist
1783       \edef\@gls@restoreat{%
1784         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1785       \makeatletter
1786       \InputIfFileExists{\jobname.glsdefs}{}{}%
1787       \@gls@restoreat
1788       \undef\@gls@restoreat

```

```

1789     \gls@defdocnewglossaryentry
1790   \else
1791     \ifnum\@glsxtr@docdefval=3\relax

```

The docdef=atom package option has been set. Create the .glsdefs file for the autocomplete support but don't read it.

```

1792     \@gls@enablesavenonumberlist
1793     \let\gls@checkseeallowed\relax
1794     \let\newglossaryentry\new@atom@glossaryentry
1795     \global\newwrite\@gls@deffile
1796     \immediate\openout\@gls@deffile=\jobname.glsdefs

```

Write all currently defined entries.

```

1797     \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
1798   \fi
1799 \fi
1800 }
1801 }
1802 {%
1803   \ifnum\@glsxtr@docdefval=3\relax
1804     \PackageError{glossaries-extra}{Package option
1805       'docdef=\@glsxtr@docdefsetting' requires at least version 4.37
1806       of the base glossaries.sty package}{}
1807   \fi
1808 }

```

m@glossaryentry

```

1809 \newrobustcmd{\new@atom@glossaryentry}[2]{%
1810   \gls@defglossaryentry{#1}{#2}%
1811   \@gls@writedef{#1}%
1812 }

```

noidxglossaries Modify \makenoidxglossaries so that it automatically sets docdef=false (unless the restricted setting is on) and disables the docdef key. This command isn't allowed with the record option.

```

1813 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1814 \renewcommand{\makenoidxglossaries}{%
1815   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1816   {%
1817     \glsxtr@orgmakenoidxglossaries

```

Add marker to \@do@seeglossary but don't increment associated counter.

```

1818   \renewcommand{\@do@seeglossary}[2]{%
1819     \@glsxtrwrglossmark
1820     \edef\@gls@label{\glsdetoklabel{##1}}%
1821     \protected@write\@auxout{}{%
1822       \string\@gls@reference
1823       {\curname glo@\@gls@label @type\endcsname}%
1824       {\@gls@label}%
1825     }%

```

```

1826         \string\glsseeformat##2{%
1827     }%
1828 }%
1829 }%

```

Check for docdefs=restricted:

```

1830     \if@glstrdocdefrestricted

```

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

```

1831     \renewcommand*{\@gls@reference}[3]{%
1832         \ifcsundef{\glsref@##1}{\csgdef{\glsref@##1}{}}{}%
1833         \ifinlistcs{##2}{\glsref@##1}%
1834         {}%
1835         {\listcsgadd{\glsref@##1}{##2}}%
1836         \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1837         {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1838         {}%
1839         \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1840     }%
1841     \else

```

Disable document definitions.

```

1842         \@glstrdocdeffalse
1843     \fi
1844     \disable@keys{glossaries-extra}{docdef}%
1845 }%
1846 {%
1847     \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1848         not permitted\MessageBreak
1849         with record=\@glstr@record@setting\space package option}%
1850     {You may only use \string\makenoidxglossaries\space with the
1851         record=off option}%
1852 }%
1853 }

```

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

```

1854 \renewcommand*{\gls@defdocnewglossaryentry}{%
1855     \ifcase\@glstr@docdefval
1856         docdef=false:
1857         \renewcommand*{\newglossaryentry}[2]{%
1858             \PackageError{glossaries-extra}{Glossary entries must
1859             be \MessageBreak defined in the preamble with \MessageBreak
1860             package option 'docdef=false'\MessageBreak(consider using
1861             'docdef=restricted')}{Move your glossary definitions to
1862             the preamble. You can also put them in a \MessageBreak separate file
1863             and load them with \string\loadglsentries.}%
1864             }%
1865         \or

```

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

```
1865 \let\gls@checkseeallowed\relax
1866 \let\newglossaryentry\new@glossaryentry
1867 \else
```

Restricted mode just needs to allow the see value.

```
1868 \let\gls@checkseeallowed\relax
1869 \fi
1870 }%
```

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`

```
1871 \newcommand*{\GlsXtrEnableOnTheFly}{%
1872 \ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1873 }
```

`\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.)

```
1874 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1875 \renewcommand*{\glsdetoklabel}[1]{%
1876 \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1877 {%
1878 \expandafter\detokenize\expandafter{##1}%
1879 }%
1880 {\detokenize{##1}}}%
1881 }%
1882 \@GlsXtrEnableOnTheFly
1883 }
1884 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1885 \expandafter\if\glsbackslash#1%
1886 #3%
1887 \else
1888 #4%
1889 \fi
1890 }
```

`\sXtrstarflywarn`

```
1891 \newcommand*{\glsxtrstarflywarn}{%
1892 \GlossariesExtraWarning{Experimental starred version of
1893 \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1894 read the warnings in the glossaries-extra user manual)}}%
1895 }
```

rEnableOnTheFly

```
1896 \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

```
1897 \newcommand*{\glsxtrcat}{general}
```

\glsxtr

```
1898 \newcommand*{\glsxtr}[1] [] {%
1899 \def\glsxtr@keylist{##1}%
1900 \@glsxtr
1901 }
```

\@glsxtr

```
1902 \newcommand*{\@glsxtr}[2] [] {%
1903 \ifglsentryexists{##2}%
1904 {%
1905 \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
1906 }%
1907 {%
1908 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1909 description={\nopostdesc},##1}%
1910 }%
1911 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1912 }
```

\Glsxtr

```
1913 \newcommand*{\Glsxtr}[1] [] {%
1914 \def\glsxtr@keylist{##1}%
1915 \@Glsxtr
1916 }
```

\@Glsxtr

```
1917 \newcommand*{\@Glsxtr}[2] [] {%
1918 \ifglsentryexists{##2}%
1919 {%
1920 \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
1921 }%
1922 {%
1923 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1924 description={\nopostdesc},##1}%
1925 }%
1926 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
1927 }
```

\glsxtrpl

```
1928 \newcommand*\glsxtrpl[1][]{%
1929 \def\glsxtr@keylist{##1}%
1930 \@glsxtrpl
1931 }
```

\@glsxtrpl

```
1932 \newcommand*\@glsxtrpl[2][]{%
1933 \ifglsentryexists{##2}%
1934 {%
1935 \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1936 }%
1937 {%
1938 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1939 description={\nopostdesc},##1}%
1940 }%
1941 \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
1942 }
```

\Glsxtrpl

```
1943 \newcommand*\Glsxtrpl[1][]{%
1944 \def\glsxtr@keylist{##1}%
1945 \@Glsxtrpl
1946 }
```

\@Glsxtrpl

```
1947 \newcommand*\@Glsxtrpl[2][]{%
1948 \ifglsentryexists{##2}
1949 {%
1950 \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1951 }%
1952 {%
1953 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1954 description={\nopostdesc},##1}%
1955 }%
1956 \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
1957 }
```

\GlsXtrWarning

```
1958 \newcommand*\GlsXtrWarning[2]{%
1959 \def\@glsxtr@optlist{##1}%
1960 \@onelevel@sanitize\@glsxtr@optlist
1961 \GlossariesExtraWarning{The options ‘\@glsxtr@optlist’ have
1962 been ignored for entry ‘##2’ as it has already been defined}%
1963 }
```

Disable commands after the glossary:

```
1964 \renewcommand\@printglossary[2]{%
```

```

1965 \def\@glxtr@printglossopts{##1}%
1966 \@glxtr@orgprintglossary{##1}{##2}%
1967 \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
1968 \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1969 \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1970 \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1971 }

```

abledflycommand

```

1972 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1973 \PackageError{glossaries-extra}%
1974 {\string##1\space can't be used after any of the \MessageBreak
1975 glossaries have been displayed}%
1976 {The on-the-fly commands enabled by
1977 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1978 before the glossaries. If you want to use any entries \MessageBreak
1979 after any of the glossaries, you must use the standard \MessageBreak
1980 method of first defining the entry and then using the \MessageBreak
1981 entry with commands like \string\glx}%
1982 \@glxtr@disabledflycommand
1983 }%
1984 \newcommand*{\@glxtr@disabledflycommand}[2][\@glxtr@disabledflycommand]{##2}

```

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

```

1985 \let\GlsXtrEnableOnTheFly\relax
1986 }
1987 \@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.

```

1988 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

Modify \setglossarystyle to set \@glxtr@current@style.

etglossarystyle

```

1989 \renewcommand*{\setglossarystyle}[1]{%
1990 \ifcsundef{@glstyle@#1}%
1991 {%
1992 \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1993 }%
1994 {%
1995 \csname @glstyle@#1\endcsname

```

Only set the current style if it exists.

```

1996 \protected@edef\@glxtr@current@style{#1}%
1997 }%

```

```

1998 \ifx\@glossary@default@style\relax
1999 \protected@edef\@glossary@default@style{#1}%
2000 \fi
2001 }

```

In case we have an old version of glossaries:

```

2002 \ifdef\@glossary@default@style
2003 {}
2004 {%
2005 \let\@glossary@default@style\relax
2006 }

```

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

```

2007 \ifdef\glslistdottedwidth
2008 {%
2009 \ifdim\glslistdottedwidth=.5\hsize
2010 \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
2011 \AtBeginDocument{%
2012 \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
2013 \setlength{\glslistdottedwidth}{.5\columnwidth}%
2014 \fi
2015 }%
2016 \fi
2017 }
2018 {}%

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

2019 \ifdef\glsdescwidth
2020 {%
2021 \ifdim\glsdescwidth=.6\hsize
2022 \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
2023 \AtBeginDocument{%
2024 \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
2025 \setlength{\glsdescwidth}{.6\columnwidth}%
2026 \fi
2027 }%
2028 \fi
2029 }
2030 {}%

```

and for `\glspagelistwidth`:

`lspagelistwidth`

```

2031 \ifdef\glspagelistwidth
2032 {%
2033 \ifdim\glspagelistwidth=.1\hsize
2034 \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}

```

```

2035 \AtBeginDocument{%
2036 \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
2037 \setlength{\glspagelistwidth}{.1\columnwidth}%
2038 \fi
2039 }%
2040 \fi
2041 }
2042 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```

2043 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
2044 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
2045 \glsnonumberlistfalse
2046 \renewcommand*{\glossaryentrynumbers}[1]{%
2047 \ifglentryexists{\glscurrententrylabel}%
2048 {%
2049 \@glxtrpreloctag
2050 \GlsXtrFormatLocationList{#1}%
2051 \@glxtrpostloctag
2052 \gls@save@numberlist{#1}%
2053 }{}%
2054 }%
2055 \else
2056 \glsnonumberlisttrue
2057 \renewcommand*{\glossaryentrynumbers}[1]{%
2058 \ifglentryexists{\glscurrententrylabel}%
2059 {%
2060 \gls@save@numberlist{#1}%
2061 }{}%
2062 }%
2063 \fi

```

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

```

2064 \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

```

2065 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
2066 \let\@glxtrpreloctag\@glxtrpreloctag
2067 \let\@glxtrpostloctag\@glxtrpostloctag
2068 \renewcommand*{\@glxtr@pagetag}{#1}%
2069 \renewcommand*{\@glxtr@pagetag}{#2}%
2070 \renewcommand*{\@glxtr@savepreloctag}[2]{%

```

```

2071 \csgdef{@glxtr@preloctag@##1}{##2}%
2072 }%
2073 \renewcommand*{\@glxtr@doloctag}{%
2074 \ifcsundef{@glxtr@preloctag@glscurrententrylabel}%
2075 {%
2076 \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
2077 Rerun required}%
2078 }%
2079 {%
2080 \csuse{@glxtr@preloctag@glscurrententrylabel}%
2081 }%
2082 }%
2083 }
2084 \@onlypreamble\GlsXtrEnablePreLocationTag

```

glxtrpreloctag

```

2085 \newcommand*{\@glxtrpreloctag}{%
2086 \let\@glxtr@org@delimN\delimN
2087 \let\@glxtr@org@delimR\delimR
2088 \let\@glxtr@org@glsignore\glsignore
\gdef is required as the delimiters may occur inside a scope.
2089 \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2090 \renewcommand*{\delimN}{%
2091 \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2092 \@glxtr@org@delimN}%
2093 \renewcommand*{\delimR}{%
2094 \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2095 \@glxtr@org@delimR}%
2096 \renewcommand*{\glsignore}[1]{%
2097 \gdef\@glxtr@thisloctag{\relax}%
2098 \@glxtr@org@glsignore{##1}}%
2099 \@glxtr@doloctag
2100 }

```

glxtrpreloctag

```

2101 \newcommand*{\@glxtrpreloctag}{%

```

@glxtr@pagetag

```

2102 \newcommand*{\@glxtr@pagetag}{%

```

glxtr@pagetag

```

2103 \newcommand*{\@glxtr@pagetag}{%

```

lsxtrpostloctag

```

2104 \newcommand*{\@glxtrpostloctag}{%
2105 \let\delimN\@glxtr@org@delimN
2106 \let\delimR\@glxtr@org@delimR
2107 \let\glsignore\@glxtr@org@glsignore

```

```

2108 \protected@write\@auxout{}\%
2109 {\string\@glxtr@savepreloctag{\glscurrententrylabel}\@glxtr@thisloctag}}%
2110 }

```

lsxtrpostloctag

```

2111 \newcommand*\@glxtrpostloctag{}\%

```

lsxtr@preloctag

```

2112 \newcommand*\@glxtr@savepreloctag}[2]\%
2113 \protected@write\@auxout{}\%
2114 \string\providecommand\string\@glxtr@savepreloctag[2]\%

```

glxtr@doloctag

```

2115 \newcommand*\@glxtr@doloctag{}\%

```

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

```

2116 \renewcommand*\KV@printgloss@nonumberlist}[1]\%
2117 \XKV@plfalse
2118 \XKV@sttrue
2119 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
2120 {\%
2121 \csname glsnonumberlist\XKV@resa\endcsname
2122 \ifglsnonumberlist
2123 \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
2124 \else
2125 \def\glossaryentrynumbers##1{\%
2126 \@glxtrpreloctag
2127 \GlsXtrFormatLocationList{##1}%
2128 \@glxtrpostloctag
2129 \gls@save@numberlist{##1}}%
2130 \fi
2131 }%
2132 }

```

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.

```

2133 \renewcommand*\glsentryfmt{\%
2134 \ifglshasshort{\glslabel}\glssetabbrvfmt{\glscategory{\glslabel}}}%
2135 \glsifregular{\glslabel}%
2136 {\glxtrregularfont{\glsentryfmt}}%

```

```

2137  {%
2138    \ifglshasshort{\glslabel}%
2139    {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}}%
2140    {\glsxtrregularfont{\glsgenentryfmt}}}%
2141  }%
2142 }

```

sxtrregularfont Font used for regular entries.

```
2143 \newcommand*{\glsxtrregularfont}[1]{#1}
```

bbreviationfont Font used for abbreviation entries.

```
2144 \newcommand*{\glsxtrabbreviationfont}[1]{#1}
```

Commands like `\glusifplural` 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 `\glxtrifwasfirstuse` etc to allow the postlink hook to work better. This now has an optional argument that sets up the defaults.

```
2145 \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 entry exists (unless indexing has been switched off).

```

2146 \@glxtr@record{#2}{#3}{glslink}%
2147 \glsdoifexists{#3}%
2148 {%

```

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).

```

2149 \let\glxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
2150 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2151 \def\glscustomtext{#4}%
2152 \@glxtr@field@linkdefs
2153 #1%
2154 \@gls@link[#2]{#3}{#4}%
2155 \let\ifKV@glslink@hyper\glxtrorg@ifKV@glslink@hyper
2156 }%
2157 \glspostlinkhook
2158 }

```

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

\@gls@ Save the original definition and redefine.

```

2159 \let\@glxtr@org@gls@\@gls@
2160 \def\@gls@#1#2{%

```

```

2161 \@glstr@record{#1}{#2}{glslink}%
2162 \@glstr@org@glstr@{#1}{#2}%
2163 }%

```

`\@glsp1@` Save the original definition and redefine.

```

2164 \let\@glstr@org@glsp1@\@glsp1@
2165 \def\@glsp1@#1#2{%
2166 \@glstr@record{#1}{#2}{glslink}%
2167 \@glstr@org@glsp1@{#1}{#2}%
2168 }%

```

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

```

2169 \let\@glstr@org@Gls@\@Gls@
2170 \def\@Gls@#1#2{%
2171 \@glstr@record{#1}{#2}{glslink}%
2172 \@glstr@org@Gls@{#1}{#2}%
2173 }%

```

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

```

2174 \let\@glstr@org@Glspl@\@Glspl@
2175 \def\@Glspl@#1#2{%
2176 \@glstr@record{#1}{#2}{glslink}%
2177 \@glstr@org@Glspl@{#1}{#2}%
2178 }%

```

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

```

2179 \let\@glstr@org@GLS@\@GLS@
2180 \def\@GLS@#1#2{%
2181 \@glstr@record{#1}{#2}{glslink}%
2182 \@glstr@org@GLS@{#1}{#2}%
2183 }%

```

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

```

2184 \let\@glstr@org@GLSpl@\@GLSpl@
2185 \def\@GLSpl@#1#2{%
2186 \@glstr@record{#1}{#2}{glslink}%
2187 \@glstr@org@GLSpl@{#1}{#2}%
2188 }%

```

`\@glsdisp` This is redefined to allow the recording on the first run. Can't save and restore `\@glsdisp` since it has an optional argument.

```

2189 \renewcommand*{\@glsdisp}[3][{}]{%
2190 \@glstr@record{#1}{#2}{glslink}%
2191 \glsdoifexists{#2}{%
2192 \let\do@glstr@link@checkfirsthyper\@glstr@link@checkfirsthyper
2193 \let\glstr@sifplural\@secondoftwo
2194 \let\glstr@scapscase\@firstofthree
2195 \def\glstr@customtext{#3}%

```

```

2196 \def\glsinsert{}%
2197 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
2198 \@gls@link[#1]{#2}{\@glo@text}%
2199 \ifKV@glslink@local
2200 \glslocalunset{#2}%
2201 \else
2202 \glsunset{#2}%
2203 \fi
2204 }%
2205 \glspostlinkhook
2206 }

```

`\@gls@@link@` Redefine to include `\@glxtr@record`

```

2207 \renewcommand*{\@gls@@link}[3][{}]{%
2208 \@glxtr@record{#1}{#2}{\glslink}%
2209 \glsdoifexistsordo{#2}%
2210 {%
2211 \let\do@gls@link@checkfirsthyper\relax

```

Post-link hook commands need initialising.

```

2212 \def\glscustomtext{#3}%
2213 \@glxtr@field@linkdefs
2214 \@gls@link[#1]{#2}{#3}%
2215 }%
2216 {%
2217 \glstextformat{#3}%
2218 }%
2219 \glspostlinkhook
2220 }

```

`glxtrinitwrgloss` Set the default if the wrgloss is omitted.

```

2221 \newcommand*{\glxtrinitwrgloss}{%
2222 \glsifattribute{\glslabel}{wrgloss}{after}%
2223 {%
2224 \glxtrinitwrglossbeforefalse
2225 }%
2226 {%
2227 \glxtrinitwrglossbeforetrue
2228 }%
2229 }

```

`gltrwrglossbefore` Conditional to determine if the indexing should be done before the link text.

```

2230 \newif\ifglxtrinitwrglossbefore
2231 \glxtrinitwrglossbeforetrue

```

Define a wrgloss key to determine whether to write the glossary information before or after the link text.

```

2232 \define@choicekey{glslink}{wrgloss}%
2233 [\@glxtr@wrglossval\@glxtr@wrglossnr]%

```

```

2234 {before,after}%
2235 {%
2236   \ifcase\@glsxtr@wrglossnr\relax
2237     \glsxtrinitwrglossbeforetrue
2238   \or
2239     \glsxtrinitwrglossbeforefalse
2240   \fi
2241 }

2242 \define@key{glslink}{thevalue}{\def\@glsxtr@thevalue{#1}}

2243 \define@key{glslink}{theHvalue}{\def\@glsxtr@theHvalue{#1}}

tr@hyperoutside  Define a hyperoutside key to determine whether \hyperlink should be outside \glstextformat.
2244 \define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
2245 \glsxtr@hyperoutsidetrue

ocal@textformat  Provide a key to locally change the text format.
2246 \define@key{glslink}{textformat}{%
2247   \ifcsdef{#1}
2248   {%
2249     \letcs{\@glsxtr@local@textformat}{#1}%
2250   }%
2251   {%
2252     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2253   }%
2254 }

2255 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

nithyperoutside  Set the default if the hyperoutside is omitted.
2256 \newcommand*{\glsxtrinithyperoutside}{%
2257   \glsifattribute{\glslabel}{hyperoutside}{false}%
2258   {%
2259     \glsxtr@hyperoutsidetrue
2260   }%
2261   {%
2262     \glsxtr@hyperoutsidetrue
2263   }%
2264 }

r@inc@linkcount  Does nothing by default.
2265 \newcommand*{\glsxtr@inc@linkcount}{}

slinkpresetkeys  User hook performed immediately before options are set. Does nothing by default.
2266 \newcommand*{\glslinkpresetkeys}{}

```

`\GlsXtrExpandedFmt` Helper command that (protected) fully expands second argument and then applies it to the first, which must be a command that takes a single argument.

```
2267 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2268   \protected@edef\@glxstr@tmp{#2}%
2269   \expandafter#1\expandafter{\@glxstr@tmp}%
2270 }
```

`\@glxstr@use@equation@counter` If in a numbered equation, change the counter to equation. This can be overridden by explicitly setting the counter in the optional argument of commands like `\gls` and `\glslink`.

```
2271 \newcommand*{\@glxstr@use@equation@counter}{%
2272   \@glxstr@ifnum@mmode{\def\@glscounter{equation}}{}%
2273 }
```

`\@glxstr@do@autoadd` If `\GlsXtrAutoAddOnFormat` is used, this will automatically use `\glsadd`. It's therefore only used with `\@glsc@link` not with `\glsadd` otherwise it could trigger an infinite loop. The argument indicates the key family (`glslink` or `glossadd`).

```
2274 \newcommand*{\@glxstr@do@autoadd}[1]{}
```

`\GlsXtrAutoAddOnFormat`

`\GlsXtrAutoAddOnFormat[⟨label⟩]{⟨format list⟩}{⟨glsadd options⟩}`

If an entry is indexed with the format set to one identified in the comma-separated list, then automatically index it using `\glsadd` with the given options, which may override the current options. Scoping is needed to prevent leakage.

```
2275 \newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
2276   \renewcommand*{\@glxstr@do@autoadd}[1]{%
2277     \begingroup
2278       \protected@edef\@glxstr@do@autoadd{%
2279         \noexpand\ifstrequal{##1}{glslink}%
2280         {%
2281           \noexpand\DTLifinlist{\@glscnumberformat}{#2}{\noexpand\glsadd[format={\@glscnumberfor
2282           }%
2283           }%
2284         }%
2285       \@glxstr@do@autoadd
2286     \endgroup
2287   }%
2288 }
```

`\@glsc@link` Redefine to allow the indexing to be placed after the link text. By default this is done before the link text to prevent problems that can occur from the `whatsit`, but there may be times when the user would like the indexing done afterwards even though it causes a `whatsit`.

```
2289 \def\@glsc@link[#1]#2#3{%
2290   \leavevmode
2291   \edef\glslabel{\glsdetoklabel{#2}}%
2292   \def\@glsc@link@opts{#1}%
```

```

2293 \let\@gls@link@label\glslabel
2294 \let\@glsnumberformat\@glsxtr@defaultnumberformat
2295 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2296 \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
2297 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

  Save current value of \glslinkprefix:
2298 \let\@glsxtr@org@glslinkprefix\glslinkprefix

  Initialise \@glsxtr@local@textformat
2299 \let\@glsxtr@local@textformat\relax

  Initialise thevalue and theHvalue (v1.19).
2300 \def\@glsxtr@thevalue{}%
2301 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%

  Initialise when indexing should occur (new to v1.14).
2302 \glsxtrinitwrgloss

  Initialise whether \hyperlink should be outside \gls@textformat (new to v1.21).
2303 \glsxtrinithyperoutside

  Note that the default link options may override \glsxtrinitwrgloss.
2304 \@gls@setdefault@glslink@opts

  Increment link counter if enabled (new to v1.26).
2305 \glsxtr@inc@linkcount

  Check if the equations option has been set (new to v1.37).
2306 \if@glsxtr@equations
2307   \@glsxtr@use@equation@counter
2308 \fi

  As the original definition.
2309 \do@gl:disablehyperinlist
2310 \do@gls@link@checkfirsthyper

  User hook before options are set (new to v1.26):
2311 \glslinkpresetkeys

  Set options.
2312 \setkeys{glslink}{#1}%

  Perform auto add if set (new to v1.37)
2313 \glsxtr@do@autoadd{glslink}%

  User hook after options are set:
2314 \glslinkpostsetkeys

  Check thevalue and theHvalue before saving (v1.19).
2315 \ifdefempty{\@glsxtr@thevalue}%
2316 {%
2317   \@gls@saveentrycounter
2318 }%
2319 {%

```

```

2320 \let\theglsentrycounter\@glstr@thevalue
2321 \def\theHglentrycounter{\@glstr@theHvalue}%
2322 }%
2323 \@gls@setsort{\glslabel}%

```

Check if the textformat key has been used.

```

2324 \ifx\@glstr@local@textformat\relax

```

Check textformat attribute (new to v1.21).

```

2325 \gls@hasattribute{\glslabel}{textformat}%
2326 {%
2327 \edef\@glstr@attrval{\gls@getattribute{\glslabel}{textformat}}%
2328 \ifcsdef{\@glstr@attrval}%
2329 {%
2330 \letcs{\@glstr@textformat}{\@glstr@attrval}%
2331 }%
2332 {%
2333 \GlossariesExtraWarning{Unknown control sequence name
2334 '\@glstr@attrval' supplied in textformat attribute
2335 for entry '\glslabel'. Reverting to default \string\glstextformat}%
2336 \let\@glstr@textformat\glstextformat
2337 }%
2338 }%
2339 {%
2340 \let\@glstr@textformat\glstextformat
2341 }%
2342 \else
2343 \let\@glstr@textformat\@glstr@local@textformat
2344 \fi

```

Do write if it should occur before the link text:

```

2345 \ifglstr@initwrglossbefore
2346 \@do@wrglossary{#2}%
2347 \fi

```

Do the link text:

```

2348 \ifKV@glslink@hyper
2349 \ifglstr@hyperoutside
2350 \@glslink{\glolinkprefix\glslabel}{\@glstr@textformat{#3}}%
2351 \else
2352 \@glstr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2353 \fi
2354 \else
2355 \ifglstr@hyperoutside
2356 \glsdonohyperlink{\glolinkprefix\glslabel}{\@glstr@textformat{#3}}%
2357 \else
2358 \@glstr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
2359 \fi
2360 \fi

```

Do write if it should occur after the link text:

```

2361 \ifglxtrinitwrglossbefore
2362 \else
2363   \@do@wrglossary{#2}%
2364 \fi

```

Restore original value of \glolinkprefix:

```
2365 \let\glolinkprefix\@glxtr@org@glolinkprefix
```

As the original definition:

```

2366 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2367 }

```

```
2368 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}
```

```
2369 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}
```

lsaddpresetkeys

```
2370 \newcommand*{\gladdpresetkeys}{}
```

saddpostsetkeys

```
2371 \newcommand*{\gladdpostsetkeys}{}
```

\gladd Redefine to include \@glxtr@record and suppress in headings

```

2372 \renewrobustcmd*{\gladd}[2][\]{%
2373   \glxtrifinmark
2374   {}}%
2375   {%
2376     \@gl@adjustmode
2377     \begingroup
2378     \@glxtr@record{#1}{#2}{glossadd}%
2379     \glsoifexists{#2}%
2380     {%
2381       \let\@gl@numberformat\@glxtr@defaultnumberformat
2382       \edef\@gl@counter{\csname glo@gl@detoklabel{#2}@counter\endcsname}%
2383       \def\@glxtr@thevalue{}%
2384       \def\@glxtr@theHvalue{\@glxtr@thevalue}%

```

Implement any default settings (before options are set)

```

2385     \gladdpresetkeys
2386     \setkeys{glossadd}{#1}%

```

Implement any default settings (after options are set)

```

2387     \gladdpostsetkeys
2388     \ifdefempty{\@glxtr@thevalue}%
2389     {%
2390       \@gl@saveentrycounter
2391     }%
2392     {%
2393       \let\thegl@entrycounter\@glxtr@thevalue
2394       \def\theHgl@entrycounter{\@glxtr@theHvalue}%
2395     }%

```

Define sort key if necessary (in case of sort=use):

```
2396 \gls@setsort{#2}%
```

Ensure that indexing occurs (since that's the point of \glsadd). If indexing has been switched off by default, don't want the setting to affect \glsadd. The ignored format \glsignore can be used for selection without location, but the indexing still needs to be performed.

```
2397 \KV@glslink@noindexfalse
```

```
2398 \@@do@wrglossary{#2}%
```

```
2399 }%
```

```
2400 \endgroup
```

```
2401 }%
```

```
2402 }
```

\glsaddeach Performs \glsadd for each entry listed in the mandatory argument.

```
2403 \newrobustcmd{\glsaddeach}[2] [] {%
```

```
2404 \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
```

```
2405 }
```

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

```
2406 \newcommand*{\@glsxtr@field@linkdefs}{%
```

```
2407 \let\glsxtrifwasfirstuse\@secondoftwo
```

```
2408 \let\glsifplural\@secondoftwo
```

```
2409 \let\glscapscase\@firstofthree
```

```
2410 \let\glsinsert\@empty
```

```
2411 }
```

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

assignfieldfont

```
2412 \newcommand*{\glsxtrassignfieldfont}[1] {%
```

```
2413 \ifglstryexists{#1}%
```

```
2414 {%
```

```
2415 \ifglshasshort{#1}%
```

```
2416 {%
```

```
2417 \glssetabbrvfmt{\glscategory{#1}}%
```

```
2418 \glsifregular{#1}%
```

```
2419 {\let\@gls@field@font\glsxtrregularfont}%
```

```
2420 {\let\@gls@field@font\@firstofone}%
```

```
2421 }%
```

```
2422 {%
```

```
2423 \glsifnotregular{#1}%
```

```
2424 {\let\@gls@field@font\@firstofone}%
```

```
2425 {\let\@gls@field@font\glsxtrregularfont}%
```

```
2426 }%
```

```
2427 }%
```

```
2428 {%
```

```
2429 \let\@gls@field@font\@gobble
```

```
2430 }%
```

```
2431 }
```

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

```
2432 \def\@glstext@#1#2[#3]{%
2433   \glstrassignfieldfont{#2}%
2434   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2435 }
```

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

```
2436 \def\@GLStext@#1#2[#3]{%
2437   \glstrassignfieldfont{#2}%
2438   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2439   {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2440 }
```

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

```
2441 \def\@Glstext@#1#2[#3]{%
2442   \glstrassignfieldfont{#2}%
2443   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2444   {\@gls@field@font{\Glsaccesstext{#2}#3}}%
2445 }
```

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`

```
2446 \newcommand*{\glstrchecknohyperfirst}[1]{%
2447   \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2448 }
```

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

```
2449 \def\@glsfirst@#1#2[#3]{%
2450   \glstrassignfieldfont{#2}%
2451   \@gls@field@link
2452   [\let\glstrifwasfirstuse\@firstoftwo
2453   \glstrchecknohyperfirst{#2}%
2454   ]{#1}{#2}%
2455   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2456 }
```

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

```
2457 \def\@Glsfirst@#1#2[#3]{%
2458   \glstrassignfieldfont{#2}%
2459   \@gls@field@link
2460   [\let\glstrifwasfirstuse\@firstoftwo
2461   \let\glscapscase\@secondofthree
2462   \glstrchecknohyperfirst{#2}%
2463   ]{#1}{#2}%
2464 }
```

```

2463 ]%
2464 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2465 }

```

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

```

2466 \def\@GLSfirst@#1#2[#3]{%
2467   \glstrassignfieldfont{#2}%
      Ensure that \@GLSfirst honours the nohyperfirst attribute.
2468   \@gls@field@link
2469   [\let\glstrifwasfirstuse\@firstoftwo
2470    \let\glscapscase\@thirdofthree
2471    \glstrchecknohyperfirst{#2}%
2472   ]%
2473   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2474 }

```

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

```

2475 \def\@glsplural@#1#2[#3]{%
2476   \glstrassignfieldfont{#2}%
2477   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2478   {\@gls@field@font{\glsaccessplural{#2}#3}}%
2479 }

```

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

```

2480 \def\@Glsplural@#1#2[#3]{%
2481   \glstrassignfieldfont{#2}%
2482   \@gls@field@link
2483   [\let\glsifplural\@firstoftwo
2484    \let\glscapscase\@secondofthree
2485   ]%
2486   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2487 }

```

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

```

2488 \def\@GLSplural@#1#2[#3]{%
2489   \glstrassignfieldfont{#2}%
2490   \@gls@field@link
2491   [\let\glsifplural\@firstoftwo
2492    \let\glscapscase\@thirdofthree
2493   ]%
2494   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2495 }

```

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

```

2496 \def\glsfirstplural@#1#2[#3]{%
2497   \glstrassignfieldfont{#2}%

```

Ensure that \glsfirstplural honours the nohyperfirst attribute.

```
2498 \@gls@field@link
2499 [\let\glstrifwasfirstuse\@firstoftwo
2500 \let\glsifplural\@firstoftwo
2501 \glstrchecknohyperfirst{#2}%
2502 ]%
2503 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}{#3}}}%
2504 }
```

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

```
2505 \def\@Glsfirstplural@#1#2[#3]{%
2506 \glstrassignfieldfont{#2}%
2507 \@gls@field@link
2508 [\let\glstrifwasfirstuse\@firstoftwo
2509 \let\glsifplural\@firstoftwo
2510 \let\glscapscase\@secondofthree
2511 \glstrchecknohyperfirst{#2}%
2512 ]%
2513 {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}{#3}}}%
2514 }
```

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

```
2515 \def\@GLSfirstplural@#1#2[#3]{%
2516 \glstrassignfieldfont{#2}%
2517 \@gls@field@link
2518 [\let\glstrifwasfirstuse\@firstoftwo
2519 \let\glsifplural\@firstoftwo
2520 \let\glscapscase\@thirdofthree
2521 \glstrchecknohyperfirst{#2}%
2522 ]%
2523 {#1}{#2}%
2524 {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2525 }
```

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

```
2526 \def\@glsname@#1#2[#3]{%
2527 \glstrassignfieldfont{#2}%
2528 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}{#3}}}%
2529 }
```

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

```
2530 \def\@Glsname@#1#2[#3]{%
2531 \glstrassignfieldfont{#2}%
2532 \@gls@field@link
2533 [\let\glscapscase\@secondoftwo]{#1}{#2}%
```

```

2534 {\@gls@field@font{\Glsaccessname{#2}#3}}%
2535 }

```

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

```

2536 \def\@GLSname@#1#2[#3]{%
2537 \glstrassignfieldfont{#2}%
2538 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2539 {#1}{#2}%
2540 {\@gls@field@font{\Glsaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2541 }

```

`\@glsdesc@`

```

2542 \def\@glsdesc@#1#2[#3]{%
2543 \glstrassignfieldfont{#2}%
2544 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
2545 }

```

`\@Glsdesc@` First letter uppercase version.

```

2546 \def\@Glsdesc@#1#2[#3]{%
2547 \glstrassignfieldfont{#2}%
2548 \@gls@field@link
2549 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2550 {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
2551 }

```

`\@GLSdesc@` All uppercase version.

```

2552 \def\@GLSdesc@#1#2[#3]{%
2553 \glstrassignfieldfont{#2}%
2554 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2555 {#1}{#2}{\@gls@field@font{\Glsaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2556 }

```

`@glsdescplural@` No case-changing version.

```

2557 \def\@glsdescplural@#1#2[#3]{%
2558 \glstrassignfieldfont{#2}%
2559 \@gls@field@link
2560 [\let\glscapscase\@secondoftwo
2561 \let\glsifplural\@firstoftwo
2562 ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
2563 }

```

`@Glsdescplural@` First letter uppercase version.

```

2564 \def\@Glsdescplural@#1#2[#3]{%
2565 \glstrassignfieldfont{#2}%
2566 \@gls@field@link
2567 [\let\glscapscase\@secondoftwo
2568 \let\glsifplural\@firstoftwo
2569 ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2570 }

```

@GLSdescplural@ All uppercase version.

```
2571 \def\@GLSdesc@#1#2[#3]{%
2572   \glstrassignfieldfont{#2}%
2573   \@gls@field@link
2574   [\let\glscapscase\@thirdoftwo
2575   \let\glsifplural\@firstoftwo
2576   ]%
2577   {#1}{#2}%
2578   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2579 }
```

\@glssymbol@

```
2580 \def\@glssymbol@#1#2[#3]{%
2581   \glstrassignfieldfont{#2}%
2582   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2583 }
```

\@Glsymbol@ First letter uppercase version.

```
2584 \def\@Glsymbol@#1#2[#3]{%
2585   \glstrassignfieldfont{#2}%
2586   \@gls@field@link
2587   [\let\glscapscase\@secondoftwo]%
2588   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2589 }
```

\@GLSsymbol@ All uppercase version.

```
2590 \def\@GLSsymbol@#1#2[#3]{%
2591   \glstrassignfieldfont{#2}%
2592   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2593   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2594 }
```

lssymbolplural@ No case-changing version.

```
2595 \def\@lssymbolplural@#1#2[#3]{%
2596   \glstrassignfieldfont{#2}%
2597   \@gls@field@link
2598   [\let\glscapscase\@secondoftwo
2599   \let\glsifplural\@firstoftwo
2600   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2601 }
```

lssymbolplural@ First letter uppercase version.

```
2602 \def\@lssymbolplural@#1#2[#3]{%
2603   \glstrassignfieldfont{#2}%
2604   \@gls@field@link
2605   [\let\glscapscase\@secondoftwo
2606   \let\glsifplural\@firstoftwo
2607   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2608 }
```

LSSymbolplural@ All uppercase version.

```
2609 \def\@GLSSymbol@#1#2[#3]{%
2610   \glstrassignfieldfont{#2}%
2611   \@gls@field@link
2612   [\let\glscapscase\@thirdoftwo
2613   \let\glsifplural\@firstoftwo
2614   ]%
2615   {#1}{#2}%
2616   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2617 }
```

\@Glsuseri@ First letter uppercase version.

```
2618 \def\@Glsuseri@#1#2[#3]{%
2619   \glstrassignfieldfont{#2}%
2620   \@gls@field@link
2621   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2622   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2623 }
```

\@GLSuseri@ All uppercase version.

```
2624 \def\@GLSuseri@#1#2[#3]{%
2625   \glstrassignfieldfont{#2}%
2626   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2627   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
2628 }
```

\@Glsuserii@ First letter uppercase version.

```
2629 \def\@Glsuserii@#1#2[#3]{%
2630   \glstrassignfieldfont{#2}%
2631   \@gls@field@link
2632   [\let\glscapscase\@secondoftwo]%
2633   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2634 }
```

\@GLSuserii@ All uppercase version.

```
2635 \def\@GLSuserii@#1#2[#3]{%
2636   \glstrassignfieldfont{#2}%
2637   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2638   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}}%
2639 }
```

\@Glsuseriii@ First letter uppercase version.

```
2640 \def\@Glsuseriii@#1#2[#3]{%
2641   \glstrassignfieldfont{#2}%
2642   \@gls@field@link
2643   [\let\glscapscase\@secondoftwo]%
2644   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
2645 }
```

```

\@GLSuseriii@ All uppercase version.
2646 \def\@GLSuseriii@#1#2[#3]{%
2647   \glstrassignfieldfont{#2}%
2648   \@gls@field@link[\let\glscapscale\@thirdoftwo]%
2649   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2650 }

\@GLSuseriv@ First letter uppercase version.
2651 \def\@GLSuseriv@#1#2[#3]{%
2652   \glstrassignfieldfont{#2}%
2653   \@gls@field@link
2654   [\let\glscapscale\@secondoftwo]%
2655   {#1}{#2}{\@gls@field@font{\Glentryuseriv{#2}#3}}}%
2656 }

\@GLSuseriv@ All uppercase version.
2657 \def\@GLSuseriv@#1#2[#3]{%
2658   \glstrassignfieldfont{#2}%
2659   \@gls@field@link[\let\glscapscale\@thirdoftwo]%
2660   {#1}{#2}%
2661   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2662 }

\@GLSuserv@ First letter uppercase version.
2663 \def\@GLSuserv@#1#2[#3]{%
2664   \glstrassignfieldfont{#2}%
2665   \@gls@field@link
2666   [\let\glscapscale\@secondoftwo]%
2667   {#1}{#2}{\@gls@field@font{\Glentryuserv{#2}#3}}}%
2668 }

\@GLSuserv@ All uppercase version.
2669 \def\@GLSuserv@#1#2[#3]{%
2670   \glstrassignfieldfont{#2}%
2671   \@gls@field@link[\let\glscapscale\@thirdoftwo]%
2672   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2673 }

\@GLSuservi@ First letter uppercase version.
2674 \def\@GLSuservi@#1#2[#3]{%
2675   \glstrassignfieldfont{#2}%
2676   \@gls@field@link
2677   [\let\glscapscale\@secondoftwo]%
2678   {#1}{#2}{\@gls@field@font{\Glentryuservi{#2}#3}}}%
2679 }

\@GLSuservi@ All uppercase version.
2680 \def\@GLSuservi@#1#2[#3]{%

```

```

2681 \glstrassignfieldfont{#2}%
2682 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2683 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}{#3}}}%
2684 }

```

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

`\@acrshort` No case change.

```

2685 \def\@acrshort#1#2[#3]{%
2686 \glsdoifexists{#2}%
2687 {%
2688 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2689 \let\glstrifwasfirstuse\@secondoftwo
2690 \let\glsifplural\@secondoftwo
2691 \let\glscapscase\@firstofthree
2692 \let\glsinsert\@empty
2693 \def\glscustomtext{%
2694 \acronymfont{\glsaccessshort{#2}}#3%
2695 }%
2696 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2697 }%
2698 \glspostlinkhook
2699 }

```

`\@Acrshort` First letter uppercase.

```

2700 \def\@Acrshort#1#2[#3]{%
2701 \glsdoifexists{#2}%
2702 {%
2703 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2704 \let\glstrifwasfirstuse\@secondoftwo
2705 \let\glsifplural\@secondoftwo
2706 \let\glscapscase\@secondofthree
2707 \let\glsinsert\@empty
2708 \def\glscustomtext{%
2709 \acronymfont{\Glsaccessshort{#2}}#3%
2710 }%
2711 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2712 }%
2713 \glspostlinkhook
2714 }

```

`\@ACRshort` All uppercase.

```

2715 \def\@ACRshort#1#2[#3]{%
2716 \glsdoifexists{#2}%
2717 {%
2718 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2719 \let\glstrifwasfirstuse\@secondoftwo
2720 \let\glsifplural\@secondoftwo

```

```

2721 \let\glscapscase\@thirdofthree
2722 \let\glsinsert\@empty
2723 \def\glscustomtext{%
2724     \mfirstucMakeUppercase{\acronymfont{\glsaccessshort{#2}}#3}%
2725 }%
2726 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2727 }%
2728 \glspostlinkhook
2729 }

```

\@acrshortpl No case change.

```

2730 \def\@acrshortpl#1#2[#3]{%
2731     \glsdoifexists{#2}%
2732     {%
2733         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2734         \let\glxtrifwasfirstuse\@secondoftwo
2735         \let\glsifplural\@firstoftwo
2736         \let\glscapscase\@firstofthree
2737         \let\glsinsert\@empty
2738         \def\glscustomtext{%
2739             \acronymfont{\glsaccessshortpl{#2}}#3%
2740         }%
2741         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2742     }%
2743     \glspostlinkhook
2744 }

```

\@Acrshortpl First letter uppercase.

```

2745 \def\@Acrshortpl#1#2[#3]{%
2746     \glsdoifexists{#2}%
2747     {%
2748         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2749         \let\glxtrifwasfirstuse\@secondoftwo
2750         \let\glsifplural\@firstoftwo
2751         \let\glscapscase\@secondofthree
2752         \let\glsinsert\@empty
2753         \def\glscustomtext{%
2754             \acronymfont{\Glsaccessshortpl{#2}}#3%
2755         }%
2756         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2757     }%
2758     \glspostlinkhook
2759 }

```

\@ACRshortpl All uppercase.

```

2760 \def\@ACRshortpl#1#2[#3]{%
2761     \glsdoifexists{#2}%
2762     {%
2763         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2764 \let\glxtrifwasfirstuse\@secondoftwo
2765 \let\glsifplural\@firstoftwo
2766 \let\glscapscase\@thirdofthree
2767 \let\glsinsert\@empty
2768 \def\glscustomtext{%
2769     \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
2770 }%
2771 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2772 }%
2773 \glspostlinkhook
2774 }

```

\@acrlong No case change.

```

2775 \def\@acrlong#1#2[#3]{%
2776     \glsdoifexists{#2}%
2777     {%
2778         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2779         \let\glxtrifwasfirstuse\@secondoftwo
2780         \let\glsifplural\@secondoftwo
2781         \let\glscapscase\@firstofthree
2782         \let\glsinsert\@empty
2783         \def\glscustomtext{%
2784             \acronymfont{\glsaccesslong{#2}}#3%
2785         }%
2786         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2787     }%
2788     \glspostlinkhook
2789 }

```

\@Acrlong First letter uppercase.

```

2790 \def\@Acrlong#1#2[#3]{%
2791     \glsdoifexists{#2}%
2792     {%
2793         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2794         \let\glxtrifwasfirstuse\@secondoftwo
2795         \let\glsifplural\@secondoftwo
2796         \let\glscapscase\@secondofthree
2797         \let\glsinsert\@empty
2798         \def\glscustomtext{%
2799             \acronymfont{\Glsaccesslong{#2}}#3%
2800         }%
2801         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2802     }%
2803     \glspostlinkhook
2804 }

```

\@ACRlong All uppercase.

```

2805 \def\@ACRlong#1#2[#3]{%
2806     \glsdoifexists{#2}%

```

```

2807 {%
2808   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2809   \let\glxtrifwasfirstuse\@secondoftwo
2810   \let\gl@ifplural\@secondoftwo
2811   \let\gl@scapscase\@thirdofthree
2812   \let\gl@insert\@empty
2813   \def\glscustomtext{%
2814     \mfirstucMakeUppercase{\acronymfont{\gl@accesslong{#2}}#3}%
2815   }%
2816   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2817 }%
2818 \glspostlinkhook
2819 }

```

\@acrlongpl No case change.

```

2820 \def\@acrlongpl#1#2[#3]{%
2821   \gl@ifexists{#2}%
2822   {%
2823     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2824     \let\glxtrifwasfirstuse\@secondoftwo
2825     \let\gl@ifplural\@firstoftwo
2826     \let\gl@scapscase\@firstofthree
2827     \let\gl@insert\@empty
2828     \def\glscustomtext{%
2829       \acronymfont{\gl@accesslongpl{#2}}#3%
2830     }%
2831     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2832   }%
2833   \glspostlinkhook
2834 }

```

\@Acrlongpl First letter uppercase.

```

2835 \def\@Acrlongpl#1#2[#3]{%
2836   \gl@ifexists{#2}%
2837   {%
2838     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2839     \let\glxtrifwasfirstuse\@secondoftwo
2840     \let\gl@ifplural\@firstoftwo
2841     \let\gl@scapscase\@secondofthree
2842     \let\gl@insert\@empty
2843     \def\glscustomtext{%
2844       \acronymfont{\Gls@accesslongpl{#2}}#3%
2845     }%
2846     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2847   }%
2848   \glspostlinkhook
2849 }

```

\@ACRlongpl All uppercase.

```

2850 \def\@ACRlongpl#1#2[#3]{%
2851   \glsdoifexists{#2}%
2852   {%
2853     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2854     \let\glxtrifwasfirstuse\@secondoftwo
2855     \let\gl@ifplural\@firstoftwo
2856     \let\gl@scapscase\@thirdofthree
2857     \let\gl@insert\@empty
2858     \def\glscustomtext{%
2859       \mfirstucMakeUppercase{\acronymfont{\gl@accesslongpl{#2}}#3}%
2860     }%
2861     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2862   }%
2863   \glspostlinkhook
2864 }

```

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

\@glsaddkey

```

2865 \renewcommand*{\@glsaddkey}[7]{%
2866   \key@ifundefined{glossentry}{#1}%
2867   {%
2868     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2869     \appto\@gl@keymap{, {#1}{#1}}%
2870     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2871     \appto\@newglossaryentryposthook{%
2872       \letcs{\@glo@tmp}{@glo@#1}%
2873       \gl@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2874     }%
2875     \newcommand*{#3}[1]{\@gl@entry@field{##1}{#1}}%
2876     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2877   \ifcsdef{@gl@user@#1@}%
2878   {%
2879     \PackageError{glossaries}%
2880     {Can't define '\string#5' as helper command
2881     '\expandafter\string\csname @gl@user@#1@\endcsname' already
2882     exists}%
2883   }%
2884 }%
2885 {%
2886   \expandafter\newcommand\expandafter*\expandafter
2887   {\csname @gl@user@#1\endcsname}[2] []{%
2888     \new@ifnextchar[%
2889       {\csuse{@gl@user@#1@}{##1}{##2}}%
2890       {\csuse{@gl@user@#1@}{##1}{##2} []}}%
2891   \csdef{@gl@user@#1@}##1##2[##3]{%
2892     \@gl@field@link{##1}{##2}{#3{##2}##3}%

```

```

2893 }%
2894 \newrobustcmd*{#5}{%
2895   \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2896 }%

```

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

```

2897 \ifcsdef{@Gls@user@#1@}%
2898 {%
2899   \PackageError{glossaries}%
2900   {Can't define '\string#6' as helper command
2901     '\expandafter\string\csname @Gls@user@#1\endcsname' already
2902     exists}%
2903 }%
2904 }%
2905 {%
2906   \expandafter\newcommand\expandafter*\expandafter
2907   {\csname @Gls@user@#1\endcsname}[2][{}]{%
2908     \new@ifnextchar[%
2909       {\csuse{@Gls@user@#1@}{##1}{##2}}%
2910       {\csuse{@Gls@user@#1@}{##1}{##2}[{}]}%
2911     \csdef{@Gls@user@#1@}##1##2[##3]{%
2912       \@gls@field@link[\let\glscapscase\@secondofthree]%
2913       {##1}{##2}{#4{##2}##3}%
2914     }%
2915     \newrobustcmd*{#6}{%
2916       \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2917   }%

```

Finally the all caps version (modified):

```

2918 \ifcsdef{@GLS@user@#1@}%
2919 {%
2920   \PackageError{glossaries}%
2921   {Can't define '\string#7' as helper command
2922     '\expandafter\string\csname @GLS@user@#1\endcsname' already
2923     exists}%
2924 }%
2925 }%
2926 {%
2927   \expandafter\newcommand\expandafter*\expandafter
2928   {\csname @GLS@user@#1\endcsname}[2][{}]{%
2929     \new@ifnextchar[%
2930       {\csuse{@GLS@user@#1@}{##1}{##2}}%
2931       {\csuse{@GLS@user@#1@}{##1}{##2}[{}]}%
2932     \csdef{@GLS@user@#1@}##1##2[##3]{%
2933       \@gls@field@link[\let\glscapscase\@thirdofthree]%
2934       {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2935     }%
2936     \newrobustcmd*{#7}{%
2937       \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2938   }%

```

```

2939 }%
2940 {%
2941   \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
2942 }%
2943 }

```

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

```

2944 \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.

```

2945 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2946 \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 to automatically set the value for the commands that change the first use flag. The other commands should set \glsxtrifwasfirstuse to \@secondoftwo (which is done in \@glsxtr@field@linkdefs).

```

2947   \ifglsused{\glslabel}%
2948   {\let\glsxtrifwasfirstuse\@secondoftwo}
2949   {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2950   \edef\glscategorylabel{\glscategory{\glslabel}}%
2951   \ifglsused{\glslabel}%
2952   {%
2953     \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
2954     {\KV@glslink@hyperfalse}{}%
2955   }%
2956   {%
2957     \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
2958     {\KV@glslink@hyperfalse}{}%
2959   }%
2960   \glslinkcheckfirsthyperhook
2961 }

```

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.

```

2962 \ifdef\do@glsdisablehyperinlist
2963 {%
2964   \let\@glsxtr@do@glsdisablehyperinlist\do@glsdisablehyperinlist
2965   \renewcommand*{\do@glsdisablehyperinlist}{%
2966     \@glsxtr@do@glsdisablehyperinlist
2967     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
2968   }
2969 }
2970 {}

```

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

```
2971 \define@boolkey{glslink}{noindex}[true]{}
2972 \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`

```
2973 \ifdef\@gls@setdefault@glslink@opts
2974 {
2975   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2976     \KV@glslink@noindexfalse
2977     \@glxtrsetaliasnoindex
2978   }
2979 }
2980 {
```

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

```
2981   \newcommand*{\@gls@setdefault@glslink@opts}{%
2982     \KV@glslink@noindexfalse
2983     \@glxtrsetaliasnoindex
2984   }
2985   \preto\do@glstdisablehyperinlist{\@gls@setdefault@glslink@opts}
2986 }
```

`setaliasnoindex` Allow user to hook into the alias noindex setting. Default behaviour switches off indexing for aliases. If the record option is on, this will have been defined to do nothing. (bib2gls will deal with records for aliased entries.)

```
2987 \providecommand*{\glxtrsetaliasnoindex}{%
2988   \KV@glslink@noindextrue
2989 }
```

`setaliasnoindex`

```
2990 \newcommand*{\@glxtrsetaliasnoindex}{%
2991   \glxtrifhasfield{alias}{\glslabel}}%
2992 {%
2993   \let\glxtrindexaliased\@glxtrindexaliased
2994   \glxtrsetaliasnoindex
2995   \let\glxtrindexaliased\@no@glxtrindexaliased
2996 }%
2997 {}%
2998 }
```

`xtrindexaliased`

```
2999 \newcommand{\@glxtrindexaliased}{%
3000   \ifKV@glslink@noindex
3001   \else
3002     \begingroup
3003     \let\@glsnumberformat\@glxtr@defaultnumberformat
```

```

3004 \edef\@gls@counter{\csname glo@\glsdetoklabel{\glslabel}@counter\endcsname}%
3005 \glstr@saveentrycounter
3006 @@do@wrglossary{\glstralias{\glslabel}}%
3007 \endgroup
3008 \fi
3009 }

```

trindexaliased

```

3010 \newcommand{\@no@glstrindexaliased}{%
3011 \PackageError{glossaries-extra}{\string\glstrindexaliased\space
3012 not permitted outside definition of \string\glstrsetaliasnoindex}%
3013 }%
3014 }

```

trindexaliased Provide a command to redirect alias indexing, but only allow it to be used within \glstrsetaliasnoindex.

```

3015 \let\glstrindexaliased\@no@glstrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

3016 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
3017 \renewcommand*{\@gls@setdefault@glslink@opts}{%
3018 \setkeys{glslink}{#1}%
3019 \@glstrsetaliasnoindex
3020 }%
3021 }

```

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

```

3022 \newcommand*{\glsxtrifindexing}[2]{%
3023 \ifKV@glslink@noindex #2\else #1\fi
3024 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

3025 \renewcommand*{\glswriteentry}[2]{%
3026 \glsxtrifindexing
3027 {%
3028 \ifglsindexonlyfirst
3029 \ifglsused{#1}
3030 {\glstrdoautoindexname{#1}{dualindex}}%
3031 {#2}%
3032 \else
3033 \glsifattribute{#1}{indexonlyfirst}{true}%
3034 {%
3035 \ifglsused{#1}%
3036 {\glstrdoautoindexname{#1}{dualindex}}%
3037 {#2}%
3038 }%
3039 {#2}%
3040 \fi
3041 }%
3042 {}%

```

3043 }

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

```
3044 \appto\@do@@wrglossary{\@glxtr@do@@wrindex
3045 \glxtrdowrglossaryhook{\@gls@label}}%
3046 }
```

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

Similarly for the “noidx” version:

s@noidxglossary

```
3047 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
3048 \glxtrdowrglossaryhook{\@gls@label}}%
3049 }
```

xtr@do@@wrindex

```
3050 \newcommand*{\@glxtr@do@@wrindex}{%
3051 \glxtrdoautoindexname{\@gls@label}{dualindex}}%
3052 }
```

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.)

```
3053 \newcommand*{\glxtrdowrglossaryhook}[1]{}
```

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

```
3054 \newcommand*{\@gls@alt@hyp@opt}[1]{%
3055 \let\glslinkvar\@firstofthree
3056 \let\@gls@hyp@opt@cs#1\relax
3057 \@ifstar{\s@gls@hyp@opt}%
3058 {\@ifnextchar+%
3059 {\@firstoftwo{\p@gls@hyp@opt}}}%
3060 {%
3061 \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
3062 {\@firstoftwo{\@alt@gls@hyp@opt}}}%
3063 {\#1}%
3064 }%
3065 }%
3066 }
```

alt@gls@hyp@opt User version

```
3067 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
3068 \let\glslinkvar\@firstofthree
3069 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
3070 \newcommand*{\@gls@alt@hyp@opt@char}{{}
```

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

```
3071 \newcommand*{\@gls@alt@hyp@opt@keys}{}
```

rSetAltModifier

```
3072 \newcommand*{\GlsXtrSetAltModifier}[2]{%
3073   \let\@gls@hyp@opt\@gls@alt@hyp@opt
3074   \def\@gls@alt@hyp@opt@char{#1}%
3075   \def\@gls@alt@hyp@opt@keys{#2}%
3076   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
3077   }{%
3078   {%
```

Let **bib2gls** know the modifier.

```
3079   \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@altmodifier}[1]{}}%
3080   \protected@write\@auxout{}{\string\@glsxtr@altmodifier{#1}}%
3081   }%
3082 }
```

org@dohyperlink

```
3083 \let\glsxtr@org@dohyperlink\glsdohyperlink
```

glsnavhyperlink Now that `\glsdohyperlink` (used by `\@glslink`) references `\glslabel` it's necessary to patch `\glsnavhyperlink` to avoid using it (since `\glslabel` won't be defined). This means temporarily redefining `\glsdohyperlink` to its original definition.

This command is provided by glossary-hypernav so it may not exist.

```
3084 \ifdef\glsnavhyperlink
3085 {
3086   \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
3087     \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
3088   {%
3089     \let\glsdohyperlink\glsxtr@org@dohyperlink
3090     \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
3091   }%
3092   }%
3093 }
3094 {}
```

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

```
3095 \renewcommand*{\glsdohyperlink}[2]{%
3096   \glsasattribute{\glslabel}{targeturl}%
```

```

3097 {%
3098   \glshasattribute{\glslabel}{targetname}%
3099   {%
3100     \glshasattribute{\glslabel}{targetcategory}%
3101     {%
3102       \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
3103         {\glsgetattribute{\glslabel}{targetcategory}}}%
3104         {\glsgetattribute{\glslabel}{targetname}}}%
3105         {{\glsxtrprotectlinks#2}}}%
3106     }%
3107   {%
3108     \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
3109       {}%
3110       {\glsgetattribute{\glslabel}{targetname}}}%
3111       {{\glsxtrprotectlinks#2}}}%
3112   }%
3113 }%
3114 {%
3115   \href{\glsgetattribute{\glslabel}{targeturl}}{%
3116     {{\glsxtrprotectlinks#2}}}%
3117   }%
3118 }%
3119 {%
  Check for alias.
3120   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
3121   \ifdefvoid\gloaliaslabel
3122   {%
3123     \glsxtrhyperlink{#1}{{\glsxtrprotectlinks#2}}}%
3124   }%
3125   {%
  Redirect link to the alias target.
3126     \glsxtrhyperlink
3127     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
3128     {{\glsxtrprotectlinks#2}}}%
3129   }%
3130 }%
3131 }

```

`\glsxtrhyperlink` Allows integration with the base glossaries package's `debug=showtargets` option.

```

3132 \ifdef\@glsshowtarget
3133 {
3134   \newcommand{\glsxtrhyperlink}[2]{%
3135     \@glsshowtarget{#1}%
3136     \hyperlink{#1}{#2}%
3137   }%
3138 }
3139 {
3140   \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%

```

3141 }

`glsdisablehyper` Redefine to set `\glslabel` (to allow it to be picked up by `\glsdohyperlink`). Also made it robust and added grouping to localise the definition of `\glslabel`. The original internal command `@glo@label` could probably be simply replaced with `\glslabel`, but it's retained in case its removal causes unexpected problems.

```

3142 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
3143   \glsdoifexists{#2}%
3144   {%
3145     \def\@glo@label{#2}%
3146     {\edef\glslabel{#2}%
3147       \@glslink{\glslinkprefix\glslabel}{#1}}%
3148   }%
3149 }
```

`glsdisablehyper` Redefine in case we have an old version of glossaries. This now uses `\def` rather than `\let` to allow for redefinitions of `\glsdohyperlink`.

```

3150 \renewcommand{\glsdisablehyper}{%
3151   \KV@glslink@hyperfalse
3152   \def\@glslink{\glsdohyperlink}%
3153   \let\@glslink\@secondoftwo
3154 }
```

`\glsenablehyper` This now uses `\def` rather than `\let` to allow for redefinitions of `\glsdohypertarget` and `\glsdohyperlink`.

```

3155 \renewcommand{\glsenablehyper}{%
3156   \KV@glslink@hypertrue
3157   \def\@glslink{\glsdohyperlink}%
3158   \def\@glslink{\glsdohypertarget}%
3159 }
```

`\glsdohyperlink` This command was only introduced in glossaries v4.20, so it may not be defined (therefore use `\def`). For older glossaries versions, this won't be used if `hyperref` hasn't been loaded, which means the indexing will still take place. The generated text is scoped (the link text in `\hyperlink` is also scoped, so it's consistent).

```

3160 \def\glsdohyperlink#1#2{{\glstrprotectlinks #2}}
```

`\@glslink` Reset `\@glslink` with patched versions:

```

3161 \ifcsundef{hyperlink}%
3162 {%
3163   \def\@glslink{\glsdohyperlink}
3164 }%
3165 {%
3166   \def\@glslink{\glsdohypertarget}
3167 }
```

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

```

3168 \newcommand*{\glxtrprotectlinks}{%
3169   \KV@glslink@hyperfalse
3170   \KV@glslink@noindextrue
3171   \let\@gls@\@glxtrp@text@
3172   \let\@Gls@\@Glsxtrp@text@
3173   \let\@GLS@\@GLSxtrp@text@
3174   \let\@glspl@\@glxtrp@plural@
3175   \let\@Glspl@\@Glsxtrp@plural@
3176   \let\@GLSpl@\@GLSxtrp@plural@
3177   \let\@glxtrshort@\@glxtrp@short@
3178   \let\@Glsxtrshort@\@Glsxtrp@short@
3179   \let\@GLSxtrshort@\@GLSxtrp@short@
3180   \let\@glxtrlong@\@glxtrp@long@
3181   \let\@Glsxtrlong@\@Glsxtrp@long@
3182   \let\@GLSxtrlong@\@GLSxtrp@long@
3183   \let\@glxtrshortpl@\@glxtrp@shortpl@
3184   \let\@Glsxtrshortpl@\@Glsxtrp@shortpl@
3185   \let\@GLSxtrshortpl@\@GLSxtrp@shortpl@
3186   \let\@glxtrlongpl@\@glxtrp@longpl@
3187   \let\@Glsxtrlongpl@\@Glsxtrp@longpl@
3188   \let\@GLSxtrlongpl@\@GLSxtrp@longpl@
3189   \let\@acrshort@\@glxtrp@acrshort@
3190   \let\@Acrshort@\@Glsxtrp@acrshort@
3191   \let\@ACRshort@\@GLSxtrp@acrshort@
3192   \let\@acrshortpl@\@glxtrp@acrshortpl@
3193   \let\@Acrshortpl@\@Glsxtrp@acrshortpl@
3194   \let\@ACRshortpl@\@GLSxtrp@acrshortpl@
3195   \let\@acrlong@\@glxtrp@acrlong@
3196   \let\@Acrlong@\@Glsxtrp@acrlong@
3197   \let\@ACRlong@\@GLSxtrp@acrlong@
3198   \let\@acrlongpl@\@glxtrp@acrlongpl@
3199   \let\@Acrlongpl@\@Glsxtrp@acrlongpl@
3200   \let\@ACRlongpl@\@GLSxtrp@acrlongpl@
3201 }

```

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

@glxtrp@text@

```
3202 \def\@glxtrp@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtrp@text@

```
3203 \def\@Glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtrp@text@

```
3204 \def\@GLSxtrp@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtrp@plural@

```
3205 \def\@glxtrp@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
3206 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2} [#3]}
```

LSxtr@p@plural@

```
3207 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2} [#3]}
```

glxtr@p@short@

```
3208 \def\@glxtr@p@short@#1#2[#3]{%
3209   {%
3210     \glsssetabbrvfmt{\glscategory{#2}}%
3211     \glsabbrvfont{\glentryshort{#2}}#3%
3212   }%
3213 }
```

Glsxtr@p@short@

```
3214 \def\@Glsxtr@p@short@#1#2[#3]{%
3215   {%
3216     \glsssetabbrvfmt{\glscategory{#2}}%
3217     \glsabbrvfont{\Glsentryshort{#2}}#3%
3218   }%
3219 }
```

GLSxtr@p@short@

```
3220 \def\@GLSxtr@p@short@#1#2[#3]{%
3221   {%
3222     \glsssetabbrvfmt{\glscategory{#2}}%
3223     \mfirstucMakeUppercase{\glsabbrvfont{\glentryshort{#2}}#3}%
3224   }%
3225 }
```

sxtr@p@shortpl@

```
3226 \def\@glxtr@p@shortpl@#1#2[#3]{%
3227   {%
3228     \glsssetabbrvfmt{\glscategory{#2}}%
3229     \glsabbrvfont{\glentryshortpl{#2}}#3%
3230   }%
3231 }
```

Sxtr@p@shortpl@

```
3232 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
3233   {%
3234     \glsssetabbrvfmt{\glscategory{#2}}%
3235     \glsabbrvfont{\Glsentryshortpl{#2}}#3%
3236   }%
3237 }
```

SXtr@p@shortpl@

```
3238 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
```

```

3239  {%
3240    \glsssetabbrvfmt{\glscategory{#2}}%
3241    \mfirstucMakeUppercase{\glssabbrvfont{\glssentryshortpl{#2}}#3}%
3242  }%
3243 }

@glssxtr@p@long@
3244 \def\@glssxtr@p@long@#1#2[#3]{\glssentrylong{#2}#3}

@Glsxtr@p@long@
3245 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

@GLSxtr@p@long@
3246 \def\@GLSxtr@p@long@#1#2[#3]{%
3247   {\mfirstucMakeUppercase{\glsslongfont{\glssentrylong{#2}}#3}}}

lssxtr@p@longpl@
3248 \def\@lssxtr@p@longpl@#1#2[#3]{\glssentrylongpl{#2}#3}

Lssxtr@p@longpl@
3249 \def\@Lssxtr@p@longpl@#1#2[#3]{\glsslongfont{\Glsentrylongpl{#2}}#3}

LSxtr@p@longpl@
3250 \def\@GLSxtr@p@longpl@#1#2[#3]{%
3251   {\mfirstucMakeUppercase{\glsslongfont{\glssentrylongpl{#2}}#3}}}

xtr@p@acrshort@
3252 \def\@glssxtr@p@acrshort@#1#2[#3]{\acronymfont{\glssentryshort{#2}}#3}

Xtr@p@acrshort@
3253 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

Xtr@p@acrshort@
3254 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
3255   {\mfirstucMakeUppercase{\acronymfont{\glssentryshort{#2}}#3}}}

r@p@acrshortpl@
3256 \def\@glssxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glssentryshortpl{#2}}#3}

R@p@acrshortpl@
3257 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}

R@p@acrshortpl@
3258 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
3259   {\mfirstucMakeUppercase{\acronymfont{\glssentryshortpl{#2}}#3}}}

ssxtr@p@acrlong@
3260 \def\@glssxtr@p@acrlong@#1#2[#3]{\glssentrylong{#2}#3}

```

sxtr@p@acrlong@

```
3261 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}
```

Sxtr@p@acrlong@

```
3262 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
3263  {\mfirstucMakeUppercase{\Glsentrylong{#2}#3}}}
```

tr@p@acrlongpl@

```
3264 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
3265 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
3266 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
3267  {\mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glsxtrp@opt

```
3268 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

\glsxtrsetpopts Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.

```
3269 \newcommand*{\glsxtrsetpopts}[1]{%
3270   \renewcommand*{\@glsxtrp@opt}{#1}%
3271 }
```

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

```
3272 \newcommand*{\glossxtrsetpopts}{%
3273   \glsxtrsetpopts{noindex}%
3274 }
```

\@@glsxtrp

```
3275 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
3276  {%
3277    \let\glspostlinkhook\relax
3278    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
3279  }%
3280 }
```

\@glsxtrp

```
3281 \newrobustcmd*{\@glsxtrp}[2]{%
3282   \ifcsdef{gls#1}%
3283   {%
3284     \@glsxtrp{gls#1}{#2}%
3285   }%
```

```

3286 {%
3287   \ifcsdef{glsxtr#1}%
3288   {%
3289     \@glsxtrp{glsxtr#1}{#2}%
3290   }%
3291   {%
3292     \PackageError{glossaries-extra}{‘#1’ not recognised by
3293       \string\glsxtrp}{}%
3294   }%
3295 }%
3296 }

```

\@Glsxtrp

```

3297 \newrobustcmd*{\@Glsxtrp}[2]{%
3298   \ifcsdef{Gls#1}%
3299   {%
3300     \@glsxtrp{Gls#1}{#2}%
3301   }%
3302   {%
3303     \ifcsdef{Glsxtr#1}%
3304     {%
3305       \@glsxtrp{Glsxtr#1}{#2}%
3306     }%
3307     {%
3308       \PackageError{glossaries-extra}{‘#1’ not recognised by
3309         \string\Glsxtrp}{}%
3310     }%
3311   }%
3312 }

```

\@GLSxtrp

```

3313 \newrobustcmd*{\@GLSxtrp}[2]{%
3314   \ifcsdef{GLS#1}%
3315   {%
3316     \@glsxtrp{GLS#1}{#2}%
3317   }%
3318   {%
3319     \ifcsdef{GLSxtr#1}%
3320     {%
3321       \@glsxtrp{GLSxtr#1}{#2}%
3322     }%
3323     {%
3324       \PackageError{glossaries-extra}{‘#1’ not recognised by
3325         \string\GLSxtrp}{}%
3326     }%
3327   }%
3328 }

```

\glsxtr@entry@p

```

3329 \newrobustcmd*{\glstr@headentry@p}[2]{%
3330   \gl@ifattribute{#1}{headuc}{true}%
3331   {%
3332     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3333   }%
3334   {%
3335     \@gls@entry@field{#1}{#2}%
3336   }%
3337 }

```

`\glstrp` Not robust as it needs to expand somewhat.

```

3338 \ifdef\texorpdfstring
3339 {
3340   \newcommand{\glstrp}[2]{%
3341     \protect\NoCaseChange
3342     {%
3343       \protect\texorpdfstring
3344       {%
3345         \protect\glstrifinmark
3346         {%
3347           \ifcsdef{glstrhead#1}%
3348           {%
3349             {\protect\csuse{glstrhead#1}{#2}}%
3350           }%
3351           {%
3352             \glstr@headentry@p{#2}{#1}%
3353           }%
3354         }%
3355         {%
3356           \@glstrp{#1}{#2}%
3357         }%
3358       }%
3359     }%
3360     \protect\@gls@entry@field{#2}{#1}%
3361   }%
3362 }%
3363 }
3364 }
3365 {
3366   \newcommand{\glstrp}[2]{%
3367     \protect\NoCaseChange
3368     {%
3369       \protect\glstrifinmark
3370       {%
3371         \ifcsdef{glstrhead#1}%
3372         {%
3373           {\protect\csuse{glstrhead#1}}%
3374         }%
3375         {%

```

```

3376         \glxtr@headentry@p{#2}{#1}%
3377     }%
3378 }%
3379 {%
3380     \@glxtrp{#1}{#2}%
3381 }%
3382 }%
3383 }
3384 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3385 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

3386 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

3387 \ifdef\texorpdfstring
3388 {
3389     \newcommand{\Glsxtrp}[2]{%
3390         \protect\NoCaseChange
3391         {%
3392             \protect\texorpdfstring
3393             {%
3394                 \protect\glxtrifinmark
3395                 {%
3396                     \ifcsdef{Glsxtrhead#1}%
3397                     {%
3398                         {\protect\csuse{Glsxtrhead#1}{#2}}%
3399                     }%
3400                     {%
3401                         \protect\@Gls@entry@field{#2}{#1}%
3402                     }%
3403                 }%
3404                 {%
3405                     \@Glsxtrp{#1}{#2}%
3406                 }%
3407             }%
3408             {%
3409                 \protect\@gl@entry@field{#2}{#1}%
3410             }%
3411         }%
3412     }
3413 }
3414 {
3415     \newcommand{\Glsxtrp}[2]{%

```

```

3416 \protect\NoCaseChange
3417 {%
3418 \protect\glxtrifinmark
3419 {%
3420 \ifcsdef{Glsxtrhead#1}%
3421 {%
3422 {\protect\csuse{Glsxtrhead#1}}%
3423 }%
3424 {%
3425 \protect\@Gls@entry@field{#2}{#1}%
3426 }%
3427 }%
3428 {%
3429 \@Glsxtrp{#1}{#2}%
3430 }%
3431 }%
3432 }
3433 }

```

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

```

3434 \ifdef\teorpdfstring
3435 {
3436 \newcommand{\GLSxtrp}[2]{%
3437 \protect\NoCaseChange
3438 {%
3439 \protect\teorpdfstring
3440 {%
3441 \protect\glxtrifinmark
3442 {%
3443 \ifcsdef{GLSxtr#1}%
3444 {%
3445 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3446 }%
3447 {%
3448 \protect\mfirstucMakeUppercase
3449 {%
3450 \protect\@gls@entry@field{#2}{#1}%
3451 }%
3452 }%
3453 }%
3454 {%
3455 \@GLSxtrp{#1}{#2}%
3456 }%
3457 }%
3458 {%
3459 \protect\@gls@entry@field{#2}{#1}%
3460 }%
3461 }%
3462 }

```

```

3463 }
3464 {
3465   \newcommand{\GLSxtrp}[2]{%
3466     \protect\NoCaseChange
3467     {%
3468       \protect\glxtrifinmark
3469       {%
3470         \ifcsdef{GLSxtr#1}%
3471         {%
3472           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3473         }%
3474         {%
3475           \protect\mfirstucMakeUppercase
3476           {%
3477             \protect\@gls@entry@field{#2}{#1}%
3478           }%
3479         }%
3480       }%
3481       {%
3482         \@GLSxtrp{#1}{#2}%
3483       }%
3484     }%
3485   }
3486 }

```

1.3.5 Entry Counting

The (use) 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 instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceeding the specified threshold. For link counting, see Section 1.4.

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be temporarily disabled, \@glsunset is let to \@glxtr@unset, which performs the actual unsetting through @@glsunset and then does the hook. This means that the unsetting (and the hook) can switched off by redefining \@glsunset and then switched back on again by changing the definition back to \@glxtr@unset.

```

\@glxtr@unset  Global unset.
3487 \newcommand*{\@glxtr@unset}[1]{%
3488   \@glsunset{#1}%
3489   \glxtrpostunset{#1}%
3490 }%

\@glsunset  Global unset.
3491 \let\@glsunset\@glxtr@unset

```

glsxtrpostunset

```
3492 \newcommand*{\glsxtrpostunset}[1]{}
```

Provide a command to store a list of labels that will need unsetting.

tUnsetBuffering

```
3493 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3494   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3495 }
```

tUnsetBuffering Unstarred version doesn't check for duplicates.

```
3496 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3497   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3498   \def\@glsxtr@unset@buffer{}%
3499   \let\@glsunset\@glsxtrbuffer@unset
3500 }
```

tUnsetBuffering Starred version checks for duplicates.

```
3501 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3502   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3503   \def\@glsxtr@unset@buffer{}%
3504   \let\@glsunset\@glsxtrbuffer@nodup@unset
3505 }
```

xtrbuffer@unset This must use a global change since \gls may have to be placed inside \mbox (for example, with soul commands).

```
3506 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3507   \listxadd\@glsxtr@unset@buffer{#1}%
3508 }
```

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using \xifinlist as the added complexity might cause problems that the buffering is trying to overcome.)

```
3509 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3510   \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
3511   {\listxadd\@glsxtr@unset@buffer{#1}}%
3512 }
```

pUnsetBuffering

```
3513 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3514   \ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3515 }
```

pUnsetBuffering Unstarred form (global unset).

```
3516 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3517   \let\@glsunset\@glsxtr@unset
3518   \forlistloop\@glsunset\@glsxtr@unset@buffer
3519   \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3520 }
```

pUnsetBuffering Starred form (local unset).

```
3521 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3522   \forlistloop\@glslocalunset\@glxtr@unset@buffer
3523   \let\@glsunset\@glxtr@unset
3524 }
```

setBufferedList Iterate over labels stored in the current buffer. The argument is the handler macro.

```
3525 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3526   \forlistloop#1\@glxtr@unset@buffer
3527 }
```

\@glslocalunset Local unset.

```
3528 \renewcommand*{\@glslocalunset}[1]{%
3529   \@glslocalunset{#1}%
3530   \glxtrpostlocalunset{#1}%
3531 }%
```

rpostlocalunset

```
3532 \newcommand*{\glxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
3533 \renewcommand*{\@glsreset}[1]{%
3534   \@glsreset{#1}%
3535   \glxtrpostreset{#1}%
3536 }%
```

glxtrpostreset

```
3537 \newcommand*{\glxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
3538 \renewcommand*{\@glslocalreset}[1]{%
3539   \@glslocalreset{#1}%
3540   \glxtrpostlocalreset{#1}%
3541 }%
```

rpostlocalreset

```
3542 \newcommand*{\glxtrpostlocalreset}[1]{}
```

slocalreseteach Locally reset a list of entries.

```
3543 \newcommand*{\glslocalreseteach}[1]{%
3544   \gls@ifnotmeasuring
3545   {%
3546     \@for\@gls@thislabel:=#1\do{%
3547       \glsdoifexists{\@gls@thislabel}%
3548       {%
3549         \@glslocalreset{\@gls@thislabel}%
3550       }%
3551     }%
```

```

3552 }%
3553 }

```

slocalunseteach Locally unset a list of entries.

```

3554 \newcommand*{\glslocalunseteach}[1]{%
3555   \gls@ifnotmeasuring
3556   {%
3557     \@for\@gls@thislabel:=#1\do{%
3558       \glsdoifexists{\@gls@thislabel}%
3559       {%
3560         \@glslocalunset{\@gls@thislabel}%
3561       }%
3562     }%
3563   }%
3564 }

```

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

```

3565 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```

3566   \glsenableentrycount

```

Redefine \gls etc:

```

3567   \renewcommand*{\gls}{\cglsl}%
3568   \renewcommand*{\Gls}{\cGls}%
3569   \renewcommand*{\glspl}{\cglspl}%
3570   \renewcommand*{\Glspl}{\cGlspl}%
3571   \renewcommand*{\GLS}{\cGLS}%
3572   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3573   \@glsxtr@setentrycountunsetattr{#1}{#2}%

```

In case this command is used again:

```

3574   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3575   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3576     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3577       can't be used with \string\GlsXtrEnableEntryCounting}%
3578     {Use one or other but not both commands}}%
3579 }

```

ycountunsetattr

```

3580 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
3581   \@for\@glsxtr@cat:=#1\do
3582   {%
3583     \ifdefempty{\@glsxtr@cat}{}%
3584     {%
3585       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3586     }%

```

```

3587 }%
3588 }

```

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

nableentrycount

```

3589 \renewcommand*{\glsenableentrycount}{%
  Enable new fields:
3590   \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
  Just in case the user has switched on the docdef option.
3591   \renewcommand*{\gls@defdocnewglossaryentry}{%
3592     \renewcommand*{\newglossaryentry[2]}{%
3593       \PackageError{glossaries}{\string\newglossaryentry\space
3594         may only be used in the preamble when entry counting has
3595         been activated}{If you use \string\glsenableentrycount\space
3596         you must place all entry definitions in the preamble not in
3597         the document environment}}%
3598   }%
3599 }%

```

New commands to access new fields:

```

3600 \newcommand*{\glstentrycurrcount}[1]{%
3601   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3602   {0}{\@gls@entry@field{##1}{currcount}}}%
3603 }%
3604 \newcommand*{\glstentryprevcount}[1]{%
3605   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3606   {0}{\@gls@entry@field{##1}{prevcount}}}%
3607 }%

```

Adjust post unset and reset:

```

3608 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
3609 \renewcommand*{\glsxtrpostunset}[1]{%
3610   \@glsxtr@entrycount@org@unset{##1}%
3611   \@gls@increment@currcount{##1}%
3612 }%
3613 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
3614 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3615   \@glsxtr@entrycount@org@localunset{##1}%
3616   \@gls@local@increment@currcount{##1}%
3617 }%
3618 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
3619 \renewcommand*{\glsxtrpostreset}[1]{%
3620   \@glsxtr@entrycount@org@reset{##1}%
3621   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3622 }%
3623 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
3624 \renewcommand*{\glsxtrpostlocalreset}[1]{%
3625   \@glsxtr@entrycount@org@localreset{##1}%

```

```

3626 \csdef{glo@glstetoklabel{##1}@currcount}{0}%
3627 }%

```

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

```

3628 \let\@cgl@s\@cgl@s@
3629 \let\@cgl@spl\@cgl@spl@

3630 \let\@cGl@s\@cGl@s@
3631 \let\@cGl@spl\@cGl@spl@
3632 \let\@cGL@s\@cGL@s@
3633 \let\@cGL@spl\@cGL@spl@

```

The rest is as the original definition.

```

3634 \AtEndDocument{\@gl@s@write@entrycounts}%
3635 \renewcommand*{\@gl@s@entry@count}[2]{%
3636 \csgdef{glo@glstetoklabel{##1}@prevcount}{##2}%
3637 }%
3638 \let\gl@senableentrycount\relax
3639 \renewcommand*{\gl@senableentryunitcount}{%
3640 \PackageError{glossaries-extra}{\string\gl@senableentryunitcount\space
3641 can't be used with \string\gl@senableentrycount}%
3642 {Use one or other but not both commands}%
3643 }%
3644 }

```

`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.

```

3645 \renewcommand*{\@gl@s@write@entrycounts}{%
3646 \immediate\write\@auxout
3647 {\string\providecommand*{\string\@gl@s@entry@count}[2]{}}%
3648 \count@=0\relax
3649 \forallgl@sentries{\@gl@sentry}{%
3650 \gl@shasattribute{\@gl@sentry}{entrycount}%
3651 {%
3652 \ifgl@sused{\@gl@sentry}%
3653 {%
3654 \immediate\write\@auxout
3655 {\string\@gl@s@entry@count{\@gl@sentry}{\gl@sentrycurrcount{\@gl@sentry}}}%
3656 }%
3657 }%
3658 \advance\count@ by \@ne
3659 }%
3660 }%
3661 }%
3662 \ifnum\count@=0
3663 \GlossariesExtraWarningNoLine{Entry counting has been enabled
3664 \MessageBreak with \string\gl@senableentrycount\space but the
3665 \MessageBreak attribute 'entrycount' hasn't
3666 \MessageBreak been assigned to any of the defined

```

```

3667 \MessageBreak entries}%
3668 \fi
3669 }

```

```

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

```

```

3670 \newcommand*\glxtrifcounttrigger}[3]{%
3671 \glshasattribute{#1}{entrycount}%
3672 {%
3673 \ifnum\gl Sentryprevcount{#1}>\glsetattribute{#1}{entrycount}\relax
3674 #3%
3675 \else
3676 #2%
3677 \fi
3678 }%
3679 {#3}%
3680 }

```

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

\@@cgl@

```

3681 \def\@@cgl@#1#2[#3]{%
3682 \glxtrifcounttrigger{#2}%
3683 {%
3684 \cglformat{#2}{#3}%
3685 \glunset{#2}%
3686 }%
3687 {%
3688 \@gls@{#1}{#2}[#3]%
3689 }%
3690 }%

```

\@@cgl@pl@

```

3691 \def\@@cgl@pl@#1#2[#3]{%
3692 \glxtrifcounttrigger{#2}%
3693 {%
3694 \cgl@plformat{#2}{#3}%
3695 \glunset{#2}%
3696 }%
3697 {%
3698 \@glspl@{#1}{#2}[#3]%
3699 }%
3700 }%

```

\@@cGls@

```

3701 \def\@@cGls@#1#2[#3]{%
3702   \glstrifcounttrigger{#2}%
3703   {%
3704     \cGlsformat{#2}{#3}%
3705     \glset{#2}%
3706   }%
3707   {%
3708     \@Gls@{#1}{#2}[#3]%
3709   }%
3710 }%

```

\@@cGlspl@

```

3711 \def\@@cGlspl@#1#2[#3]{%
3712   \glstrifcounttrigger{#2}%
3713   {%
3714     \cGlsplformat{#2}{#3}%
3715     \glset{#2}%
3716   }%
3717   {%
3718     \@Glspl@{#1}{#2}[#3]%
3719   }%
3720 }%

```

\@@cGLS@

```

3721 \def\@@cGLS@#1#2[#3]{%
3722   \glstrifcounttrigger{#2}%
3723   {%
3724     \cGLSformat{#2}{#3}%
3725     \glset{#2}%
3726   }%
3727   {%
3728     \@GLS@{#1}{#2}[#3]%
3729   }%
3730 }%

```

\@@cGLSpl@

```

3731 \def\@@cGLSpl@#1#2[#3]{%
3732   \glstrifcounttrigger{#2}%
3733   {%
3734     \cGLSplformat{#2}{#3}%
3735     \glset{#2}%
3736   }%
3737   {%
3738     \@GLSpl@{#1}{#2}[#3]%
3739   }%
3740 }%

```

Remove default warnings from \cgl's etc so that it can be used interchangeable with \gls etc.

```
\@cgl@s@
3741 \def\@cgl@s@#1#2[#3]{\@gls@{#1}-{#2} [#3]}
```

```
\@cGls@
3742 \def\@cGls@#1#2[#3]{\@Gls@{#1}-{#2} [#3]}
```

```
\@cgl spl@
3743 \def\@cgl spl@#1#2[#3]{\@gl spl@{#1}-{#2} [#3]}
```

```
\@cGlspl@
3744 \def\@cGlspl@#1#2[#3]{\@Glspl@{#1}-{#2} [#3]}
```

Add all upper case versions not provided by glossaries.

```
\cGLS
3745 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

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

```
3746 \newcommand*{\@cGLS}[2] [] {%
3747   \new@ifnextchar[\@cGLS@{#1}-{#2}]{\@cGLS@{#1}-{#2} []}%
3748 }
```

```
\@cGLS@
3749 \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.

```
3750 \newcommand*{\cGLSformat}[2] {%
3751   \expandafter\mfirstucMakeUppercase\expandafter{\cgl sformat{#1}-{#2}}%
3752 }
```

```
\cGLSpl
3753 \newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

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

```
3754 \newcommand*{\@cGLSpl}[2] [] {%
3755   \new@ifnextchar[\@cGLSpl@{#1}-{#2}]{\@cGLSpl@{#1}-{#2} []}%
3756 }
```

```
\@cGLSpl@
3757 \def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}-{#2} [#3]}
```

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

```
3758 \newcommand*{\cGLSplformat}[2] {%
3759   \expandafter\mfirstucMakeUppercase\expandafter{\cgl splformat{#1}-{#2}}%
3760 }
```

Modify the trigger formats to check for the regular attribute.

`\cglsformat`

```
3761 \renewcommand*{\cglsformat}[2]{%
3762   \glsifregular{#1}
3763   {\glsentryfirst{#1}}%
3764   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%2%
3765 }
```

`\cGlsformat`

```
3766 \renewcommand*{\cGlsformat}[2]{%
3767   \glsifregular{#1}
3768   {\Glsentryfirst{#1}}%
3769   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%2%
3770 }
```

`\cglsplformat`

```
3771 \renewcommand*{\cglsplformat}[2]{%
3772   \glsifregular{#1}
3773   {\glsentryfirstplural{#1}}%
3774   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%2%
3775 }
```

`\cGlsplformat`

```
3776 \renewcommand*{\cGlsplformat}[2]{%
3777   \glsifregular{#1}
3778   {\Glsentryfirstplural{#1}}%
3779   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%2%
3780 }
```

New code similar to above for unit counting.

`defunitcounters`

```
3781 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3782   \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
3783   \ifdefvoid\@glo@countunit
3784   {}%
3785   {%
3786     \@glsxtr@ifunitcounter{\@glo@countunit}%
3787     {}%
3788     {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
3789   }%
3790 }
```

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

```
3791 \newcommand*{\@glsxtr@unitcountlist}{}
```

@addunitcounter

```
3792 \newcommand*{\@glxstr@addunitcounter}[1]{%
3793   \listadd{\@glxstr@unitcountlist}{#1}%
3794   \ifcsundef{glxstr@theunit@#1}
3795   {%
3796     \ifcsdef{theH#1}%
3797     {\csdef{glxstr@theunit@#1}{\csuse{theH#1}}}%
3798     {\csdef{glxstr@theunit@#1}{\csuse{the#1}}}%
3799   }%
3800   {}%
3801 }
```

r@ifunitcounter

```
3802 \newcommand*{\@glxstr@ifunitcounter}[3]{%
3803   \xifinlist{#1}{\@glxstr@unitcountlist}{#2}{#3}%
3804 }
```

urrentunitcount

```
3805 \newcommand*\@glxstr@currentunitcount[1]{%
3806   glo@\glsdetoklabel{#1}@currunit@\glsggetattribute{#1}{unitcount}.%
3807   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3808 }
```

eviousunitcount

```
3809 \newcommand*\@glxstr@previousunitcount[1]{%
3810   glo@\glsdetoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
3811   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3812 }
```

t@currunitcount

```
3813 \newcommand*{\@glx@increment@currunitcount}[1]{%
3814   \glshasattribute{#1}{unitcount}%
3815   {%
3816     \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3817     \ifcsundef{\@glxstr@csname}%
3818     {%
3819       \csgdef{\@glxstr@csname}{1}%
3820       \listcsxadd
3821       {glo@\glsdetoklabel{#1}@unitlist}%
3822       {\glsggetattribute{#1}{unitcount}.%
3823       \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3824     }%
3825   }%
3826   {%
3827     \csxdef{\@glxstr@csname}%
3828     {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3829   }%
3830 }%
3831 {}%
```

3832 }

t@currunitcount

```
3833 \newcommand*{\@gls@local@increment@currunitcount}[1]{%
3834   \gls@hasattribute{#1}{unitcount}%
3835   {%
3836     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3837     \ifcsundef{\@glsxtr@csname}%
3838     {%
3839       \csdef{\@glsxtr@csname}{1}%
3840       \listcseadd
3841       {glo@glsdetoklabel{#1}@unitlist}%
3842       {\glsgetattribute{#1}{unitcount}.%
3843       \csuse{glsxtr@theunit@glsggetattribute{#1}{unitcount}}}%
3844     }%
3845   }%
3846   {%
3847     \csedef{\@glsxtr@csname}%
3848     {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3849   }%
3850 }%
3851 {}%
3852 }
```

r@currunitcount

```
3853 \newcommand*{\@glsxtr@currunitcount}[2]{%
3854   \ifcsundef
3855   {glo@glsdetoklabel{#1}@currunit@#2}%
3856   {0}%
3857   {\csuse{glo@glsdetoklabel{#1}@currunit@#2}}%
3858 }%
```

r@prevunitcount

```
3859 \newcommand*{\@glsxtr@prevunitcount}[2]{%
3860   \ifcsundef
3861   {glo@glsdetoklabel{#1}@prevunit@#2}%
3862   {0}%
3863   {\csuse{glo@glsdetoklabel{#1}@prevunit@#2}}%
3864 }%
```

entryunitcount

```
3865 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3866   \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3867   \renewcommand*{\gls@defdocnewglossaryentry}{%
3868     \renewcommand*\newglossaryentry[2]{%
3869       \PackageError{glossaries}{\string\newglossaryentry\space
```

```

3870     may only be used in the preamble when entry counting has
3871     been activated}{If you use \string\glsenableentryunitcount\space
3872     you must place all entry definitions in the preamble not in
3873     the document environment}%
3874 }%
3875 }%

```

New commands to access new fields:

```

3876 \newcommand*{\glsentrycurrcount}[1]{%
3877   \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3878   \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3879 }%
3880 \newcommand*{\glsentryprevcount}[1]{%
3881   \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3882   \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3883 }%

```

Access total count:

```

3884 \newcommand*{\glsentryprevtotalcount}[1]{%
3885   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3886     {0}%
3887     {%
3888       \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3889     }%
3890 }%

```

Access max value:

```

3891 \newcommand*{\glsentryprevmaxcount}[1]{%
3892   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3893     {0}%
3894     {%
3895       \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3896     }%
3897 }%

```

Adjust post unset and reset:

```

3898 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
3899 \renewcommand*{\glsxtrpostunset}[1]{%
3900   \@glsxtr@entryunitcount@org@unset{##1}%
3901   \@gls@increment@currunitcount{##1}%
3902 }%
3903 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
3904 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3905   \@glsxtr@entryunitcount@org@localunset{##1}%
3906   \@gls@local@increment@currunitcount{##1}%
3907 }%
3908 \let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset
3909 \renewcommand*{\glsxtrpostreset}[1]{%
3910   \glshasattribute{##1}{unitcount}%
3911   {%
3912     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%

```

```

3913     \ifcsundef{\@glxtr@csname}%
3914     {}%
3915     {\csgdef{\@glxtr@csname}{0}}%
3916     }%
3917     {}%
3918     }%
3919     \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
3920     \renewcommand*{\glxtrpostlocalreset}[1]{%
3921       \@glxtr@entryunitcount@org@localreset{##1}%
3922       \glshasattribute{##1}{unitcount}%
3923       {%
3924         \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3925         \ifcsundef{\@glxtr@csname}%
3926         {}%
3927         {\csdef{\@glxtr@csname}{0}}%
3928         }%
3929         {}%
3930     }%

```

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

```

3931     \let\@cgl@\@cgl@
3932     \let\@cgl@pl@\@cgl@pl@

3933     \let\@cGls@\@cGls@
3934     \let\@cGls@pl@\@cGls@pl@
3935     \let\@cGLS@\@cGLS@
3936     \let\@cGLS@pl@\@cGLS@pl@

```

Write information to the aux file.

```

3937     \AtEndDocument{\@gls@write@entryunitcounts}%
3938     \renewcommand*{\@gls@entry@unitcount}[3]{%
3939       \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
3940       \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3941       {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
3942       {%
3943         \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
3944           \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
3945         }%
3946         \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3947         {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
3948         {%
3949           \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3950             \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3951           \fi
3952         }%
3953       }%
3954     \let\glsenableentryunitcount\relax
3955     \renewcommand*{\glsenableentrycount}{%
3956       \PackageError{glossaries-extra}{\string\glsenableentrycount\space

```

```

3957     can't be used with \string\glsenableentryunitcount}%
3958     {Use one or other but not both commands}%
3959   }%
3960 }
3961 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3962 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3963 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3964   \immediate\write\@auxout
3965   {\string\@gls@entry@unitcount
3966    {\@gls@entry}%
3967    {\@gls@xtr@currunitcount{\@gls@entry}{#1}%
3968    }%
3969    {#1}}%
3970 }

```

entryunitcounts

```

3971 \newcommand*{\@gls@write@entryunitcounts}{%
3972   \immediate\write\@auxout
3973   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3974   \count@=0\relax
3975   \forallglsentries{\@gls@entry}{%
3976     \gls@hasattribute{\@gls@entry}{unitcount}%
3977     {%
3978       \ifglsused{\@gls@entry}%
3979       {%
3980         \forlistcsloop
3981           {\@gls@write@entryunitcounts@do}%
3982           {glo@\gls@detoklabel{\@gls@entry}@unitlist}%
3983         }%
3984       }%
3985       \advance\count@ by \@ne
3986     }%
3987   }%
3988 }%
3989 \ifnum\count@=0
3990   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3991   \MessageBreak with \string\glsenableentryunitcount\space but the
3992   \MessageBreak attribute 'unitcount' hasn't
3993   \MessageBreak been assigned to any of the defined
3994   \MessageBreak entries}%
3995 \fi
3996 }

```

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

```

3997 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
    Enable entry counting:
3998   \glsenableentryunitcount
    Redefine \gls etc:
3999   \renewcommand*{\gls}{\cglss}%
4000   \renewcommand*{\Gls}{\cGls}%
4001   \renewcommand*{\glspl}{\cglsspl}%
4002   \renewcommand*{\Glspl}{\cGlspl}%
4003   \renewcommand*{\GLS}{\cGLS}%
4004   \renewcommand*{\GLSpl}{\cGLSpl}%
    Set the entrycount attribute:
4005   \@glsextr@setentryunitcountunsetattr{#1}{#2}{#3}%
    In case this command is used again:
4006   \let\GlsXtrEnableEntryUnitCounting\@glsextr@setentryunitcountunsetattr
4007   \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
4008     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
4009       can't be used with \string\GlsXtrEnableEntryUnitCounting}%
4010     {Use one or other but not both commands}}%
4011 }

```

countunsetattr

```

4012 \newcommand*{\@glsextr@setentryunitcountunsetattr}[3]{%
4013   \@for\@glsextr@cat:=#1\do
4014   {%
4015     \ifdefempty{\@glsextr@cat}{}%
4016     {%
4017       \glsssetcategoryattribute{\@glsextr@cat}{entrycount}{#2}%
4018       \glsssetcategoryattribute{\@glsextr@cat}{unitcount}{#3}%
4019     }%
4020   }%
4021 }

```

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.

genericNewAcronym

```

4022 \renewcommand*{\SetGenericNewAcronym}{%
4023   \let\@Gls@entryname\@Gls@acrenryname
4024   \renewcommand{\newacronym}[4][ ]{%
4025     \ifdefempty{\@glsacronymlists}%

```

```

4026 {%
4027   \def\@glo@type{\acronymtype}%
4028   \setkeys{glossentry}{##1}%
4029   \DeclareAcronymList{\@glo@type}%
4030 }%
4031 {}%
4032 \glskeylisttok{##1}%
4033 \glslabeltok{##2}%
4034 \glsshorttok{##3}%
4035 \glslongtok{##4}%
4036 \newacronymhook
4037 \protected@edef\@do@newglossaryentry{%
4038   \noexpand\newglossaryentry{\the\glslabeltok}%
4039   {%
4040     type=\acronymtype,%
4041     name={\expandonce{\acronymentry{##2}}},%
4042     sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
4043     text={\the\glsshorttok},%
4044     short={\the\glsshorttok},%
4045     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4046     long={\the\glslongtok},%
4047     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4048     category=acronym,
4049     \GenericAcronymFields,%
4050     \the\glskeylisttok
4051   }%
4052 }%
4053 \@do@newglossaryentry
4054 }%
4055 \renewcommand*{\acrfullfmt}[3]{%
4056   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
4057 \renewcommand*{\Acrfullfmt}[3]{%
4058   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
4059 \renewcommand*{\ACRfullfmt}[3]{%
4060   \glslink[##1]{##2}{%
4061     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
4062 \renewcommand*{\acrfullplfmt}[3]{%
4063   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
4064 \renewcommand*{\Acrfullplfmt}[3]{%
4065   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
4066 \renewcommand*{\ACRfullplfmt}[3]{%
4067   \glslink[##1]{##2}{%
4068     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
4069 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
4070 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
4071 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
4072 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
4073 }

```

This will cause a problem for glossaries that contain a mixture of acronyms and abbrevia-

tions, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```
4074 \let\@glxtr@org@setacronymstyle\setacronymstyle
4075 \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`.

```
4076 \newcommand*{\MakeAcronymsAbbreviations}{%
4077   \renewcommand*{\newacronym}[4][]{%
4078     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4079   }%
4080   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4081   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
4082   \renewcommand*{\setacronymstyle}[1]{%
4083     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
4084       unavailable.
4085       Use \string\setabbreviationstyle\space instead.
4086       The original acronym interface can be restored with
4087       \string\RestoreAcronyms}{}%
4088   }%
4089   \renewcommand*{\newacronymstyle}[1]{%
4090     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4091       available unless you restore the original acronym interface with
4092       \string\RestoreAcronyms}%
4093     \@glxtr@org@newacronymstyle{##1}%
4094   }%
4095 }
```

Switch acronyms to abbreviations:

```
4096 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```
4097 \newcommand*{\RestoreAcronyms}{%
4098   \SetGenericNewAcronym
4099   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
4100   \renewcommand{\acronymfont}[1]{##1}%
4101   \let\setacronymstyle\@glxtr@org@setacronymstyle
4102   \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.

```
4103   \renewcommand*\@gls@link@checkfirsthyper{%
4104     \ifglused{\glslabel}%
4105     {\let\glxtrifwasfirstuse\@secondoftwo}
4106     {\let\glxtrifwasfirstuse\@firstoftwo}%
4107     \@glxtr@org@checkfirsthyper
4108   }
4109   \glsssetcategoryattribute{acronym}{regular}{false}%
```

```
4110 \setacronymstyle{long-short}%
4111 }
```

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

```
4112 \renewcommand*{\glsacspace}[1]{%
4113   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4114   \ifdim\dimen@<\glsacspacemax~\else\space\fi
4115 }
```

`\glsacspacemax` Value used in the above.

```
4116 \newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

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

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

`r@reg@glosslist`

```
4117 \newcommand*{\@glxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
4118 \let\@glxtr@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. The optional argument version shouldn't be used with record.

`\makeglossaries`

```
4119 \renewcommand*{\makeglossaries}[1][]{%
4120   \@glxtr@if@record@only
4121   {%
4122     \PackageError{glossaries-extra}{\string\makeglossaries\space
4123       not permitted\MessageBreak with record=\@glxtr@record@setting\space
4124       package option}%
4125     {You may only use \string\makeglossaries\space with
4126       record=off or record=alsoindex options}%
4127   }%
4128   {%
4129     \ifblank{#1}%
4130     {\@glxtr@org@makeglossaries}%
4131     {%
4132       \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
4133         \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4134           not permitted\MessageBreak with record=alsoindex package option}%
4135         {You may only use the hybrid \string\makeglossaries[...]\space with
4136           record=off option}%
4137       \else
```

\@gls@@automake@immediate was introduced to glossaries v4.42 so it may not be defined.

```
4138 \ifdef\@gls@@automake@immediate{\@gls@@automake@immediate}{}%
4139 \edef\@glsxtr@reg@glosslist{#1}%
4140 \ifundef\@gls@write{\newwrite\@gls@write}{}%
4141 \protected@write\@auxout{\string\providecommand
4142 \string\@glsorder[1]{}%
4143 \protected@write\@auxout{\string\providecommand
4144 \string\@istfilename[1]{}%
4145 \protected@write\@auxout{\string\@istfilename{\istfilename}}%
4146 \protected@write\@auxout{\string\@glsorder{\glsorder}}%
4147 \protected@write\@auxout{\string\@glsxtr@makeglossaries{#1}}%
4148 \write\@auxout{\string\providecommand\string\@gls@reference[3]{}%}
```

Iterate through each supplied glossary type and activate it.

```
4149 \@for\@glo@type:=#1\do{%
4150 \ifempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4151 }%
```

New glossaries must be created before \makeglossaries:

```
4152 \renewcommand*\newglossary[4][]{%
4153 \PackageError{glossaries}{New glossaries
4154 must be created before \string\makeglossaries}{You need
4155 to move \string\makeglossaries\space after all your
4156 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
4157 \let\@makeglossary\relax
4158 \let\makeglossary\relax
4159 \renewcommand\makeglossaries[1][]{}%
```

Disable all commands that have no effect after \makeglossaries

```
4160 \@disable@onlypremakeg
```

Allow see key:

```
4161 \let\@gls@checkseeallowed\relax
```

Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```
4162 \renewcommand*\@do@seeglossary[2]{%
4163 \glsdoifexists{##1}%
4164 {%
4165 \edef\@gls@label{\glsdetoklabel{##1}}%
4166 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4167 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4168 {\@glsxtr@org@doseeglossary{##1}{##2}}%
4169 {%
4170 \@@glsxtrwrglossmark
4171 \protected@write\@auxout{\%
4172 \string\@gls@reference
4173 {\@gls@type}{\@gls@label}{\string\glsseeformat##2}}%
4174 }%
```

```

4175         }%
4176     }%
4177 }%

Adjust \@do@wrglossary
4178     \let\@glxtr@do@wrglossary\@do@wrglossary
4179     \def\@do@wrglossary{%
4180         \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4181         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
4182         {\@glxtr@do@wrglossary}%
4183         {\gls@noidxglossary}%
4184     }%

Suppress warning about no \makeglossaries
4185     \let\warn@nomakeglossaries\relax
4186     \def\warn@noprintglossary{%
4187         \GlossariesWarningNoLine{No \string\printglossary\space
4188             or \string\printglossaries\space
4189             found.^^J(Remove \string\makeglossaries\space if you don't want
4190             any glossaries.)^^JThis document will not have a glossary}%
4191     }%

Only warn for glossaries not listed.
4192     \renewcommand{\@gls@noref@warn}[1]{%
4193         \edef\@gls@type{##1}%
4194         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
4195         {%
4196             \GlossariesExtraWarning{Can't use
4197                 \string\printnoidxglossary[type={\@gls@type}]
4198                 when '\@gls@type' is listed in the optional argument of
4199                 \string\makeglossaries}%
4200         }%
4201         {%
4202             \GlossariesWarning{Empty glossary for
4203                 \string\printnoidxglossary[type={##1}].
4204                 Rerun may be required (or you may have forgotten to use
4205                 commands like \string\gls)}%
4206         }%
4207     }%

Adjust display number list to check for type:
4208     \renewcommand*\@glsdisplaynumberlist[1]{%
4209         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
4210         {\@glxtr@idx@displaynumberlist{##1}}%
4211         {\@glxtr@noidx@displaynumberlist{##1}}%
4212     }%

Adjust entry list:
4213     \renewcommand*\@glsentrynumberlist[1]{%
4214         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
4215         {\@glxtr@idx@entrynumberlist{##1}}%

```

```

4216      {\@glxtr@noidx@entrynumberlist{##1}}}%
4217    }%

```

Adjust number list loop

```

4218      \renewcommand*{\glsnumberlistloop}[2]{%
4219        \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
4220        {%
4221          \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4222            not available for glossary ‘##1’}{}%
4223        }%
4224        {\@glxtr@noidx@numberlistloop{##1}{##2}}}%
4225      }%

```

Only sanitize sort for normal indexing glossaries.

```

4226      \renewcommand*{\glsprestandardsort}[3]{%
4227        \expandafter\DTLifinlist\expandafter{##2}{\@glxtr@reg@glosslist}%
4228        {%
4229          \glsdosanitizesort
4230        }%
4231        {%
4232          \ifglssanitizesort
4233            \@gls@noidx@sanitizesort
4234          \else
4235            \@gls@noidx@nosanitizesort
4236          \fi
4237        }%
4238      }%

```

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

```

4239      \renewcommand*\new@glossaryentry[2]{%
4240        \PackageError{glossaries-extra}{Glossary entries must be defined
4241          in the preamble\MessageBreak when you use the optional argument
4242          of \string\makeglossaries}{Either move your definitions to the
4243          preamble or don’t use the optional argument of
4244          \string\makeglossaries}%
4245      }%

```

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).

```

4246      \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
4247      \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

4248        \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
4249        type=\glsdefaulttype,\@end@glxtr@gettype
4250        \def\@glo@sorttype{\@glo@default@sorttype}%
4251      }%

```

Check automake setting:

```

4252      \ifglautomake

```

```

4253         \renewcommand*{\@gls@doautomake}{%
4254             \@for\@gls@type:=\@glstr@reg@glosslist\do{%
4255                 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
4256             }%
4257         }%
4258     \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

4259         \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4260     \fi
4261 }%
4262 }%
4263 }

```

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

`\@printglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the `glstex` file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

4264 \newcommand{\@glstr@orgprintglossary}[2]{%
4265     \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

4266     \def\glossarytitle{%
4267         \ifcsdef{\@glo@type@\@glo@type @title}%
4268         {\csuse{\@glo@type@\@glo@type @title}}%
4269         {\glossaryname}}%
4270     \def\glossarytoctitle{\glossarytitle}%
4271     \let\org@glossarytitle\glossarytitle
4272     \def\@glossarystyle{%
4273         \ifx\@glossary@default@style\relax
4274             \GlossariesWarning{No default glossary style provided \MessageBreak
4275                 for the glossary '\@glo@type'. \MessageBreak
4276                 Using deprecated fallback. \MessageBreak
4277                 To fix this set the style with \MessageBreak
4278                 \string\setglossarystyle\space or use the \MessageBreak
4279                 style key=value option}%
4280     \fi
4281 }%
4282     \def\gls@dotoc@title{\gls@set@toc@title{\@glo@type}}%
4283     \let\org@glossaryentrynumbers\glossaryentrynumbers
4284     \bgroup
4285         \@printgloss@setsort
4286         \setkeys{printgloss}{#1}%
4287         \ifx\glossarytitle\org@glossarytitle
4288         \else
4289             \cslet{\@glo@type@\@glo@type @title}{\glossarytitle}%
4290     \fi

```

```

4291 \let\currentglossary\@glo@type
4292 \let\org@glossaryentrynumbers\glossaryentrynumbers
4293 \let\glsnonextpages\@glsnonextpages
4294 \let\glsnextpages\@glsnextpages

4295 \glsextractivatenopost
4296 \gls@dotoc@title
4297 \@glossarystyle
4298 \let\gls@org@glossaryentryfield\glossentry
4299 \let\gls@org@glossarysubentryfield\subglossentry
4300 \renewcommand{\glossentry}[1]{%
4301   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4302   \gls@org@glossaryentryfield{##1}%
4303 }%
4304 \renewcommand{\subglossentry}[2]{%
4305   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4306   \gls@org@glossarysubentryfield{##1}{##2}%
4307 }%
4308 \@gls@preglossaryhook
4309 #2%
4310 \egroup
4311 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4312 \global\let\warn@noprintglossary\relax
4313 }

```

extractivatenopost Change \nopostdesc and \glsextrnopostpunc to behave as they do in the glossary.

```

4314 \newcommand*{\glsextractivatenopost}{%
4315   \let\nopostdesc\@nopostdesc
4316   \let\glsextrnopostpunc\@glsextr@nopostpunc
4317 }

```

glsextrnopostpunc

```

4318 \newrobustcmd*{\glsextrnopostpunc}{%

```

glsextr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

4319 \newcommand{\@glsextr@nopostpunc}{%
4320   \let\@@glsextr@org@postdescription\glspostdescription
4321   \ifglsnopostdot
4322     \renewcommand{\glspostdescription}{%
4323       \glsnopostdottrue
4324       \let\glspostdescription\@@glsextr@org@postdescription
4325       \let\glsextrrestorepostpunc\@glsextr@restore@postpunc
4326       \glsextrpostdescription
4327       \@glsextr@nopostpunc@postdesc}%
4328   \else
4329     \renewcommand{\glspostdescription}{%
4330       \let\glspostdescription\@@glsextr@org@postdescription
4331       \let\glsextrrestorepostpunc\@glsextr@restore@postpunc

```

```

4332 \glxtrpostdescription
4333 \@glxtr@nopostpunc@postdesc}%
4334 \fi
4335 \glsnopostdotfalse
4336 }

```

stpostpunc@postdesc

```

4337 \newcommand*{\@glxtr@nopostpunc@postdesc}{ }

```

restore@postpunc

```

4338 \newcommand*{\@glxtr@restore@postpunc}{%
4339 \def\@glxtr@nopostpunc@postdesc{%
4340 \@glxtr@org@postdescription
4341 \let\@glxtr@nopostpunc@postdesc\@empty
4342 \let\glxtrrestorepostpunc\@empty
4343 }%
4344 }

```

restorepostpunc Does nothing outside of glossary.

```

4345 \newcommand*{\glxtrrestorepostpunc}{ }

```

\@printglossary Redefine.

```

4346 \renewcommand{\@printglossary}[2]{%
4347 \def\@glxtr@printglossopts{#1}%
4348 \@glxtr@orgprintglossary{#1}{#2}%
4349 }

```

Add a key that switches off the entry targets:

```

4350 \define@choicekey{printgloss}{target}
4351 [\@glxtr@printglossval\@glxtr@printglossnr]%
4352 {true,false}[true]%
4353 {%
4354 \ifcase\@glxtr@printglossnr
4355 \def\@glstarget{\glsdohypertarget}%
4356 \else
4357 \let\@glstarget\@secondoftwo
4358 \fi
4359 }

```

hypernameprefix

```

4360 \newcommand{\@glxtrhypernameprefix}{ }

```

New to v1.20:

```

4361 \define@key{printgloss}{targetnameprefix}{%
4362 \renewcommand{\@glxtrhypernameprefix}{#1}%
4363 }

```

```

4364 \define@key{printgloss}{prefix}{%
4365   \renewcommand{\glolinkprefix}{#1}%
4366 }

```

```

4367 \define@key{printgloss}{label}{%
4368   \glstrsetglossarylabel{#1}%
4369 }

```

`etglossarylabel` Set the label for subsequent glossaries. If the label is fixed (that is, doesn't change with each glossary) this will need to be scoped or changed again to prevent duplicate labels.

```

4370 \newcommand{\glstrsetglossarylabel}[1]{%
4371   \renewcommand*{\@glossaryseclabel}{%
4372     \protected@edef\@currentlabelname{\glossarytoctitle}%
4373     \label{#1}%
4374   }%
4375 }

```

`lsdohypertarget` Redefine to insert `\@glstrhypernameprefix` before the target name.

```

4376 \let\@glstr@org@glldohypertarget\glldohypertarget
4377 \renewcommand{\glldohypertarget}[2]{%
4378   \@glstr@org@glldohypertarget{\@glstrhypernameprefix#1}{#2}%
4379 }

```

Update `\@glstarget` to use `\def` instead being assigned with `\let` so that it can pick up the new definition and allow any further redefinitions:

```

4380 \ifx\@glstarget\@glstr@org@glldohypertarget
4381   \def\@glstarget{\glldohypertarget}%
4382 \fi

```

`@makeglossaries` For the benefit of `makeglossaries`

```

4383 \newcommand*{\glstr@makeglossaries}[1]{%

```

`@glstr@gettype` Get just the type.

```

4384 \def\@glstr@gettype#1,type=#2,#3\end@glstr@gettype{%
4385   \def\@glo@type{#2}%
4386 }

```

`@assign@sortkey` Assign the sort key.

```

4387 \newcommand\@glstr@mixed@assign@sortkey[1]{%
4388   \edef\@glo@type{\@glo@type}%
4389   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glstr@reg@glosslist}%
4390   {%
4391     \@glo@no@assign@sortkey{#1}%
4392   }%
4393   {%
4394     \@glo@assign@sortkey{#1}%
4395   }%
4396 }%

```

Display number list for the regular version:

splaynumberlist

```
4397 \let\@glstr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4398 \newcommand*{\@glstr@noidx@displaynumberlist}[1]{%
4399   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4400   \ifdef\@gls@loclist
4401   {%
4402     \def\@gls@noidxloclist@sep{%
4403       \def\@gls@noidxloclist@sep{%
4404         \def\@gls@noidxloclist@sep{%
4405           \glsnumlistsep
4406         }%
4407         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4408       }%
4409     }%
4410     \def\@gls@noidxloclist@finalsep{}%
4411     \def\@gls@noidxloclist@prev{}%
4412     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4413     \@gls@noidxloclist@finalsep
4414     \@gls@noidxloclist@prev
4415   }%
4416   {%
4417     \glstrundeftag
4418     \glsdoifexists{#1}%
4419     {%
4420       \GlossariesWarning{Missing location list for ‘#1’. Either
4421         a rerun is required or you haven’t referenced the entry.}%
4422     }%
4423   }%
4424 }%
4425
```

And for the number list loop:

@numberlistloop

```
4426 \newcommand*{\@glstr@noidx@numberlistloop}[3]{%
4427   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4428   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4429   \let\@gls@org@glsseeformat\glsseeformat
4430   \let\glsnoidxdisplayloc#2\relax
4431   \let\glsseeformat#3\relax
4432   \ifdef\@gls@loclist
4433   {%
4434     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4435   }%
```

```

4436 {%
4437   \glxtrundeftag
4438   \glsdoifexists{#1}%
4439   {%
4440     \GlossariesWarning{Missing location list for ‘##1’. Either
4441       a rerun is required or you haven’t referenced the entry.}%
4442   }%
4443 }%
4444 \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4445 \let\glsseeformat\@gls@org\glsseeformat
4446 }%

```

Same for entry number list.

entrynumberlist

```

4447 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4448   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4449   \ifdef\@gls@loclist
4450   {%
4451     \glsnoidxloclist{\@gls@loclist}%
4452   }%
4453   {%
4454     \glxtrundeftag
4455     \glsdoifexists{#1}%
4456     {%
4457       \GlossariesWarning{Missing location list for ‘#1’. Either
4458         a rerun is required or you haven’t referenced the entry.}%
4459     }%
4460   }%
4461 }%

```

entrynumberlist

```

4462 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\@glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4463 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4464   \protected@edef\@glsxtr@titlelabel{#1}%
4465   \ifdefvoid\@glsxtr@titlelabel
4466   {}%
4467   {%
4468     \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4469   }%
4470   \ifdefvoid{\@glsxtr@titlelabel}%
4471   {%
4472     \DTLifint{#1}%
4473     {%
4474       \ifnum#1<256\relax
4475       \edef#2{\char#1\relax}%

```

```

4476     \else
4477     \edef#2{#1}%
4478     \fi
4479 }%
4480 {%
4481     \ifcsundef{#1groupname}%
4482     {\def#2{#1}}%
4483     {\letcs#2{#1groupname}}%
4484 }%
4485 }%
4486 {%
4487     \let#2\@glsxtr@titlelabel
4488 }%
4489 }

```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```

4490 \let\glsxtr@org@getgrouptitle\@gls@getgrouptitle

```

trgetgrouptitle Provide a user-level command to fetch the group title. The first argument is the group label. The second argument is a control sequence in which to store the title.

```

4491 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4492   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4493   \@onelevel@sanitize\@glsxtr@titlelabel
4494   \ifcsdef{\@glsxtr@titlelabel}
4495   {\letcs{#2}{\@glsxtr@titlelabel}}%
4496   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4497 }
4498 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

4499 \newcommand{\glsxtrsetgrouptitle}[2]{%
4500   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4501   \@onelevel@sanitize\@glsxtr@titlelabel
4502   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4503 }

```

alsetgrouptitle As above put only locally defines the title.

```

4504 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4505   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4506   \@onelevel@sanitize\@glsxtr@titlelabel
4507   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4508 }

```

\glsnavigation Redefine to use new user-level command.

```

4509 \renewcommand*{\glsnavigation}{%
4510   \def\@gls@between{}%
4511   \ifcsundef{\@gls@hypergroup@list@{\@gls@type}}%
4512   {%

```

```

4513 \def\@gls@list{%
4514 }%
4515 {%
4516 \expandafter\let\expandafter\@gls@list
4517 \csname @gls@hypergroup\list@\@glo@type\endcsname
4518 }%
4519 \@for\@gls@tmp:=\@gls@list\do{%
4520 \@gls@between
4521 \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4522 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4523 \let\@gls@between\glshypernavsep
4524 }%
4525 }

```

`@noidx@glossary`

```

4526 \renewcommand*{\@print@noidx@glossary}{%
4527 \ifcsdef{\@gls@ref@\@glo@type}%
4528 {%
4529 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4530 {%
4531 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4532 }%
4533 {%
4534 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4535 }%
4536 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4537 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4538 \def\@gls@currentlettergroup{%
4539 \begin{theglossary}%
4540 \glossaryheader
4541 \glsresetentrylist
4542 \forlistcsloop{\@gls@noidx@do}{\@gls@ref@\@glo@type}%
4543 \end{theglossary}%
4544 \glossarypostamble
4545 }%
4546 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4547 \glsxtrifemptyglossary{\@glo@type}%
4548 }%
4549 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4550 \@gls@noref@warn{\@glo@type}%
4551 }%
4552 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4553 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4554   \setentrycounter[#1]{#2}%
4555   \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4556 }

```

`xtr@display@loc` Patch to check for range formations.

```

4557 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4558   \ifx#1(\relax
4559     \glsxtrdisplaystartloc{#2}{#3}%
4560   \else
4561     \ifx#1)\relax
4562       \glsxtrdisplayendloc{#2}{#3}%
4563     \else
4564       \glsxtrdisplaysingleloc{#1#2}{#3}%
4565     \fi
4566   \fi
4567 }

```

`isplayingleloc` Single location.

```

4568 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4569   \csuse{#1}{#2}%
4570 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

`displaystartloc` Start of a location range.

```

4571 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4572   \edef\glsxtrlocrangefmt{#1}%
4573   \ifx\glsxtrlocrangefmt\empty
4574     \def\glsxtrlocrangefmt{glsnumberformat}%
4575   \fi
4576   \expandafter\glsxtrdisplaysingleloc
4577   \expandafter{\glsxtrlocrangefmt}{#2}%
4578 }

```

`trdisplayendloc` End of a location range.

```

4579 \newcommand*{\glsxtrdisplayendloc}[2]{%
4580   \edef\@glsxtr@tmp{#1}%
4581   \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}}%
4582   \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4583   \else
4584     \GlossariesExtraWarning{Mismatched end location range
4585       (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
4586   \fi
4587   \expandafter\glsxtrdisplayendloohook\expandafter{\@glsxtr@tmp}{#2}%
4588   \expandafter\glsxtrdisplaysingleloc
4589   \expandafter{\glsxtrlocrangefmt}{#2}%
4590   \def\glsxtrlocrangefmt{}%
4591 }

```

`splayendlochook` Allow the user to hook into the end of range command.

```
4592 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```
4593 \newcommand*{\glxtrlocrangefmt}{}%
```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```
4594 \renewcommand*{\setentrycounter}[2][{}]{%
4595   \def\glxtrcounterprefix{#1}%
4596   \ifx\glxtrcounterprefix\@empty
4597     \def\@glo@counterprefix{.}%
4598   \else
4599     \def\@glo@counterprefix{.#1.}%
4600   \fi
4601   \def\glsetentrycounter{#2}%
4602 }
```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```
4603 \def\@gls@removespaces#1 #2\@nil{%
4604   \toks@=\expandafter{\the\toks@#1}%
4605   \ifx\@#2\%
4606     \edef\@glo@tmp{\the\toks@}%
4607     \ifx\@glo@tmp\empty
4608     \else
```

Expand location (just in case `\toks@` is needed for something else).

```
4609     \expandafter\glxtrlocationhyperlink\expandafter
4610     \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4611     \fi
4612   \else
4613     \@gls@ReturnAfterFi{%
4614       \@gls@removespaces#2\@nil
4615     }%
4616   \fi
4617 }
```

`ocationhyperlink` `\glxtrlocationhyperlink{<counter>}{<prefix>}{<location>}`

```
4618 \newcommand*{\glxtrlocationhyperlink}[3]{%
4619   \ifdefined\glxtrsupplocationurl
4620   {%
4621     \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4622   }%
4623   {%
4624     \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
```

```

4625 }%
4626 }

```

supphypernumber

```

4627 \newcommand*{\glxtrsupphypernumber}[1]{%
4628 {%
4629   \glshasattribute{\glscurrententrylabel}{externallocation}%
4630   {%
4631     \def\glxtrsupplocationurl{%
4632       \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4633   }%
4634   {%
4635     \def\glxtrsupplocationurl{%
4636     }%
4637     \glshypernumber{#1}%
4638   }%
4639 }

```

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

@print@glossary

```

4640 \renewcommand{\@print@glossary}{%
4641   \makeatletter
4642   \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4643   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4644   {}%
4645   {\glxtrNoGlossaryWarning{\@glo@type}}%
4646   \ifglxindy
4647     \ifcsundef{@xdy@\@glo@type @language}%
4648     {%
4649       \edef\@do@auxoutstuff{%
4650         \noexpand\AtEndDocument{%
4651           \noexpand\immediate\noexpand\write\@auxout{%
4652             \string\providecommand\string\@xdylanguage[2]{}}%
4653           \noexpand\immediate\noexpand\write\@auxout{%
4654             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4655         }%
4656       }%
4657     }%
4658     {%
4659       \edef\@do@auxoutstuff{%
4660         \noexpand\AtEndDocument{%
4661           \noexpand\immediate\noexpand\write\@auxout{%
4662             \string\providecommand\string\@xdylanguage[2]{}}%
4663           \noexpand\immediate\noexpand\write\@auxout{%
4664             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4665               @language\endcsname}}%
4666         }%

```

```

4667     }%
4668 }%
4669 \@do@auxoutstuff
4670 \edef\@do@auxoutstuff{%
4671     \noexpand\AtEndDocument{%
4672         \noexpand\immediate\noexpand\write\@auxout{%
4673             \string\providecommand\string\@gls@codepage[2]{}}%
4674         \noexpand\immediate\noexpand\write\@auxout{%
4675             \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4676     }%
4677 }%
4678 \@do@auxoutstuff
4679 \fi
4680 \renewcommand*{\@warn@nomakeglossaries}{%
4681     \GlossariesWarningNoLine{\string\makeglossaries\space
4682     hasn't been used,^^Jthe glossaries will not be updated}%
4683 }%
4684 }

```

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

`\GlsWarningHead` Header message.

```

4685 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4686     This document is incomplete. The external file associated with
4687     the glossary '#1' (which should be called \texttt{#2})
4688     hasn't been created.%
4689 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4690 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4691     This has probably happened because there are no entries defined
4692     in this glossary.%
4693 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4694 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4695     If you don't want this glossary,
4696     add \texttt{nomain} to your package option list when you load
4697     \texttt{glossaries-extra.sty}. For example:%
4698 }

```

`\GlsWarningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4699 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4700     Did you forget to use \texttt{type=#1} when you defined your
4701     entries? If you tried to load entries into this glossary with
4702     \texttt{\string\loadglsentries} did you remember to use
4703     \texttt{[#1]} as the optional argument? If you did, check that
4704     the definitions in the file you loaded all had the type set
4705     to \texttt{\string\glsdefaulttype}.%
4706 }

```

arningCheckFile Advisory message to check the file contents.

```
4707 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4708   Check the contents of the file \texttt{#1}. If
4709   it's empty, that means you haven't indexed any of your entries in this
4710   glossary (using commands like \texttt{\string\gls} or
4711   \texttt{\string\glsadd}) so this list can't be generated.
4712   If the file isn't empty, the document build process hasn't been
4713   completed.%
4714 }
```

WarningAutoMake Message when automake option has been used.

```
4715 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4716   You may need to rerun \LaTeX. If you already have, it may be that
4717   \TeX's shell escape doesn't allow you to run
4718   \ifglxindy xindy\else makeindex\fi. Check the
4719   transcript file \texttt{\jobname.log}. If the shell escape is
4720   disabled, try one of the following:
4721
4722   \begin{itemize}
4723     \item Run the external (Lua) application:
4724
4725       \texttt{makeglossaries-lite \string"\jobname\string"}
4726
4727     \item Run the external (Perl) application:
4728
4729       \texttt{makeglossaries \string"\jobname\string"}
4730   \end{itemize}
4731
4732   Then rerun \LaTeX\ on this document.
4733   \GlossariesExtraWarning{Rerun required to build the
4734   glossary '#1' or check TeX's shell escape allows
4735   you to run \ifglxindy xindy\else makeindex\fi}%
4736 }
```

WarningMismatch Mismatching \makenoidxglossaries.

```
4737 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4738   You need to either replace \texttt{\string\makenoidxglossaries}
4739   with \texttt{\string\makeglossaries} or replace
4740   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4741   \texttt{\string\printnoidxglossary}
4742   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4743   this document.%
4744 }
```

arningBuildInfo Build advice.

```
4745 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4746   Try one of the following:
4747   \begin{itemize}
```

```

4748 \item Add \texttt{automake} to your package option list when you load
4749 \texttt{glossaries-extra.sty}. For example:
4750
4751 \texttt{\string\usepackage[automake]%
4752 \glsopenbrace glossaries-extra\glsclosebrace}
4753
4754 \item Run the external (Lua) application:
4755
4756 \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4757
4758 \item Run the external (Perl) application:
4759
4760 \texttt{makeglossaries \string"\jobname\string"}
4761 \end{itemize}
4762
4763 Then rerun \LaTeX\ on this document.%.
4764 }

```

trRecordWarning Paragraph for record=only.

```

4765 \newcommand{\GlsXtrRecordWarning}[1]{%
4766 \texttt{\string\printglossary} doesn't work
4767 with the \texttt{record=only} package option
4768 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4769 instead (or change the package option).%
4770 }

```

oGlsWarningTail Final paragraph.

```

4771 \newcommand{\GlsXtrNoGlsWarningTail}{%
4772 This message will be removed once the problem has been fixed.%
4773 }

```

GlsWarningNoOut No out file created. Build advice.

```

4774 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4775 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4776 \texttt{\string\makeglossaries} or you have used
4777 \texttt{\string\nofiles}. If this is just a draft version of the
4778 document, you can suppress this message using the
4779 \texttt{nomissingglstext} package option.%
4780 }

```

glossarywarning

```

4781 \newcommand*{@\glstr@defaultnoglossarywarning}[1]{%
4782 \glossarysection[\glossarytoctitle]{\glossarytitle}
4783 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4784 \par
4785 \glstrifemptyglossary{#1}%
4786 {%
4787 \GlsXtrNoGlsWarningEmptyStart\space
4788 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par

```

```

4789 \medskip
4790 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4791 \glsopenbrace glossaries-extra\glsclosebrace}
4792 \medskip
4793 }%
4794 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4795 }%
4796 {%
4797 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4798 {%
4799 \GlsXtrNoGlsWarningCheckFile
4800 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4801
4802 \ifglsautomake
4803
4804 \GlsXtrNoGlsWarningAutoMake{#1}
4805
4806 \else
4807
4808 \ifthenelse{\equal{#1}{main}}{%
4809 {%
4810 \GlsXtrNoGlsWarningEmptyMain\par
4811 \medskip
4812 \noindent\texttt{\string\usepackage[nomain]%
4813 \glsopenbrace glossaries-extra\glsclosebrace}
4814 \medskip
4815 }%
4816 }%
4817
4818 \ifdefequal\makeglossaries\@no@makeglossaries
4819 {%
4820 \GlsXtrNoGlsWarningMisMatch
4821 }%
4822 {%
4823 \GlsXtrNoGlsWarningBuildInfo
4824 }%
4825 \fi
4826 }%
4827 {%
4828 \GlsXtrNoGlsWarningNoOut
4829 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4830 }%
4831 }%
4832 \par
4833 \GlsXtrNoGlsWarningTail
4834 }

```

`glossarywarning` Warn about using `\printglossary` with record

```

4835 \newcommand*{\@glsxtr@record@noglossarywarning}[1]{%

```

```

4836 \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4837 with record=only package option\MessageBreak(use
4838 \string\printunsrtglossary[type=#1])\MessageBreak
4839 instead (or change the package option)}%
4840 \glossarysection[\glossarytoctitle]{\glossarytitle}
4841 \GlsXtrRecordWarning{#1}
4842 \GlsXtrNoGlsWarningTail
4843 }

```

Provide some commands to accompany the record option for use with **bib2gls**.

xtrresourcefile Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```

4844 \newcommand*{\glsxtrresourcefile}[2] [] {%

```

The record option can't be set after this command.

```

4845 \disable@keys{glossaries-extra.sty}{record}%
4846 \glsxtr@writefields
4847 \protected@write\@auxout{\glsxtrresourceinit}{\string\glsxtr@resource{#1}{#2}}%
4848 \let\@glsxtr@org@see@noindex\@gl@see@noindex
4849 \let\@gl@see@noindex\relax
4850 \IfFileExists{#2.glstex}%
4851 {%

```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```

4852 \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4853 \makeatletter
4854 \@input{#2.glstex}%
4855 \@bibgls@restoreat

```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```

4856 \glsxtr@check@bibgls@nameref
4857 }%
4858 {%
4859 \GlossariesExtraWarning{No file '#2.glstex'}%
4860 }%
4861 \let\@gl@see@noindex\@glsxtr@org@see@noindex
4862 }
4863 \onlypreamble\glsxtrresourcefile

```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```

4864 \newcommand{\glsxtr@check@bibgls@nameref}{%
4865 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4866 \ifdef\bibglshrefchar
4867 {}%
4868 {%
4869 \GlossariesExtraWarning{record=nameref requires at least
4870 version 1.8 of bib2gls}%

```

```

4871     }%
4872   \fi
4873   \let\@glxtr@check@bibgls@nameref\relax
4874 }

xtrresourceinit  Code used during the protected write operation.
4875 \newcommand*\glxtrresourceinit{}\}

trresourcecount
4876 \newcount\glxtrresourcecount

trLoadResources  Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.
4877 \newcommand*\GlsXtrLoadResources}[1][\]{}{
4878   \ifnum\glxtrresourcecount=0\relax
4879     \glxtrresourcefile[#1]{\jobname}%
4880   \else
4881     \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4882   \fi
4883   \advance\glxtrresourcecount by 1\relax
4884 }

glxtr@resource
4885 \newcommand*\glxtr@resource}[2]{}

\glxtr@fields
4886 \newcommand*\glxtr@fields}[1]{}

xtr@texencoding
4887 \newcommand*\glxtr@texencoding}[1]{}

\glxtr@langtag
4888 \newcommand*\glxtr@langtag}[1]{}

@pluralsuffixes
4889 \newcommand*\glxtr@pluralsuffixes}[4]{}

tr@shortcutsval
4890 \newcommand*\glxtr@shortcutsval}[1]{}

sxtr@linkprefix
4891 \newcommand*\glxtr@linkprefix}[1]{}

xtr@writefields  This information only needs to be written once, so disable it after it's been used.
4892 \newcommand*\glxtr@writefields{}\}

```

```

4893 \protected@write\@auxout{}%
4894   {\string\providecommand*{\string\glsxtr@fields}[1]{}}%
4895 \protected@write\@auxout{}%
4896   {\string\providecommand*{\string\glsxtr@resource}[2]{}}%
4897 \protected@write\@auxout{}%
4898   {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}}%
4899 \protected@write\@auxout{}%
4900   {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}}%
4901 \protected@write\@auxout{}%
4902   {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}}%
4903 \protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%

4904 \protected@write\@auxout{}%
4905   {\string\providecommand*{\string\glsxtr@record}[5]{}}%

4906 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4907   \protected@write\@auxout{}%
4908     {\string\providecommand*{\string\glsxtr@record@nameref}[8]{}}%
4909 \fi

```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by `tracklang`). For multilingual documents, the required locale will have to be indicated in the sort key when using `\glsxtrresourcefile`.

```

4910 \ifdef\CurrentTrackedLanguageTag
4911   {%
4912     \protected@write\@auxout{}{%
4913       \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
4914   }%
4915   {%
4916     \protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
4917       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4918       {\glsxtrabbrvpluralsuffix}}%
4919   \ifdef\inputencodingname
4920     {%
4921       \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
4922     }%
4923   {%

```

If `fontspec` has been loaded, assume UTF-8. (The encoding can be changed with `\XeTeXinputencoding`, but I can't work out how to determine the current encoding.)

```

4924     \@ifpackageloaded{fontspec}%
4925     {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
4926     {}%
4927   }%
4928 \protected@write\@auxout{}{\string\glsxtr@shortcutsval{\@glsxtr@shortcutsval}}%

```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by `bib2gls` when the external option is used.

```

4929 \AtBeginDocument
4930   {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glslinkprefix}}}%
4931 \let\glsxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. This has to be done before the aux file is opened (so package options automake=immediate and automake=true are identical if just bib2gls is used). The double-quotes around \jobname have been removed (v1.19) since \jobname will include double-quotes if the file name has spaces.

```
4932 \ifglsautomake
4933 \IfFileExists{\jobname.aux}%
4934 {\immediate\write18{bib2gls \jobname}}{}}%
```

If \makeglossaries is also used, allow makeindex/xindy to also be run, otherwise disable the error message about requiring \makeglossaries with automake=true.

```
4935 \ifx\@gls@doautomake\@gls@doautomake@err
4936 \let\@gls@doautomake\relax
4937 \fi
4938 \fi
4939 }
```

do@automake@err

```
4940 \newcommand*{\@gls@doautomake@err}{%
4941 \PackageError{glossaries}{You must use
4942 \string\makeglossaries\space with automake=true}
4943 {%
4944 Either remove the automake=true setting or
4945 add \string\makeglossaries\space to your document preamble.%
4946 }%
4947 }
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```
4948 \newcommand*{\glsxtr@record}[5]{}%
```

@record@nameref Used with record=nameref to include current label information.

```
4949 \newcommand*{\glsxtr@record@nameref}[8]{}%
```

r@counterrecord Aux file command.

```
4950 \newcommand*{\glsxtr@counterrecord}[3]{%
4951 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4952 }
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
4953 \newcommand*{\@glsxtr@counterrecordhook}{}%
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4954 \newcommand*{\GlsXtrRecordCounter}[1]{%
4955 \@glsxtr@recordcounter{#1}%
4956 }
4957 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4958 \newcommand*{\@glstr@docounterrecord}[1]{%
4959   \protected@write\@auxout{}\string\glstr@counterrecord
4960     {\@glsl@label}{#1}{\csuse{the#1}}}%
4961 }
```

lsxtrglossentry Users may prefer to have entries displayed throughout the document rather than gathered together in a list. This command emulates the way `\glossentry` behaves (without the style formatting commands like `\item`). This needs to define `\currentglossary` to the current glossary type (normally set at the start of `\@printglossary`) and needs to define `\glscurrententrylabel` to the entry's label (normally set before `\glossentry` and `\subglossentry`). This needs some protection in case it's used in a section heading.

```
4962 \newcommand*{\glstrglossentry}[1]{%
4963   \glstrtitleorpdforheading
4964   {\@glstrglossentry{#1}}%
4965   {\glsentryname{#1}}%
4966   {\glstrheadname{#1}}%
4967 }
```

lsxtrglossentry Another test is needed in case `\@glstrglossentry` has been written to the table of contents.

```
4968 \newrobustcmd*{\@glstrglossentry}[1]{%
4969   \glstrtitleorpdforheading
4970   {%
4971     \glstoifexists{#1}%
4972     {%
4973       \begingroup
4974       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4975       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4976       \ifglshasparent{#1}%
4977       {\GlsXtrStandaloneSubEntryItem{#1}}%
4978       {\glsentryitem{#1}}%
4979       \GlsXtrStandaloneEntryName{#1}%
4980     \endgroup
4981   }%
4982 }%
4983 {\glsentryname{#1}}%
4984 {\glstrheadname{#1}}%
4985 }
```

daloneEntryName

```
4986 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
4987   \glstarget{#1}{\glossentryname{#1}}%
4988 }
```

oneGlossaryType To make it easier to adjust the definition of `\currentglossary` within `\glstrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```
4989 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}
```

oneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```
4990 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4991   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}}%
4992 }
```

glossentryother As \glsxtrglossentry but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```
4993 \newcommand*{\glsxtrglossentryother}[3]{%
4994   \ifstrempy{#1}%
4995   {%
4996     \ifcsdef{glsxtrhead#3}%
4997     {%
4998       \glsxtrtitleorpdforheading
4999       {\@glsxtrglossentryother{#2}{#3}{#1}}%
5000       {\@gls@entry@field{#2}{#3}}%
5001       {\csuse{glsxtrhead#3}{#2}}%
5002     }%
5003     {%
5004       \glsxtrtitleorpdforheading
5005       {\@glsxtrglossentryother{#2}{#3}{#1}}%
5006       {\@gls@entry@field{#2}{#3}}%
5007       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
5008     }%
5009   }%
5010   {%
5011     \glsxtrtitleorpdforheading
5012     {\@glsxtrglossentryother{#2}{#3}{#1}}%
5013     {\@gls@entry@field{#2}{#3}}%
5014     {#1}%
5015   }%
5016 }
```

glossentryother As \@glsxtrglossentry but uses a different field.

```
5017 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
5018   \glsxtrtitleorpdforheading
5019   {%
5020     \glsdoifexists{#1}%
5021     {%
5022       \begingroup
5023       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
5024       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5025       \ifglshasparent{#1}%
5026       {\GlsXtrStandaloneSubEntryItem{#1}}%
5027       {\glsentryitem{#1}}%
5028       \GlsXtrStandaloneEntryOther{#1}%

```

```

5029      \endgroup
5030    }%
5031  }%
5032  {\@gls@entry@field{#1}{#2}}%
5033  {#3}%
5034 }

\standaloneEntryOther
5035 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
5036   \glstarget{#1}{\glossentrynameother{#1}{#2}}%
5037 }

\printunsrtglossary Similar to \printnoidxglossary but it displays all entries defined for the given glossary
                    without sorting.
5038 \newcommand*{\printunsrtglossary}{%
5039   \@ifstar\s@printunsrtglossary\@printunsrtglossary
5040 }

\printunsrtglossary Unstarred version.
5041 \newcommand*{\@printunsrtglossary}[1][ ]{%
5042   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
5043 }

\printunsrtglossary Starred version.
5044 \newcommand*{\s@printunsrtglossary}[2][ ]{%
5045   \begingroup
5046     #2%
5047     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
5048   \endgroup
5049 }

\printunsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary
                     without sorting.
5050 \newcommand*{\printunsrtglossaries}{%
5051   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
5052 }

\@unsrt@glossary
5053 \newcommand*{\@print@unsrt@glossary}{%
5054   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5055   \glossarypreamble
5056   check for empty list
5057   \glstrifemptyglossary{\@glo@type}%
5058   {\%
5059     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
5060   }%
5061 }

```

```

5061 \key@ifundefined{glossentry}{group}%
5062 {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
5063 {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
5064 \def\@gls@currentlettergroup{}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

5065 \def\@glsxtr@doglossary{%
5066   \begin{theglossary}%
5067   \glossaryheader
5068   \glsresetentrylist
5069 }%
5070 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
5071   :\expandafter=\csname glo@list@\@glo@type\endcsname\do{%
5072   \ifdefempty{\glscurrententrylabel}
5073   }%
5074   {%

```

Provide a hook (for example to measure width).

```

5075   \let\glsxtr@process\@firstofone
5076   \let\printunsrtglossaryskipentry
5077   \@glsxtr@printunsrtglossaryskipentry
5078   \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5079   \glsxtr@process
5080   {%
5081     \ifglshasparent{\glscurrententrylabel}{}%
5082     {%
5083       \@glsxtr@checkgroup\glscurrententrylabel
5084       \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
5085         {\@glsxtr@groupheading}%
5086     }%
5087     \eappto\@glsxtr@doglossary{%
5088       \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5089   }%
5090 }%
5091 }%
5092 \appto\@glsxtr@doglossary{\end{theglossary}}%
5093 \printunsrtglossarypredoglossary
5094 \@glsxtr@doglossary
5095 }%
5096 \glossarypostamble
5097 }

```

entryprocesshook

```

5098 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{%

```

ossaryskipentry

```

5099 \newcommand*{\printunsrtglossaryskipentry}{%
5100   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space

```

```
5101 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5102 }
```

entryprocesshook

```
5103 \newcommand*{\@glstr@printunsrtglossaryskipentry}{%
5104   \let\glstr@process\@gobble
5105 }
```

rypredoglossary

```
5106 \newcommand*{\printunsrtglossarypredoglossary}{}%
```

lossary@handler

```
5107 \newcommand{\@printunsrt@glossary@handler}[1]{%
5108   \xdef\glscurrententrylabel{#1}%
5109   \printunsrtglossaryhandler\glscurrententrylabel
5110 }
```

glossaryhandler

```
5111 \newcommand{\printunsrtglossaryhandler}[1]{%
5112   \glstrunsrtdo{#1}%
5113 }
```

striflabelinlist

```
\glstriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of \@gls@ifinlist which ensures the label and list are fully expanded.

```
5114 \newrobustcmd*{\glstriflabelinlist}[4]{%
5115   \protected@edef\@glstr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
5116   \@glstr@doiflabelinlist{#3}{#4}%
5117 }
```

srtglossaryunit

```
5118 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5119   \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
5120     \printunsrtglossaryunitsetup{#2}%
5121   }%
5122 }
```

ossaryunitsetup

```
5123 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
5124   \renewcommand{\printunsrtglossaryhandler}[1]{%
5125     \glstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5126     {\glstrunsrtdo{##1}}%
5127   }%
5128   }%
```

Only the target names should have the prefixes adjusted as \gls etc need the original \glslinkprefix. The \@gobble part discards \glslinkprefix.

```

5129 \ifcsundef{theH#1}%
5130 {%
5131 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
5132 }%
5133 {%
5134 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
5135 }%
5136 \renewcommand*{\glossarysection}[2][{}]{%
5137 \appto\glossarypostamble{\glspar\medskip\glspar}%
5138 }

```

unsrtglossaryunit

```

5139 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
5140 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5141 requires the record=only or record=alsoindex package option}{}%
5142 }

```

t@getgrouptitle

```

5143 \newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
5144 \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
5145 \@onelevel@sanitize\@glsxtr@titlelabel
5146 \ifcsdef{\@glsxtr@titlelabel}
5147 {\letcs{#2}{\@glsxtr@titlelabel}}%
5148 {\def#2{#1}}%
5149 }

```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```

5150 \newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}

```

lsxtrgroupfield bib2gls provides a supplementary field labelled secondarygroup for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```

5151 \newcommand*{\glsxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glsxtr@doglossary is being constructed rather than in the handler.

lsxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glsxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glsxtr@groupheading, which will be empty if no heading is required.

```

5152 \newcommand*{\@glsxtr@checkgroup}[1]{%
5153 \def\@glsxtr@groupheading{}%
5154 \key@ifundefined{glossentry}{group}%
5155 {%

```

```

5156 \letcs{\@gls@sort}{glo@glstdetoklabel{#1}@sort}%
5157 \expandafter\glo@grabfirst\@gls@sort{}\@nil
5158 }%
5159 {%

5160 \protected@edef\@glo@thislettergrp{%
5161 \csuse{glo@glstdetoklabel{#1}@glstxtrgroupfield}}%
5162 }%
5163 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5164 {}%
5165 {%
5166 \ifdefempty{\@gls@currentlettergroup}{}%
5167 {\def\@glstxtr@groupheading{\glsgroupskip}}%
5168 \eappto\@glstxtr@groupheading{%
5169 \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
5170 }%
5171 }%
5172 \let\@gls@currentlettergroup\@glo@thislettergrp
5173 }

```

trLocationField Stores the internal name of the location field.

```
5174 \newcommand*\@GlsXtrLocationField{location}
```

glstxtr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present, but also need to check for the group field.

```

5175 \newcommand{\@glstxtr@noidx@do}[1]{%
5176 \ifglstentryexists{#1}%
5177 {%
5178 \global\letcs{\@gls@loclist}{glo@glstdetoklabel{#1}@loclist}%
5179 \global\letcs{\@gls@location}{glo@glstdetoklabel{#1}@GlsXtrLocationField}%
5180 \ifglshasparent{#1}%
5181 {%
5182 \gls@level=\csuse{glo@glstdetoklabel{#1}@level}\relax
5183 \ifdefvoid{\@gls@location}%
5184 {%
5185 \ifdefvoid{\@gls@loclist}%
5186 {%
5187 \subglossentry{\gls@level}{#1}{}%
5188 }%
5189 {%
5190 \subglossentry{\gls@level}{#1}%
5191 {%
5192 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5193 }%
5194 }%
5195 }%
5196 {%
5197 \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5198 }%

```

```

5199 }%
5200 {%

5201 \ifdefvoid{\@gls@location}%
5202 {%
5203 \ifdefvoid{\@gls@loclist}
5204 {%
5205 \glossentry{#1}{}%
5206 }%
5207 {%
5208 \glossentry{#1}%
5209 {%
5210 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5211 }%
5212 }%
5213 }%
5214 {%
5215 \glossentry{#1}%
5216 {%
5217 \glossaryentrynumbers{\@gls@location}%
5218 }%
5219 }%
5220 }%
5221 }%
5222 {}%
5223 }

```

Provide a way to conveniently define commands that behaves like `\gls` with a label prefix.

It's possible that the user might want minor variations with the same prefix but different default options, so use a counter to provide unique inner commands.

`\glsxtrnewgls`

```

5224 \newcount\@glsxtrnewgls@inner

```

(The default options supplied in *<options>* below could possibly be used to form the inner control sequence name to help make it unique, but it might feasibly contain the value where the value might contain commands.)

`r@providenewgls`

```

5225 \newcommand*{\@glsxtr@providenewgls}{%
5226 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
5227 \let\@glsxtr@providenewgls\relax
5228 }

```

`identifyglslike` Identify the command given in the second argument for the benefit of **bib2gls**.

```

5229 \newcommand{\glsxtridentifyglslike}[2]{%
5230 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
5231 }%
5232 {%

```

```

5233 \@glsxtr@providenewgls
5234 \protected@write\@auxout{}\string\@glsxtr@newglslike{#1}\string#2}}%
5235 }%
5236 }

```

```
\@glsxtrnewgls \glsxtrnewgls[<options>]{<prefix>}{<cs>}{<inner cs name>}
```

```

5237 \newcommand*{\@glsxtrnewgls}[4]{%
5238 \ifdef{#3}%
5239 {%
5240 \PackageError{glossaries-extra}{Command \string#3\space already
5241 defined}{}%
5242 }%
5243 {%

```

Write information to the aux file for bib2gls.

```

5244 \glsxtridentifyglslike{#2}{#3}%
5245 \ifcsdef{@#4like@#2}%
5246 {%
5247 \advance\@glsxtrnewgls@inner by \@ne
5248 \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
5249 }%
5250 {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
5251 \expandafter\newrobustcmd\expandafter*\expandafter
5252 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
5253 \ifstrempy{#1}%
5254 {%
5255 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
5256 \new@ifnextchar[%
5257 {\csname @#4@\endcsname{##1}{#2##2}}%
5258 {\csname @#4@\endcsname{##1}{#2##2}[]}%
5259 }%
5260 }%
5261 {%
5262 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
5263 \new@ifnextchar[%
5264 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5265 {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
5266 }%
5267 }%
5268 }%
5269 }

```

```
\glsxtrnewgls \glsxtrnewgls[<options>]{<prefix>}{<cs>}
```

The first argument prepends to the options and the second argument is the prefix.

```
5270 \newrobustcmd*{\glxstrnewgls}[3] [] {%
5271   \@glxstrnewgls{#1}{#2}{#3}{gls}%
5272 }
```

glxstrnewglslike Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5273 \newrobustcmd*{\glxstrnewglslike}[6] [] {%
5274   \@glxstrnewgls{#1}{#2}{#3}{gls}%
5275   \@glxstrnewgls{#1}{#2}{#4}{glspl}%
5276   \@glxstrnewgls{#1}{#2}{#5}{Gls}%
5277   \@glxstrnewgls{#1}{#2}{#6}{Glspl}%
5278 }
```

glxstrnewGLSlike Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5279 \newrobustcmd*{\glxstrnewGLSlike}[4] [] {%
5280   \@glxstrnewgls{#1}{#2}{#3}{GLS}%
5281   \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
5282 }
```

\glxstrnewrgls As `\glxstrnewgls` but for `\rgls`.

```
5283 \newrobustcmd*{\glxstrnewrgls}[3] [] {%
5284   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5285 }
```

glxstrnewrglslike As `\glxstrnewglslike` but for `\rgls` etc.

```
5286 \newrobustcmd*{\glxstrnewrglslike}[6] [] {%
5287   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5288   \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
5289   \@glxstrnewgls{#1}{#2}{#5}{rGls}%
5290   \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
5291 }
```

glxstrnewrGLSlike As `\glxstrnewGLSlike` but for `\rGLS` etc.

```
5292 \newrobustcmd*{\glxstrnewrGLSlike}[4] [] {%
5293   \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
5294   \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
5295 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
5296 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
5297   \ifcsdef{glo@glsdetoklabel{#1}@recordcount}%
5298   {\csname glo@glsdetoklabel{#1}@recordcount\endcsname}%
5299   {0}%
5300 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
5301 \newcommand*{\GlsXtrRecordCount}[2]{%
5302   \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2}%
5303   {\csname glo@glstetoklabel{#1}@recordcount.#2\endcsname}%
5304   {0}}%
5305 }
```

tionRecordCount Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless `\glstetoklocation` can be set to something sensible.

```
5306 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
5307   \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}}%
5308   {\csname glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}\endcsname}%
5309   {0}}%
5310 }
```

trdetoklocation

```
5311 \newcommand*{\glstetoklocation}[1]{#1}
```

ablerecordcount

```
5312 \newcommand*{\glstrenablerecordcount}{%
5313   \renewcommand*{\gls}{\rgls}%
5314   \renewcommand*{\Gls}{\rGls}%
5315   \renewcommand*{\glspl}{\rglspl}%
5316   \renewcommand*{\Glspl}{\rGlspl}%
5317   \renewcommand*{\GLS}{\rGLS}%
5318   \renewcommand*{\GLSpl}{\rGLSpl}%
5319 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
5320 \newcommand*{\glstxrrecordtriggervalue}[1]{%
5321   \GlsXtrTotalRecordCount{#1}}%
5322 }
```

dCountAttribute

```
5323 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
5324   \@for\@glstxr@cat:=#1\do
5325   {%
5326     \ifdefempty{\@glstxr@cat}{}%
5327     {%
5328       \glsssetcategoryattribute{\@glstxr@cat}{recordcount}{#2}%
5329     }%
5330   }%
5331 }
```

trifrecordtrigger

`\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}`

```
5332 \newcommand*{\glxtrifrecordtrigger}[3]{%
5333   \glshasattribute{#1}{recordcount}%
5334   {%
5335     \ifnum\glxtrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
5336     #3%
5337   \else
5338     #2%
5339   \fi
5340 }%
5341 {#3}%
5342 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
5343 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
5344   \edef\glslabel{\glsdetoklabel{#2}}%
5345   \let\@gls@link@label\glslabel
5346   \def\@glxtr@thevalue{%
5347     \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5348     \def\@glsnumberformat{glstriggerrecordformat}%
5349     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5350     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5351     \def\@glxtr@thevalue{%
5352       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5353       \glxtrinitwrgloss
5354       \glslinkpresetkeys
5355       \setkeys{glslink}{#1}%
5356       \glslinkpostsetkeys
5357       \ifdefempty{\@glxtr@thevalue}%
5358       {%
5359         \@gls@saveentrycounter
5360       }%
5361       {%
5362         \let\thegl Sentrycounter\@glxtr@thevalue
5363         \def\theHgl Sentrycounter{\@glxtr@theHvalue}%
5364       }%
5365       \ifglxtrinitwrglossbefore
5366         \@do@wrglossary{#2}%
5367       \fi
5368       #3%
5369       \ifglxtrinitwrglossbefore
5370       \else
5371         \@do@wrglossary{#2}%
5372       \fi
5373       \ifKV@glslink@local
5374         \glslocalunset{#2}%
5375       \fi
5376     }%
5377   }
```

```

5375 \else
5376 \glsunset{#2}%
5377 \fi
5378 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5379 \newcommand*{\glstriggerrecordformat}[1]{

```

```

\rgls

```

```

5380 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rgls

```

```

5381 \newcommand*{\@rgls}[2][{}]{%
5382 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5383 }

```

```

\@rgls@

```

```

5384 \def\@rgls@#1#2[#3]{%
5385 \glstrifrecordtrigger{#2}%
5386 {%
5387 \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5388 }%
5389 {%
5390 \@gls@{#1}{#2}[#3]%
5391 }%
5392 }%

```

```

\rglspl

```

```

5393 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

```

```

\@rglspl

```

```

5394 \newcommand*{\@rglspl}[2][{}]{%
5395 \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5396 }

```

```

\@rglspl@

```

```

5397 \def\@rglspl@#1#2[#3]{%
5398 \glstrifrecordtrigger{#2}%
5399 {%
5400 \@glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5401 }%
5402 {%
5403 \@glspl@{#1}{#2}[#3]%
5404 }%
5405 }%

```

```

\rgls

```

```

5406 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rGls
5407 \newcommand*{\@rGls}[2][{}]{%
5408   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5409 }

\@rGls@
5410 \def\@rGls@#1#2[#3]{%
5411   \glxtrifrecordtrigger{#2}%
5412   {%
5413     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5414   }%
5415   {%
5416     \@Gls@{#1}{#2}[#3]%
5417   }%
5418 }%

\rGlspl
5419 \newrobustcmd*{\rGlspl}{\@glshyp@opt\@rGlspl}

\@rGlspl
5420 \newcommand*{\@rGlspl}[2][{}]{%
5421   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5422 }

\@rGlspl@
5423 \def\@rGlspl@#1#2[#3]{%
5424   \glxtrifrecordtrigger{#2}%
5425   {%
5426     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5427   }%
5428   {%
5429     \@Glspl@{#1}{#2}[#3]%
5430   }%
5431 }%

\rGLS
5432 \newrobustcmd*{\rGLS}{\@glshyp@opt\@rGLS}

\@rGLS
5433 \newcommand*{\@rGLS}[2][{}]{%
5434   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5435 }

\@rGLS@
5436 \def\@rGLS@#1#2[#3]{%
5437   \glxtrifrecordtrigger{#2}%
5438   {%
5439     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%

```

```

5440 }%
5441 {%
5442   \@GLS@{#1}{#2}[#3]%
5443 }%
5444 }%

```

\rGLSpl

```

5445 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

```

\@rGLSpl

```

5446 \newcommand*{\@rGLSpl}[2][{}]{%
5447   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[{}]}%
5448 }

```

\@rGLSpl@

```

5449 \def\rGLSpl@#1#2[#3]{%
5450   \glsxtrifrecordtrigger{#2}%
5451   {%
5452     \@glsxtr@rglsttrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5453   }%
5454   {%
5455     \@GLSpl@{#1}{#2}[#3]%
5456   }%
5457 }%

```

\rglsformat

```

5458 \newcommand*{\rglsformat}[2]{%
5459   \glsifregular{#1}
5460   {\glsentryfirst{#1}}%
5461   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5462 }

```

\rglsplformat

```

5463 \newcommand*{\rglsplformat}[2]{%
5464   \glsifregular{#1}
5465   {\glsentryfirstplural{#1}}%
5466   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5467 }

```

\rGlsformat

```

5468 \newcommand*{\rGlsformat}[2]{%
5469   \glsifregular{#1}
5470   {\Glsentryfirst{#1}}%
5471   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5472 }

```

\rGlsplformat

```

5473 \newcommand*{\rGlsplformat}[2]{%

```

```

5474 \glsifregular{#1}
5475 {\Glsentryfirstplural{#1}}%
5476 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%2%
5477 }

```

\rGLSformat

```

5478 \newcommand*{\rGLSformat}[2]{%
5479 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5480 }

```

\rGLSplformat

```

5481 \newcommand*{\rGLSplformat}[2]{%
5482 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5483 }

```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into \@gls@link (through \glsxtr@inc@linkcount) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The “link” part of the name refers to \@gls@link not \hyperlink.)

\@inc@linkcount This performs the actual incrementing and counter definition. The counter is given by \c@glxtr@linkcount@<label> where <label> is the entry’s label. Since this is performed within \@gls@link the label can be accessed with \glslabel.

```

5484 \newcommand{\@glsxtr@do@inc@linkcount}{%
    Does this entry have the linkcount attribute set?
5485 \glsifattribute{\glslabel}{linkcount}{true}%
5486 {%
    Does the counter exist?
5487 \ifcsdef{c@glxtr@linkcount@\glslabel}{}%
5488 {%
    Counter doesn't exist, so define it.
5489 \newcounter{glxtr@linkcount@\glslabel}%
    If linkcountmaster is set, add to counter reset.
5490 \glshasattribute{\glslabel}{linkcountmaster}%
5491 {%
    Need to ensure values are fully expanded.
5492 \begingroup
5493 \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@\glslabel}%
5494 {\glsggetattribute{\glslabel}{linkcountmaster}}}%
5495 \x
5496 }%

```

```

5497     {}%
5498   }%

  Increment counter:
5499   \glxstrinclinkcounter{glxstr@linkcount@glslabel}%
5500 }%
5501 {}%
5502 }

rlinkcounter  May be redefined to use \refstepcounter if required.
5503 \newcommand*{\glxstrinclinkcounter}[1]{\stepcounter{#1}}

linkCounterValue  Expands to the associated link counter register or 0 if not defined.
5504 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5505   \ifcsundef{c@glxstr@linkcount@#1}{0}{\csname c@glxstr@linkcount@#1\endcsname}%
5506 }

rTheLinkCounter  Expands to the display value of the associated link counter or 0 if not defined.
5507 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5508   \ifcsundef{theglsxtr@linkcount@#1}{0}%
5509   {\csname theglxstr@linkcount@#1\endcsname}%
5510 }

fLinkCounterDef  Tests if the counter has been defined
5511 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5512   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5513 }

LinkCounterName  Expands to the associated link counter name. (No check for existence.)
5514 \newcommand*{\GlsXtrLinkCounterName}[1]{glxstr@linkcount@#1}

ableLinkCounting  \GlsXtrEnableLinkCounting[master counter]{categories}

  Enable link counting for the given categories.
5515 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5516   \let\glxstr@inc@linkcount\@glxstr@do@inc@linkcount
5517   \@for\@glxstr@label:=#2\do
5518   {%
5519     \glsssetcategoryattribute{\@glxstr@label}{linkcount}{true}%
5520     \ifstrempy{#1}{}%
5521     {%
5522       \ifcsundef{c@#1}%
5523       {\@nocounterr{#1}}%
5524       {\glsssetcategoryattribute{\@glxstr@label}{linkcountmaster}{#1}}%
5525     }%
5526   }%
5527 }
5528 \@onlypreamble\GlsXtrEnableLinkCounting

```

1.5 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.

```
5529 \@ifpackageloaded{glossaries-accsupp}
5530 {
```

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

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

```
5531 \newcommand*{\glsaccessname}[1]{%
5532 \glsnameaccessdisplay
5533 {%
5534 \glstryname{#1}%
5535 }%
5536 {#1}%
5537 }
```

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

```
5538 \newcommand*{\Glsaccessname}[1]{%
5539 \glsnameaccessdisplay
5540 {%
5541 \Glsstryname{#1}%
5542 }%
5543 {#1}%
5544 }
```

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

```
5545 \newcommand*{\GLSaccessname}[1]{%
5546 \glsnameaccessdisplay
5547 {%
5548 \mfirstucMakeUppercase{\glstryname{#1}}%
5549 }%
5550 {#1}%
5551 }
```

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

```
5552 \newcommand*{\glsaccesstext}[1]{%
5553 \glstextaccessdisplay
5554 {%
5555 \glstrytext{#1}%
5556 }%
5557 {#1}%
5558 }
```

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

```

5559 \newcommand*{\Glsaccesstext}[1]{%
5560   \glstextaccessdisplay
5561   {%
5562     \Glsentrytext{#1}%
5563   }%
5564   {#1}%
5565 }
```

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

```

5566 \newcommand*{\GLSaccesstext}[1]{%
5567   \glstextaccessdisplay
5568   {%
5569     \mfirstucMakeUppercase{\Glsentrytext{#1}}%
5570   }%
5571   {#1}%
5572 }
```

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

```

5573 \newcommand*{\glsaccessplural}[1]{%
5574   \glspluralaccessdisplay
5575   {%
5576     \glsentryplural{#1}%
5577   }%
5578   {#1}%
5579 }
```

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

```

5580 \newcommand*{\GLSaccessplural}[1]{%
5581   \glspluralaccessdisplay
5582   {%
5583     \Glsentryplural{#1}%
5584   }%
5585   {#1}%
5586 }
```

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

```

5587 \newcommand*{\GLSaccessplural}[1]{%
5588   \glspluralaccessdisplay
5589   {%
5590     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5591   }%
5592   {#1}%
5593 }
```

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

```

5594 \newcommand*{\glsaccessfirst}[1]{%
5595   \glsfirstaccessdisplay
5596   {%
5597     \glstryfirst{#1}%
5598   }%
5599   {#1}%
5600 }

```

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

```

5601 \newcommand*{\Glsaccessfirst}[1]{%
5602   \glsfirstaccessdisplay
5603   {%
5604     \Glsstryfirst{#1}%
5605   }%
5606   {#1}%
5607 }

```

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

```

5608 \newcommand*{\GLSaccessfirst}[1]{%
5609   \glsfirstaccessdisplay
5610   {%
5611     \mfirstucMakeUppercase{\glstryfirst{#1}}%
5612   }%
5613   {#1}%
5614 }

```

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

```

5615 \newcommand*{\glsaccessfirstplural}[1]{%
5616   \glsfirstpluralaccessdisplay
5617   {%
5618     \glstryfirstplural{#1}%
5619   }%
5620   {#1}%
5621 }

```

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

```

5622 \newcommand*{\Glsaccessfirstplural}[1]{%
5623   \glsfirstpluralaccessdisplay
5624   {%
5625     \Glsstryfirstplural{#1}%
5626   }%
5627   {#1}%
5628 }

```

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

```

5629 \newcommand*{\GLSaccessfirstplural}[1]{%

```

```

5630   \glsfirstpluralaccessdisplay
5631   {%
5632       \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5633   }%
5634   {#1}%
5635   }

```

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

```

5636   \newcommand*{\glsaccesssymbol}[1]{%
5637       \glssymbolaccessdisplay
5638       {%
5639           \glsentrysymbol{#1}%
5640       }%
5641       {#1}%
5642   }

```

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

```

5643   \newcommand*{\Glsaccesssymbol}[1]{%
5644       \glssymbolaccessdisplay
5645       {%
5646           \Glsentrysymbol{#1}%
5647       }%
5648       {#1}%
5649   }

```

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

```

5650   \newcommand*{\GLSaccesssymbol}[1]{%
5651       \glssymbolaccessdisplay
5652       {%
5653           \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5654       }%
5655       {#1}%
5656   }

```

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

```

5657   \newcommand*{\glsaccesssymbolplural}[1]{%
5658       \glssymbolpluralaccessdisplay
5659       {%
5660           \glsentrysymbolplural{#1}%
5661       }%
5662       {#1}%
5663   }

```

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

```

5664   \newcommand*{\Glsaccesssymbolplural}[1]{%
5665       \glssymbolpluralaccessdisplay

```

```

5666   {%
5667       \Glsentrysymbolplural{#1}%
5668   }%
5669   {#1}%
5670 }

```

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

```

5671 \newcommand*{\GLSaccesssymbolplural}[1]{%
5672     \glssymbolpluralaccessdisplay
5673     {%
5674         \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5675     }%
5676     {#1}%
5677 }

```

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

```

5678 \newcommand*{\glsaccessdesc}[1]{%
5679     \glsdescriptionaccessdisplay
5680     {%
5681         \glsentrydesc{#1}%
5682     }%
5683     {#1}%
5684 }

```

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

```

5685 \newcommand*{\Glsaccessdesc}[1]{%
5686     \glsdescriptionaccessdisplay
5687     {%
5688         \Glsentrydesc{#1}%
5689     }%
5690     {#1}%
5691 }

```

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

```

5692 \newcommand*{\GLSaccessdesc}[1]{%
5693     \glsdescriptionaccessdisplay
5694     {%
5695         \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5696     }%
5697     {#1}%
5698 }

```

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

```

5699 \newcommand*{\glsaccessdescplural}[1]{%
5700     \glsdescriptionpluralaccessdisplay
5701     {%
5702         \glsentrydescplural{#1}%

```

```

5703     }%
5704     {#1}%
5705 }

```

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

```

5706 \newcommand*\Glsaccessdescplural}[1]{%
5707   \glsdescriptionpluralaccessdisplay
5708   {%
5709     \Glsentrydescplural{#1}%
5710   }%
5711   {#1}%
5712 }

```

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

```

5713 \newcommand*\GLSaccessdescplural}[1]{%
5714   \glsdescriptionpluralaccessdisplay
5715   {%
5716     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5717   }%
5718   {#1}%
5719 }

```

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

```

5720 \newcommand*\glsaccessshort}[1]{%
5721   \glsshortaccessdisplay
5722   {%
5723     \glsentryshort{#1}%
5724   }%
5725   {#1}%
5726 }

```

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

```

5727 \newcommand*\Glsaccessshort}[1]{%
5728   \glsshortaccessdisplay
5729   {%
5730     \Glsentryshort{#1}%
5731   }%
5732   {#1}%
5733 }

```

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

```

5734 \newcommand*\GLSAccessshort}[1]{%
5735   \glsshortaccessdisplay
5736   {%
5737     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5738   }%

```

```

5739     {#1}%
5740 }

```

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

```

5741 \newcommand*{\lsaccessshortpl}[1]{%
5742   \glshortpluralaccessdisplay
5743   {%
5744     \glentryshortpl{#1}%
5745   }%
5746   {#1}%
5747 }

```

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

```

5748 \newcommand*{\lSaccessshortpl}[1]{%
5749   \glshortpluralaccessdisplay
5750   {%
5751     \Glentryshortpl{#1}%
5752   }%
5753   {#1}%
5754 }

```

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

```

5755 \newcommand*{\LSaccessshortpl}[1]{%
5756   \glshortpluralaccessdisplay
5757   {%
5758     \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5759   }%
5760   {#1}%
5761 }

```

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

```

5762 \newcommand*{\glaccesslong}[1]{%
5763   \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5764 }

```

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

```

5765
5766 \newcommand*{\Glaccesslong}[1]{%
5767   \glslongaccessdisplay{\Glentrylong{#1}}{#1}%
5768 }

```

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

```

5769 \newcommand*{\Glsaccesslong}[1]{%
5770   \glslongaccessdisplay
5771   {%
5772     \mfirstucMakeUppercase{\glentrylong{#1}}%
5773   }%

```

```

5774     {#1}%
5775 }

```

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

```

5776 \newcommand*{\glsaccesslongpl}[1]{%
5777   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5778 }

```

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

```

5779
5780 \newcommand*{\Glsaccesslongpl}[1]{%
5781   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5782 }

```

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

```

5783 \newcommand*{\GLSaccesslongpl}[1]{%
5784   \glslongpluralaccessdisplay
5785   {%
5786     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5787   }%
5788   {#1}%
5789 }

```

Keys for accessibility support.

```

5790 \define@key{glsxtrabbrv}{access}{%
5791   \def\@gls@nameaccess{#1}%
5792 }
5793 \define@key{glsxtrabbrv}{textaccess}{%
5794   \def\@gls@textaccess{#1}%
5795 }
5796 \define@key{glsxtrabbrv}{firstaccess}{%
5797   \def\@gls@firstaccess{#1}%
5798 }
5799 \define@key{glsxtrabbrv}{shortaccess}{%
5800   \def\@gls@shortaccess{#1}%
5801 }
5802 \define@key{glsxtrabbrv}{shortpluralaccess}{%
5803   \def\@gls@shortaccesspl{#1}%
5804 }

```

@initaccesskeys

```

5805 \newcommand*{\@gls@initaccesskeys}{%
5806   \def\@gls@nameaccess{}%
5807   \def\@gls@textaccess{}%
5808   \def\@gls@firstaccess{}%
5809   \def\@gls@shortaccess{}%
5810   \def\@gls@shortaccesspl{}%
5811 }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```
5812 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5813   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5814   {#2}%
5815   {%
5816     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5817     {#3}%
5818     {%
5819       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5820       {#2}%
5821       {#3}%
5822     }%
5823   }%
5824 }
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5825 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
5826 \ifdefempty\@gls@shortaccess
5827 {%
5828   \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5829   {%
5830     \@glsxtr@insertdots\@gls@shortaccess{#1}%
5831     \eappto\ExtraCustomAbbreviationFields{%
5832       shortaccess={\expandonce\@gls@shortaccess},}%
5833     }%
5834   }%
5835 }%
5836 {}%
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5837 \ifdefempty\@gls@shortaccess
5838 {}%
5839 {%
5840   \ifdefempty\@gls@shortaccesspl
5841   {%
5842     \@gls@ifaccessattribute@set{aposplural}%
5843     {%
5844       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5845         \@gls@shortaccess'\abbrvpluralsuffix}%
5846     }%
5847   }%
5848   \@gls@ifaccessattribute@set{noshortplural}%
5849   {%
```

```

5850         \let\@gls@shortaccesspl\@gls@shortaccess
5851     }%
5852     {%
5853         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5854             \@gls@shortaccess\abbrvpluralsuffix}%
5855     }%
5856 }%
5857 \eappto\ExtraCustomAbbreviationFields{%
5858     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5859 }%
5860 {}%
5861 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5862 \ifdefempty\@gls@nameaccess
5863 {%
5864     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5865     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5866         \ifdefempty\@gls@shortaccess
5867         {}%
5868         {%
5869             \eappto\ExtraCustomAbbreviationFields{%
5870                 access={\expandonce\@gls@shortaccess},%
5871             }%
5872         }%
5873     }%
5874     {}%
5875 }%
5876 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5877 \ifdefempty\@gls@textaccess
5878 {%
5879     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5880     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5881         \ifdefempty\@gls@shortaccess
5882         {}%
5883         {%
5884             \eappto\ExtraCustomAbbreviationFields{%
5885                 textaccess={\expandonce\@gls@shortaccess},%
5886             }%
5887         }%
5888     }%
5889     {}%
5890 }%
5891 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5892 \ifdefempty\@gls@firstaccess
5893 {%
5894 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5895 {%
```

Do nothing if the shortaccess key hasn't been set.

```
5896 \ifdefempty\@gls@shortaccess
5897 {}%
5898 {%
5899 \eappto\ExtraCustomAbbreviationFields{%
5900 firstaccess={\expandonce\@gls@shortaccess},%
5901 }%
5902 }%
5903 }%
5904 {}%
5905 }%
5906 {}%
5907 }
```

End of if accsupp part

```
5908 }
5909 {
```

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

```
\glsaccessname Display the name value (no link and no check for existence).
5910 \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.
5911 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
5912 \newcommand*{\GLSaccessname}[1]{%
5913 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}}

\glsaccesstext Display the text value (no link and no check for existence).
5914 \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.
5915 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
5916 \newcommand*{\GLSaccesstext}[1]{%
5917 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}}

glsaccessplural Display the plural value (no link and no check for existence).
5918 \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.
5919 \newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}

GLSaccessplural Display the plural value (no link and no check for existence). converted to upper case.
5920 \newcommand*{\GLSaccessplural}[1]{%
5921 \protect\mfirstucMakeUppercase{\glentryplural{#1}}}

\glsaccessfirst Display the first value (no link and no check for existence).
5922 \newcommand*{\glsaccessfirst}[1]{\glentryfirst{#1}}

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.
5923 \newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}

\GLSaccessfirst Display the first value (no link and no check for existence). converted to upper case.
5924 \newcommand*{\GLSaccessfirst}[1]{%
5925 \protect\mfirstucMakeUppercase{\glentryfirst{#1}}}

cessfirstplural Display the firstplural value (no link and no check for existence).
5926 \newcommand*{\glsaccessfirstplural}[1]{\glentryfirstplural{#1}}

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5927 \newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}

cessfirstplural Display the firstplural value (no link and no check for existence). converted to upper case.
5928 \newcommand*{\GLSaccessfirstplural}[1]{%
5929 \protect\mfirstucMakeUppercase{\glentryfirstplural{#1}}}

glsaccesssymbol Display the symbol value (no link and no check for existence).
5930 \newcommand*{\glsaccesssymbol}[1]{\glentrysymbol{#1}}

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.
5931 \newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}

GLSaccesssymbol Display the symbol value (no link and no check for existence). converted to upper case.
5932 \newcommand*{\GLSaccesssymbol}[1]{%
5933 \protect\mfirstucMakeUppercase{\glentrysymbol{#1}}}

esssymbolplural Display the symbolplural value (no link and no check for existence).
5934 \newcommand*{\glsaccesssymbolplural}[1]{\glentrysymbolplural{#1}}

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
5935 \newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}

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

```
5936 \newcommand*{\GLSaccesssymbolplural}[1]{%
5937 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

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

```
5938 \newcommand*{\glsaccessdesc}[1]{\glentrydesc{#1}}
```

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

```
5939 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

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

```
5940 \newcommand*{\GLSaccessdesc}[1]{%
5941 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

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

```
5942 \newcommand*{\glsaccessdescplural}[1]{\glentrydescplural{#1}}
```

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

```
5943 \newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

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

```
5944 \newcommand*{\GLSaccessdescplural}[1]{%
5945 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

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

```
5946 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

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

```
5947 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

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

```
5948 \newcommand*{\GLSaccessshort}[1]{%
5949 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

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

```
5950 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

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

```
5951 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

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

```
5952 \newcommand*{\GLSaccessshortpl}[1]{%
5953 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

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

```
5954 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

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

```
5955 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

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

```
5956 \newcommand*{\GLSaccesslong}[1]{%
5957 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

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

```
5958 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

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

```
5959 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

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

```
5960 \newcommand*{\GLSaccesslongpl}[1]{%
5961 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
5962 \newcommand*{@gls@initaccesskeys}{}%
```

lt@short@access This does nothing if there's no accessibility support.

```
5963 \newcommand{\@gls@setup@default@short@access}[1]{%
End of else part
5964 }
```

1.6 Categories

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

```
5965 \glsaddstoragekey{category}{general}{\glscategory}
```

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

```
5966 \newcommand{\glsifcategory}[4]{%
5967 \ifglsfield{#1}{category}{#2}{#3}{#4}%
5968 }
```

Categories can have attributes.

categoryattribute

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

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

```
5969 \newcommand*{\glissetcategoryattribute}[3]{%
5970   \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%
5971 }
```

categoryattribute

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

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

```
5972 \newcommand*{\glsggetcategoryattribute}[2]{%
5973   \csuse{@glsxtr@categoryattr@@#1@#2}%
5974 }
```

categoryattribute

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

Tests if the category has the given attribute set.

```
5975 \newcommand*{\glshascategoryattribute}[4]{%
5976   \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%
5977 }
```

\glissetattribute

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

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

```
5978 \newcommand*{\glissetattribute}[3]{%
5979   \glissetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5980 }
```

\glsggetattribute

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

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

```
5981 \newcommand*{\glsggetattribute}[2]{%
5982   \glsggetcategoryattribute{\glscategory{#1}}{#2}%
5983 }
```

\glshasattribute

`\glshasattribute{<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.

```
5984 \newcommand*{\glshasattribute}[4]{%
5985   \ifglentryexists{#1}%
5986   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5987   {#4}%
5988 }
```

categoryattribute

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

True if category has the attribute with the given value.

```
5989 \newcommand{\glcifcategoryattribute}[5]{%
5990   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5991   {#5}%
5992   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5993 }
```

\glcifattribute

`\glcifattribute{<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.

```
5994 \newcommand{\glcifattribute}[5]{%
5995   \ifglentryexists{#1}%
5996   {\glcifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5997   {#5}%
5998 }
```

Set attributes for the default general category:

```
5999 \glsetcategoryattribute{general}{regular}{true}
```

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

```
6000 \glsetcategoryattribute{acronym}{regular}{true}
```

regularcategory

Convenient shortcut to add the regular attribute.

```
6001 \newcommand*{\glsetregularcategory}[1]{%
6002   \glsetcategoryattribute{#1}{regular}{true}%
6003 }
```

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

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

```
6004 \newcommand{\glsifregularcategory}[3]{%
6005   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
6006 }
```

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

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

```
6007 \newcommand{\glsifnotregularcategory}[3]{%
6008   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
6009 }
```

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

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

```
6010 \newcommand{\glsifregular}[3]{%
6011   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
6012 }
```

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

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

```
6013 \newcommand{\glsifnotregular}[3]{%
6014   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
6015 }
```

`\foreachincategory` `\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.

```
6016 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```

6017 \forallglossaries[#1]{#3}%
6018 {%
6019     \forallsentries[#3]{#4}%
6020     {%
6021         \glsifcategory{#4}{#2}{#5}{}%
6022     }%
6023 }%
6024 }

```

achwithattribute \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.

```

6025 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
6026     \forallglossaries[#1]{#4}%
6027     {%
6028         \forallsentries[#4]{#5}%
6029         {%
6030             \glsifattribute{#5}{#2}{#3}{#6}{}%
6031         }%
6032     }%
6033 }

```

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

```

6034 \ifdef\newterm
6035 {%

```

`\newterm`

```

6036     \renewcommand*{\newterm}[2][ ]{%
6037         \newglossaryentry{#2}%
6038         {type=index,category=index,name={#2},%
6039          description={\glstrpostdescription\nopostdesc},#1}%
6040     }

```

Indexed terms are regular by default.

```

6041     \glssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

6042     \newcommand*{\glstrpostdescindex}{}

6043 }
6044 {}

```

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.

```
6045 \ifdef\printsymbols
6046 {%
```

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

```
6047 \newcommand*{\glsxtrnewsymbol}[3] [] {%
6048 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
6049 }
```

Symbols are regular by default.

```
6050 \glssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
6051 \newcommand*{\glsxtrpostdescsymbol}{}
6052 }
6053 {%
```

Similar for the numbers option.

```
6054 \ifdef\printnumbers
6055 {%
```

`glsxtrnewnumber`

```
6056 \ifdef\printnumbers
6057 \newcommand*{\glsxtrnewnumber}[3] [] {%
6058 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
6059 }
```

Numbers are regular by default.

```
6060 \glssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
6061 \newcommand*{\glsxtrpostdescnumber}{}
6062 }
6063 {%
```

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

```
6064 \newcommand*{\glsxtrsetcategory}[2] {%
6065 \@for\@glsxtr@label:=#1\do
6066 {%
6067 \glsfieldxddef{\@glsxtr@label}{category}{#2}%
6068 }%
6069 }
```

`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.

```
6070 \newcommand*{\glxtrsetcategoryforall}[2]{%
6071   \forallglossaries[#1]{\@glxtr@type}{%
6072     \forallglsentries[\@glxtr@type]{\@glxtr@label}%
6073     {%
6074       \glsfieldxdef{\@glxtr@label}{category}{#2}%
6075     }%
6076   }%
6077 }
```

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

Apply title casing to the contents of the given field.

```
6078 \newcommand*{\glxtrfieldtitlecase}[2]{%
6079   \expandafter\glxtrfieldtitlecasesecs\expandafter
6080   {\csname glo\glsdetoklabel{#1}@#2\endcsname}%
6081 }
```

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

```
6082 \newcommand*{\glxtrfieldtitlecasesecs}[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.

```
6083 \ifpackageloaded{glossaries-accsupp}
6084 {
6085   \renewcommand*{\glossentrydesc}[1]{%
6086     \glsdoifexistsorwarn{#1}%
6087     {%
6088       \glssetabbrvfmt{\glscategory{#1}}%
```

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

```
6089   \glshasattribute{#1}{glossdescfont}%
6090   {%
6091     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6092     \ifcsdef{\@glxtr@attrval}%
6093     {%
6094       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6095     }%
6096     {%
6097       \GlossariesExtraWarning{Unknown control sequence name
6098         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
```

```

6099         for entry '#1'. Ignoring}%
6100         \let\@glxtr@glossdescfont\@firstofone
6101     }%
6102 }%
6103 {\let\@glxtr@glossdescfont\@firstofone}%
6104 \glsifattribute{#1}{glossdesc}{firstuc}%
6105 {%
6106     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6107 }%
6108 {%
6109     \glsifattribute{#1}{glossdesc}{title}%
6110     {%
6111         \@glxtr@do@titlecaps@warn
6112         \glsdescriptionaccessdisplay
6113         {%
6114             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6115         }%
6116         {#1}%
6117     }%
6118     {%
6119         \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6120     }%
6121 }%
6122 }%
6123 }
6124 }
6125 {
6126 \renewcommand*{\glossentrydesc}[1]{%
6127     \glsdoifexistsorwarn{#1}%
6128     {%
6129         \glssetabbrvfmt{\glscategory{#1}}%
6130         \glsattribute{#1}{glossdescfont}%
6131         {%
6132             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6133             \ifcsdef{\@glxtr@attrval}%
6134             {%
6135                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6136             }%
6137             {%
6138                 \GlossariesExtraWarning{Unknown control sequence name
6139                     '\@glxtr@attrval' supplied in glossdescfont attribute
6140                     for entry '#1'. Ignoring}%
6141                 \let\@glxtr@glossdescfont\@firstofone
6142             }%
6143         }%
6144         {\let\@glxtr@glossdescfont\@firstofone}%
6145         \glsifattribute{#1}{glossdesc}{firstuc}%
6146         {%
6147             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%

```

```

6148     }%
6149     {%
6150         \glsifattribute{#1}{glossdesc}{title}%
6151         {%
6152             \@glsxtr@do@titlecaps@warn
6153             \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
6154         }%
6155         {%
6156             \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
6157         }%
6158     }%
6159 }%
6160 }
6161 }

```

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

```

6162 \ifpackageloaded{glossaries-accsupp}
6163 {
6164   \renewcommand*{\glossentryname}[1]{%
6165     \@glsdoifexistsorwarn{#1}%
6166     {%
6167       \glssetabbrvfmt{\glscategory{#1}}%

```

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

```

6168   \glsifattribute{#1}{glossnamefont}%
6169   {%
6170     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6171     \ifcsdef{\@glsxtr@attrval}%
6172     {%
6173       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6174     }%
6175     {%
6176       \GlossariesExtraWarning{Unknown control sequence name
6177         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
6178         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6179       \let\@glsxtr@glossnamefont\glsnamefont
6180     }%
6181   }%
6182   {\let\@glsxtr@glossnamefont\glsnamefont}%
6183   \glsifattribute{#1}{glossname}{firstuc}%
6184   {%
6185     \glsnameaccessdisplay
6186     {%
6187       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6188     }%
6189     {#1}%
6190   }%
6191   {%
6192     \glsifattribute{#1}{glossname}{title}%

```

```

6193      {%
6194      \@glstr@do@titlecaps@warn
6195      \glsnameaccessdisplay
6196      {%
6197      \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}}%
6198      }%
6199      {#1}%
6200      }%
6201      {%
6202      \glsifattribute{#1}{glossname}{uc}%
6203      {%
6204      \glsnameaccessdisplay
6205      {%

```

Hide the label from the upper-casing command.

```

6206      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6207      \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6208      }%
6209      {#1}%
6210      }%
6211      {%
6212      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6213      \glsnameaccessdisplay
6214      {%
6215      \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6216      }%
6217      {#1}%
6218      }%
6219      }%
6220      }%

```

Do post-name hook:

```

6221      \glstrpostnamehook{#1}%
6222      }%
6223  }
6224 }
6225 {
6226 \renewcommand*{\glossentryname}[1]{%
6227   \@glsdofexistsorwarn{#1}%
6228   {%
6229     \glsetabbrvfmt{\glscategory{#1}}%
6230     \glshasattribute{#1}{glossnamefont}%
6231     {%
6232       \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
6233       \ifcsdef{\@glstr@attrval}%
6234       {%
6235         \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6236       }%
6237       {%
6238         \GlossariesExtraWarning{Unknown control sequence name

```

```

6239         ‘\@glxtr@attrval’ supplied in glossnamefont attribute
6240         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6241         \let\@glxtr@glossnamefont\glsnamefont
6242     }%
6243 }%
6244 {\let\@glxtr@glossnamefont\glsnamefont}%
6245 \glsifattribute{#1}{glossname}{firstuc}%
6246 {%
6247     \@glxtr@glossnamefont{\Glsentryname{#1}}%
6248 }%
6249 {%
6250     \glsifattribute{#1}{glossname}{title}%
6251     {%
6252         \@glxtr@do@titlecaps@warn
6253         \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
6254     }%
6255     {%
6256         \glsifattribute{#1}{glossname}{uc}%
6257         {%

```

Hide the label from the upper-casing command.

```

6258         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
6259         \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6260     }%
6261     {%

```

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

```

6262         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
6263         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
6264     }%
6265 }%
6266 }%

```

Do post-name hook.

```

6267     \glxtrpostnamehook{#1}%
6268 }%
6269 }
6270 }

```

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

```

6271 \@ifpackageloaded{glossaries-accsupp}
6272 {
6273     \renewcommand*{\Glossentryname}[1]{%
6274         \@glstdoifexistsorwarn{#1}%
6275         {%
6276             \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

6277         \glshasattribute{#1}{glossnamefont}%
6278         {%

```

```

6279     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6280     \ifcsdef{\@glsxtr@attrval}%
6281     {%
6282         \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6283     }%
6284     {%
6285         \GlossariesExtraWarning{Unknown control sequence name
6286             '\@glsxtr@attrval' supplied in glossnamefont attribute
6287             for entry '#1'. Reverting to default \string\glsnamefont}%
6288         \let\@glsxtr@glossnamefont\glsnamefont
6289     }%
6290 }%
6291 {\let\@glsxtr@glossnamefont\glsnamefont}%
6292 \glsnameaccessdisplay
6293 {%
6294     \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6295 }%
6296 {#1}%

```

Do post-name hook:

```

6297     \glsxtrpostnamehook{#1}%
6298 }%
6299 }
6300 }
6301 {
6302     \renewcommand*{\Glossentryname}[1]{%
6303         \@glsdoifexistsorwarn{#1}%
6304         {%
6305             \glssetabbrvfmt{\glscategory{#1}}%
6306             \glsattribute{#1}{glossnamefont}%
6307             {%
6308                 \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6309                 \ifcsdef{\@glsxtr@attrval}%
6310                 {%
6311                     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6312                 }%
6313                 {%
6314                     \GlossariesExtraWarning{Unknown control sequence name
6315                         '\@glsxtr@attrval' supplied in glossnamefont attribute
6316                         for entry '#1'. Reverting to default \string\glsnamefont}%
6317                     \let\@glsxtr@glossnamefont\glsnamefont
6318                 }%
6319             }%
6320             {\let\@glsxtr@glossnamefont\glsnamefont}%
6321             \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6322     \glsxtrpostnamehook{#1}%
6323 }%
6324 }

```

6325 }

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.

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

```
6326 \newcommand*{\glstrpostnamehook}[1]{%
6327   \let\@glsnumberformat\@glstr@defaultnumberformat
6328   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
6329   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
6330   \csuse{glstrpostname\glscategory{#1}}%
6331 }
```

`\glsextrapostnamehook`

```
6332 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
6333 \newcommand*{\glsdefpostname}[2]{%
6334   \csdef{glstrpostname#1}{#2}%
6335 }
```

`\glsaccessdisplay`

```
6336 \@ifpackageloaded{glossaries-accsupp}
6337 {
6338   \newcommand*{\glstr@setaccessdisplay}[1]{%
6339     \ifcsdef{gls#1accessdisplay}%
6340       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
6341     {%
```

This is essentially the reverse of `\@gls@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gls<field>accessdisplay` commands use the key name.

```
6342     \edef\@gls@thisval{#1}%
6343     \@for\@gls@map:=\@gls@keymap\do{%
6344       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6345       \ifdefequal{\@this@key}{\@gls@thisval}%
6346       {%
6347         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6348         \@endfortrue
6349       }%
6350     }%
6351   }%
6352   \ifcsdef{gls\@gls@thisval accessdisplay}%
6353   {\letcs\@glstr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```

6354      {\let\@glxtr@accessdisplay\@firstoftwo}%
6355    }%
6356  }
6357 }
6358 {%
6359 \newcommand*{\glxtr@setaccessdisplay}[1]{%
6360   \let\@glxtr@accessdisplay\@firstoftwo}
6361 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6362 \newrobustcmd*{\glossentrynameother}[2]{%
6363   \@glsdoifexistsorwarn{#1}%
6364   {%

```

Accessibility support:

```

6365   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6366   \glssetabbrvfmt{\glscategory{#1}}%
6367   \glshasattribute{#1}{glossnamefont}%
6368   {%
6369     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6370     \ifcsdef{\@glxtr@attrval}%
6371     {%
6372       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6373     }%
6374     {%
6375       \GlossariesExtraWarning{Unknown control sequence name
6376         '\@glxtr@attrval' supplied in glossnamefont attribute
6377         for entry '#1'. Reverting to default \string\glsnamefont}%
6378       \let\@glxtr@glossnamefont\glsnamefont
6379     }%
6380   }%
6381   {\let\@glxtr@glossnamefont\glsnamefont}%
6382   \glsifattribute{#1}{glossname}{firstuc}%
6383   {%
6384     \@glxtr@accessdisplay
6385     {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6386     {#1}%
6387   }%
6388   {%
6389     \glsifattribute{#1}{glossname}{title}%
6390     {%
6391       \@glxtr@do@titlecaps@warn
6392       \@glxtr@accessdisplay
6393       {\@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{#2}}}%
6394       {#1}%
6395     }%
6396     {%

```

```

6397      \glsifattribute{#1}{glossname}{uc}%
6398      {%
6399      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6400      \@glsxtr@accessdisplay
6401      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6402      {#1}%
6403      }%
6404      {%
6405      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6406      \@glsxtr@accessdisplay
6407      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6408      {#1}%
6409      }%
6410      }%
6411      }%

```

Do post-name hook.

```

6412      \glsxtrpostnamehook{#1}%
6413      }%
6414 }

```

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

```

6415 \newif\if@glsxtr@format@override
6416 \@glsxtr@format@overridefalse

```

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

`xFormatOverride`

```

6417 \@ifpackageloaded{hyperref}
6418 {

```

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

```

6419   \ifHy@hyperindex
6420     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6421       \@glsxtr@format@overridetrue
6422       \appto\theindex{\let\glshypernumber\@firstofone}%
6423     }
6424   \else
6425     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6426       \@glsxtr@format@overridetrue
6427       \appto\theindex{\let\glshypernumber\hyperpage}%
6428     }
6429   \fi
6430 }
6431 {
6432   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6433     \@glsxtr@format@overridetrue
6434   }

```

```

6435 }
6436 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6437 \newcommand*{\glstrdoautoindexname}[2]{%
6438   \glshasattribute{#1}{#2}%
6439   {%

```

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.

```

6440   \@glstr@autoindex@setname{#1}%

```

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

```

6441   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6442   \if@glstr@format@override

6443     \ifx\@glstrnumberformat\@glstr@defaultnumberformat
6444     \else
6445       \let\@glstr@attrval\@glstrnumberformat
6446     \fi
6447   \fi
6448   \ifdefstring{\@glstr@attrval}{true}%
6449   {}%
6450   {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6451   \expandafter\glstrautoindex\expandafter{\@glo@name}%
6452 }%
6453 {}%
6454 }

```

glstrautoindex

```

6455 \newcommand*{\glstrautoindex}{\index}

```

xtrautoindexesc

```

6456 \newcommand{\glstrautoindexesc}{%
6457   \@gls@checkmkidxchars\@glo@sort
6458   \@glstr@autoindex@doextra@esc\@glo@sort
6459 }

```

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

```

6460 \newcommand*{\@glstr@autoindex@setname}[1]{%
6461   \protected@edef\@glo@name{\glstrautoindexentry{#1}}%
6462   \glstrautoindexassignsort{\@glo@sort}{#1}%
6463   \glstrautoindexesc
6464   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
6465 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6466 \newcommand*{\glstrautoindexentry}[1]{\string\glstryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```
6467 \newcommand*{\glxtrautoindexassignsort}[2]{%
6468   \glslentryfield{#1}{#2}{sort}%
6469 }
```

dex@doextra@esc

```
6470 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6471   \ifx\@glxtr@autoindex@esc\@gl@quotechar
6472   \else
6473     \def\@gl@checkedmkidx{}%
6474     \edef\@glxtr@checkspch{%
6475       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6476       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6477       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6478     \@glxtr@checkspch
6479     \let#1\@gl@checkedmkidx\relax
6480   \fi
```

Escape actual character unless it has already been escaped.

```
6481   \ifx\@glxtr@autoindex@at\@gl@actualchar
6482   \else
6483     \def\@gl@checkedmkidx{}%
6484     \edef\@glxtr@checkspch{%
6485       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6486       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6487       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6488     \@glxtr@checkspch
6489     \let#1\@gl@checkedmkidx\relax
6490   \fi
```

Escape level character unless it has already been escaped.

```
6491   \ifx\@glxtr@autoindex@level\@gl@levelchar
6492   \else
6493     \def\@gl@checkedmkidx{}%
6494     \edef\@glxtr@checkspch{%
6495       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6496       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6497       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6498     \@glxtr@checkspch
6499     \let#1\@gl@checkedmkidx\relax
6500   \fi
```

Escape encap character unless it has already been escaped.

```
6501   \ifx\@glxtr@autoindex@encap\@gl@encapchar
6502   \else
6503     \def\@gl@checkedmkidx{}%
6504     \edef\@glxtr@checkspch{%
6505       \noexpand\@glxtr@autoindex@escencap\expandonce{#1}%
6506       \noexpand\@empty\@glxtr@autoindex@encap\noexpand\@nnil
```

```

6507      \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6508      \@glsxtr@checkspch
6509      \let#1\@gls@checkedmkidx\relax
6510      \fi
6511 }

```

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`.

```
6512 \newcommand*{\@glsxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```

6513 \newcommand*{\GlsXtrSetActualChar}[1]{%
6514   \gdef\@glsxtr@autoindex@at{#1}%
6515   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6516     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6517   }%
6518 }
6519 \@onlypreamble\GlsXtrSetActualChar
6520 \makeatother
6521 \GlsXtrSetActualChar{@}
6522 \makeatletter

```

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

```
6523 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```

6524 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6525   \gdef\@glsxtr@autoindex@encap{#1}%
6526   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
6527     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6528   }%
6529 }
6530 \GlsXtrSetEncapChar{||}
6531 \@onlypreamble\GlsXtrSetEncapChar

```

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

```
6532 \newcommand*{\@glsxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```

6533 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6534   \gdef\@glsxtr@autoindex@level{#1}%
6535   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
6536     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
6537   }%
6538 }
6539 \GlsXtrSetLevelChar{!}
6540 \@onlypreamble\GlsXtrSetLevelChar

```

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

```
6541 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
6542 \newcommand*{\GlsXtrSetEscChar}[1]{%
6543   \gdef\@glsxtr@autoindex@esc{#1}%
6544   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6545     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6546   }%
6547 }
6548 \GlsXtrSetEscChar{"}
6549 \@onlypreamble\GlsXtrSetEscChar
```

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

```
6550 \ifdef\actualchar
6551 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6552 {}
```

Quote character \quotechar:

```
6553 \ifdef\quotechar
6554 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6555 {}
```

Level character \levelchar:

```
6556 \ifdef\levelchar
6557 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6558 {}
```

Encap character \encapchar:

```
6559 \ifdef\encapchar
6560 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6561 {}
```

leto@endescspch

```
6562 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

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

```
6563 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
6564   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
6565   \toks@={#3}%
6566   \ifx\@nnil#3\relax
6567     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
6568   \else
6569     \ifx\@nnil#4\relax
6570       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
6571       \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```

6572      #4#5\@glxtr@endescspch}%
6573  \else
6574      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
6575      \@glxtr@autoindex@esc#1}%
6576      \def\@@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6577  \fi
6578  \fi
6579  \@glxtr@checkspch
6580 }

```

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

```

6581 \renewcommand*{\Glossentrydesc}[1]{%
6582   \glsdoifexistsorwarn{#1}%
6583   {%
6584     \glssetabbrvfmt{\glscategory{#1}}%
6585     \Glsaccessdesc{#1}%
6586   }%
6587 }

```

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

```

6588 \renewcommand*{\glossentrysymbol}[1]{%
6589   \glsdoifexistsorwarn{#1}%
6590   {%
6591     \glssetabbrvfmt{\glscategory{#1}}%
6592     \glsaccesssymbol{#1}%
6593   }%
6594 }

```

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

```

6595 \renewcommand*{\Glossentrysymbol}[1]{%
6596   \glsdoifexistsorwarn{#1}%
6597   {%
6598     \glssetabbrvfmt{\glscategory{#1}}%
6599     \Glsaccesssymbol{#1}%
6600   }%
6601 }

```

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

\eInitialTagging 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.

```

6602 \newcommand*{\GlsXtrEnableInitialTagging}{%
6603   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6604 }
6605 \@onlypreamble\GlsXtrEnableInitialTagging

```

\r@enabletagging Starred version undefines command.

```

6606 \newcommand*{\s@glstr@enabletagging}[2]{%
6607   \undef#2%
6608   \@glstr@enabletagging{#1}{#2}%
6609 }

```

r@enabletagging Internal command.

```

6610 \newcommand*{\@glstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6611   \@for\@glstr@cat:=#1\do
6612   {%
6613     \ifdefempty\@glstr@cat
6614     {}%
6615     {\glsssetcategoryattribute{\@glstr@cat}{tagging}{true}}%
6616   }%
6617   \newrobustcmd*#2[1]{##1}%
6618   \def\@glstr@taggingcs{#2}%
6619   \renewcommand*{\@glstr@activate@initialtagging{%
6620     \let#2\@glstr@tag
6621   }%
6622   \ifundef\@glstr@preglossaryhook
6623   {\GlossariesExtraWarning{Initial tagging requires at least
6624     glossaries.sty v4.19 to work correctly}}%
6625   {}%
6626 }

```

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

```

6627 \ifundef\mfu@checkword@do
6628 {
6629   \newcommand*{\mfu@checkword@do}[1]{%
6630     \ifdefstring{\mfu@checkword@arg}{#1}%
6631     {%
6632       \let\@mfu@domakefirstuc\@firstofone
6633       \listbreak
6634     }%
6635     {}%
6636   }

```

\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.

```

6637 \ifundef\mfu@checkword
6638 {
6639   \newcommand{\@glstr@do@titlecaps@warn}{%
6640     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6641       support not available}%

```

One warning should suffice.

```
6642 \let\@glxtr@do@titlecaps@warn\relax
6643 }
6644 }
6645 {
6646 \renewcommand*{\mfu@checkword}[1]{%
6647 \def\mfu@checkword@arg{#1}%
6648 \let\@mfu@domakefirstuc\makefirstuc
6649 \forlistloop\mfu@checkword@do\@mfu@nocaplist
6650 }
6651 }
6652 }
6653 {}% no patch required
```

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

```
6654 \newcommand*{\@glxtr@do@titlecaps@warn}{}%
```

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

```
6655 \newcommand*{\@glxtr@activate@initialtagging}{}%
```

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

```
6656 \newrobustcmd*{\@glxtr@tag}[1]{%
6657 \glsifattribute{\glscurrententrylabel}{tagging}{true}%
6658 {\glxtrtagfont{#1}}{#1}%
6659 }
```

\glxtrtagfont Used in the glossary.

```
6660 \newcommand*{\glxtrtagfont}[1]{\underline{#1}}
```

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

```
6661 \ifdef\@gls@preglossaryhook
6662 {
6663 \renewcommand*{\@gls@preglossaryhook}{%
6664 \@glxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, \@glxtr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```
6665 \ifundef\@glxtr@org@postdescription
6666 {%
6667 \let\@glxtr@org@postdescription\glspostdescription
6668 \renewcommand*{\glspostdescription}{%
6669 \ifglentryexists{\glscurrententrylabel}%
6670 {%
6671 \glxtrpostdescription
6672 \@glxtr@org@postdescription
```

```

6673         }%
6674         {}%
6675     }%
6676     }%
6677     {}%

    Enable the options used by \@glsxtrp:
6678     \glossxtrsetpopts
6679 }%
6680 }
6681 {}

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.
6682 \newcommand*{\glsxtrpostdescription}{%
6683     \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6684 }

postdescgeneral
6685 \newcommand*{\glsxtrpostdescgeneral}{}

xtrpostdescterm
6686 \newcommand*{\glsxtrpostdescterm}{}

postdescacronym
6687 \newcommand*{\glsxtrpostdescacronym}{}

descabbreviation
6688 \newcommand*{\glsxtrpostdescabbreviation}{}

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.
6689 \newcommand*{\glsdefpostdesc}[2]{%
6690     \csdef{glsxtrpostdesc#1}{#2}%
6691 }

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.
6692 \renewcommand*{\glspostlinkhook}{%
6693     \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6694 }

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.
6695 \newcommand*{\glsxtrpostlinkhook}{%
6696     \glsxtrdiscardperiod{\glslabel}%

```

```

6697 {\glxtrpostlinkendsentence}%
6698 {\glxtrifcustomdiscardperiod
6699 {\glxtrifperiod{\glxtrpostlinkendsentence}{\glxtrpostlink}}}%
6700 {\glxtrpostlink}%
6701 }%
6702 }

```

`\omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```

6703 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}

```

`\glxtrpostlink`

```

6704 \newcommand*{\glxtrpostlink}{%
6705 \csuse{\glxtrpostlink\glscategory{\glslabel}}}%
6706 }

```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glxtrpostlink`.

```

6707 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6708 \ifthenelse{\equal{#1}{}}{%
6709 {\PackageError{glossaries-extra}
6710 {Invalid empty category label in \string\glxtrpostlink}{}}%
6711 {\csdef{\glxtrpostlink#1}{#2}}%
6712 }

```

`\linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```

6713 \newcommand*{\glxtrpostlinkendsentence}{%
6714 \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
6715 {%
6716 \csuse{\glxtrpostlink\glscategory{\glslabel}}%
  Put the full stop back.
6717 .\spacefactor\sfcode'\. \relax
6718 }%
6719 {%

```

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

```

6720 \spacefactor\sfcode'\. \relax
6721 }%
6722 }

```

`\dDescOnFirstUse` 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.

```

6723 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6724 \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}%
6725 }

```

`\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.

```
6726 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6727   \glxtrifwasfirstuse
6728   {%
6729     \ifglshassymbol{\glslabel}%
6730     {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
6731     {}}%
6732   }%
6733   {}%
6734 }
```

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

```
6735 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6736   \glxtrifwasfirstuse
6737   {%
6738     \space\glxtrparen
6739     {%
6740       \ifglshassymbol{\glslabel}%
6741       {\glssaccesssymbol{\glslabel}, }%
6742       {}}%
6743     \glssaccessdesc{\glslabel}%
6744     }%
6745   }%
6746   {}%
6747 }
```

`\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).

```
6748 \newcommand*{\glxtrdiscardperiod}[3]{%
6749   \glxtrifwasfirstuse
6750   {%
6751     \glusifattribute{#1}{retainfirstuseperiod}{true}%
6752     {#3}%
6753     {%
6754       \glusifattribute{#1}{discardperiod}{true}%
6755       {%
6756         \glusifplural
6757         {%
6758           \glusifattribute{#1}{pluraldiscardperiod}{true}%
6759           {\glxtrifperiod{#2}{#3}}%
6760           {#3}%
6761         }%
6762         {%
6763           \glxtrifperiod{#2}{#3}%

```

```

6764     }%
6765     }%
6766     {#3}%
6767     }%
6768 }%
6769 {%
6770   \glsifattribute{#1}{discardperiod}{true}%
6771   {%
6772     \glsifplural
6773     {%
6774       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6775       {\glsxtrifperiod{#2}{#3}}%
6776       {#3}%
6777     }%
6778     {%
6779       \glsxtrifperiod{#2}{#3}%
6780     }%
6781   }%
6782   {#3}%
6783 }%
6784 }

```

`\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`

```

6785 \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 ' ').

```

6786 \newcommand*{\glsxtr@punclist}{.,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

6787 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

6788 \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`.)

```

6789 \newcommand*{\glsxtrifnextpunc}[2]{%
6790   \def\reserved@a{#1}%
6791   \def\reserved@b{#2}%
6792   \futurelet\@glspunc@token\glsxtr@ifnextpunc
6793 }

```

sxtr@ifnextpunc

```
6794 \newcommand*{\glsxtr@ifnextpunc}{%
6795   \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}}%
6796   \reserved@b
6797 }
```

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

```
6798 \newcommand*{\glsxtr@ifpunctoken}[1]{%
6799   \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
6800 }
```

xtr@ifpunctoken

```
6801 \def\@glsxtr@ifpunctoken#1#2{%
6802   \let\reserved@d=#2%
6803   \ifx\reserved@d\@nnil
6804     \let\glsxtr@next\@glsxtr@notfoundinlist
6805   \else
6806     \ifx#1\reserved@d
6807       \let\glsxtr@next\@glsxtr@foundinlist
6808     \else
6809       \let\glsxtr@next\@glsxtr@ifpunctoken
6810     \fi
6811   \fi
6812   \glsxtr@next#1%
6813 }
```

xtr@foundinlist

```
6814 \def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6815 \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.

```
6816 \newcommand{\glsxtrdopostpunc}[1]{%
6817   \glsxtr@ifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
6818 }
```

@glsxtr@swaptwo

```
6819 \newcommand{\@glsxtr@swaptwo}[2]{#2#1}
```

1.7 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.

```
6820 \define@key{glstrabbrv}{category}{%
6821   \edef\glscategorylabel{#1}%
6822   \ifcsdef{@glstrabbrv@current@#1}%
6823   {%
```

Warning should already have been issued.

```
6824   \let\@glstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6825   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6826   \glstr@applyabbrvstyle{\csname @glstrabbrv@current@#1\endcsname}%
6827   \let\GlsXtrWarnDeprecatedAbbrStyle\@glstr@orgwarndep
6828   }%
6829   }%
6830 }
```

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

```
6831 \define@key{glstrabbrv}{shortplural}{%
6832   \def\@glstr@shortpl{#1}%
6833 }
```

Similarly for the long plural form.

```
6834 \define@key{glstrabbrv}{longplural}{%
6835   \def\@glstr@longpl{#1}%
6836 }
```

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

`\glsshortpltok`

```
6837 \newtoks\glsshortpltok
```

`\glslongpltok`

```
6838 \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.

```
6839 \newcommand*{\@glstr@insertdots}[2]{%
6840   \def#1{%
6841     \@glstr@insert@dots#1#2\@nnil
6842   }
```

xtr@insert@dots

```
6843 \newcommand*{\@glsxtr@insert@dots}[2]{%
6844   \ifx\@nnil#2\relax
6845   \let\@glsxtr@insert@dots@next\@gobble
6846   \else
6847   \ifx\relax#2\relax
6848   \else
6849     \appto#1{#2.}%
6850   \fi
6851   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6852   \fi
6853   \@glsxtr@insert@dots@next#1%
6854 }
```

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be defined:

\glsxtrwordsep

```
6855 \newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

\glsxtrword

```
6856 \newcommand*{\glsxtrword}[1]{#1}
```

tr@markwordseps

```
6857 \newcommand*{\@glsxtr@markwordseps}[2]{%
6858   \def#1{}%
6859   \@glsxtr@mark@wordseps#1#2 \@nnil
6860 }
```

r@mark@wordseps

```
6861 \def\@glsxtr@mark@wordseps#1#2 #3{%
6862   \ifdefempty{#1}%
6863   {\def#1{\protect\glsxtrword{#2}}}%
6864   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
6865   \ifx\@nnil#3\relax
6866   \let\@glsxtr@mark@wordseps@next\relax
6867   \else
6868   \def\@glsxtr@mark@wordseps@next{%
6869     \@glsxtr@mark@wordseps#1#3}%
6870   \fi
6871   \@glsxtr@mark@wordseps@next
6872 }
```

newabbreviation Define a new generic abbreviation.

```
6873 \newcommand*{\newabbreviation}[4][ ]{%
6874   \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
6875 }
```

`\newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This just makes it easier to save and restore the original definition.)

```
6876 \newcommand*{\glxtr@newabbreviation}[4]{%
6877   \glskeylisttok{#1}%
6878   \glslabeltok{#2}%
6879   \glsshorttok{#3}%
6880   \glslongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
6881   \def\glxtrorgshort{#3}%
6882   \def\glxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
6883   \def\ExtraCustomAbbreviationFields{}%
```

Initialise accessibility settings if required.

```
6884   \@gls@initaccesskeys
```

Get the category.

```
6885   \def\glscategorylabel{abbreviation}%
6886   \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
6887   \setkeys*{\glxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
6888   \def\@gls@longpl{#4\glspluralsuffix}%
6889   \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
6890   \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6891   {%
6892     \@glxtr@markwordseps\@gls@long{#4}%
6893     \expandafter\def\expandafter\@gls@longpl\expandafter
6894       {\@gls@long\glspluralsuffix}%
6895     \let\@gls@default@longpl\@gls@longpl
```

Update `\glslongtok`.

```
6896     \expandafter\glslongtok\expandafter{\@gls@long}%
6897   }%
6898   {%%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6899   \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6900   {%
6901     \@glxtr@markwordseps\@gls@short{#3}%
6902   }%
6903   {%%
```

Has the insertdots attribute been set?

```
6904     \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6905     {%
6906       \@glxtr@insertdots\@gls@short{#3}%
```

```

6907      \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6908  }%
6909  {\def\@gls@short{#3}}%
6910  }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

6911  \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6912  {%
6913    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6914      '\abbrvpluralsuffix}%
6915  }%
6916  {%

```

Has the noshortplural attribute been set?

```

6917  \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6918  {%
6919    \let\@gls@shortpl\@gls@short
6920  }%
6921  {%
6922    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6923      '\abbrvpluralsuffix}%
6924  }%
6925  }%

```

Update \glsshorttok:

```

6926  \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

6927  \glsextrnewabbrevpresetkeyhook{#1}{#2}{#3}%

```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```

6928  \setkeys*{\glsextrabbrv}[category]{#1}%

```

Has the plural been explicitly set?

```

6929  \ifx\@gls@default@longpl\@gls@longpl
6930  \else

```

Has the markwords attribute been set?

```

6931  \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6932  {%
6933    \expandafter\@glsextr@keywordseps\expandafter\@gls@longpl\expandafter
6934    {\@gls@longpl}%
6935  }%
6936  {}%
6937  \fi

```

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

```

6938  \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6939  \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```

6940  \@gls@setup@default@short@access{#3}%

```

Do any extra setup provided by hook:

6941 \newabbreviationhook

Define this entry:

```
6942 \protected@edef\@do@newglossaryentry{%
6943   \noexpand\newglossaryentry{\the\glslabeltok}%
6944   {%
6945     type=\glstrabbrvtype,%
6946     category=abbreviation,%
6947     short={\the\glsshorttok},%
6948     shortplural={\the\glsshortpltok},%
6949     long={\the\glslongtok},%
6950     longplural={\the\glslongpltok},%
6951     name={\the\glsshorttok},%
6952     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

6953 \ExtraCustomAbbreviationFields

Any explicit fields set in the optional argument override all other settings.

```
6954   \the\glskeylisttok
6955 }%
6956 }%
6957 \do@newglossaryentry
6958 \GlsXtrPostNewAbbreviation
6959 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

6960 \newcommand*{\glstrnewabbrevpresetkeyhook}[3]{}

NewAbbreviation Hook used by abbreviation styles.

6961 \newcommand*{\GlsXtrPostNewAbbreviation}{}

bbreviationhook Hook for use with \newabbreviation.

6962 \newcommand*{\newabbreviationhook}{}

reviationFields

6963 \newcommand*{\CustomAbbreviationFields}{}

\glstrparen For the parenthetical styles.

6964 \newcommand*{\glstrparen}[1]{(#1)}

lsxtrfullformat Full format without case change.

```
6965 \newcommand*{\glstrfullformat}[2]{%
6966   \glsfirstlongfont{\glsaccesslong{#1}}#2\glstrfullsep{#1}%
6967   \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6968 }
```

`\lsxtrfullformat` Full format with case change.

```
6969 \newcommand*{\Glsxtrfullformat}[2]{%
6970   \glsfirstlongfont{\Glsaccesslong{#1}}#2\gsxtrfullsep{#1}%
6971   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
6972 }
```

`\xtrfullplformat` Plural full format without case change.

```
6973 \newcommand*{\gsxtrfullplformat}[2]{%
6974   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6975   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6976 }
```

`\xtrfullplformat` Plural full format with case change.

```
6977 \newcommand*{\Glsxtrfullplformat}[2]{%
6978   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6979   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6980 }
```

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

```
6981 \newcommand*{\gsxtrfullsep}[1]{\space}
```

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

`\nlinefullformat` Full format without case change.

```
6982 \newcommand*{\Glsxtrinlinefullformat}{\gsxtrfullformat}
```

`\nlinefullformat` Full format with case change.

```
6983 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\xtrfullplformat` Plural full format without case change.

```
6984 \newcommand*{\gsxtrinlinefullplformat}{\gsxtrfullplformat}
```

`\inefullplformat` Plural full format with case change.

```
6985 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\gsxtrfull` set of commands instead.

`\glentryfull`

```
6986 \renewcommand*{\glentryfull}[1]{\gsxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
6987 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glentryfullpl`

```
6988 \renewcommand*{\glentryfullpl}[1]{\gsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

6989 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

6990 `\newcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvdefaultfont{#1}}`

`\glbbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

6991 `\newcommand*{\glbbrvdefaultfont}[1]{\glbbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.

6992 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glbbrvdefaultfont`

6993 `\newcommand*{\glbbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.

6994 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.

6995 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glfirstlongfont` Font changing command used for the long form on first use or in the full format.

6996 `\newcommand*{\glfirstlongfont}[1]{\glslongfont{#1}}`

`\glslongdefaultfont`

6997 `\newcommand*{\glfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

6998 `\newcommand*{\glxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glbrvpluralsuffix` Default plural suffix.

6999 `\newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).

7000 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`

7001 `\newcommand*{\ns@glsxtrfull}[2][\{%`

7002 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{\%`

7003 `{\@glsxtr@full{#1}{#2}[]}\%`

7004 `}`

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

7005 `\def\@glsxtr@full#1#2[#3]{\%`

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7006 \@glxtr@record{#1}{#2}{glslink}%
7007 \glsdoifexists{#2}%
7008 {%
7009 \glsssetabbrvfmt{\glscategory{#2}}%
7010 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7011 \let\glslifplural\@secondoftwo
7012 \let\glscapscase\@firstofthree
7013 \let\glslinsert\@empty
7014 \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.

```

7015 \glxtrsetupfulldefs
7016 \@glsl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7017 }%
7018 \glspostlinkhook
7019 }

```

trsetupfulldefs

```

7020 \newcommand*{\glxtrsetupfulldefs}{%
7021 \let\glxtrifwasfirstuse\@firstoftwo
7022 }

```

\Glsxtrfull Full form (first letter uppercase).

```

7023 \newrobustcmd*{\Glsxtrfull}{\@glsl@hyp@opt\@ns@Glsxtrfull}
7024 \newcommand*\ns@Glsxtrfull[2][{}]{%
7025 \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
7026 {\@Glsxtr@full{#1}{#2}[]}%
7027 }

```

\@Glsxtr@full Low-level macro:

```

7028 \def\@Glsxtr@full#1#2[#3]{%
7029 \glsdoifexists{#2}%
7030 {%
7031 \glsssetabbrvfmt{\glscategory{#2}}%
7032 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7033 \let\glslifplural\@secondoftwo
7034 \let\glscapscase\@secondofthree
7035 \let\glslinsert\@empty
7036 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
7037 \glxtrsetupfulldefs
7038 \@glsl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7039 }%
7040 \glspostlinkhook
7041 }

```

`\GLSxtrfull` Full form (all uppercase).

```
7042 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
7043 \newcommand*\ns@GLSxtrfull[2][\%
7044   \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{%
7045     {\@GLSxtr@full{#1}{#2}[]}%
7046 }
```

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

```
7047 \def\@GLSxtr@full#1#2[#3]{%
7048   \glsdoifexists{#2}%
7049   {%
7050     \glssetabbrvfmt{\glscategory{#2}}%
7051     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7052     \let\glsifplural\@secondoftwo
7053     \let\glscapscase\@thirdofthree
7054     \let\glsinsert\@empty
7055     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
7056     \glsxtrsetupfulldefs
7057     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7058   }%
7059   \glspostlinkhook
7060 }
```

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

```
7061 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
7062 \newcommand*\ns@glsxtrfullpl[2][\%
7063   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{%
7064     {\@glsxtr@fullpl{#1}{#2}[]}%
7065 }
```

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

```
7066 \def\@glsxtr@fullpl#1#2[#3]{%
  If the record option has been used, the information needs to be written to the aux file regard-
  less of whether the entry exists (unless indexing has been switched off).
7067   \@glsxtr@record{#1}{#2}{\glslink}%
7068   \glsdoifexists{#2}%
7069   {%
7070     \glssetabbrvfmt{\glscategory{#2}}%
7071     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7072     \let\glsifplural\@firstoftwo
7073     \let\glsapsase\@firstofthree
7074     \let\glsinsert\@empty
7075     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
7076     \glsxtrsetupfulldefs
7077     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7078   }%
7079   \glspostlinkhook
7080 }
```

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

```
7081 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
7082 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
7083   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}]{%
7084     {\@Glsxtr@fullpl{#1}{#2}[]}%
7085 }
```

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

```
7086 \def\@Glsxtr@fullpl#1#2[#3]{%
  If the record option has been used, the information needs to be written to the aux file regard-
  less of whether the entry exists (unless indexing has been switched off).
7087   \@glsxtr@record{#1}{#2}{glslink}%
7088   \glsdoifexists{#2}%
7089   {%
7090     \glssetabbrvfmt{\gls@category{#2}}%
7091     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7092     \let\glsifplural\@firstoftwo
7093     \let\gls@scapscase\@secondofthree
7094     \let\glsinsert\@empty
7095     \def\gls@customtext{\Glsxtrinlinefullplformat{#2}{#3}}%
7096     \glsxtrsetupfulldefs
7097     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7098   }%
7099   \gls@postlinkhook
7100 }
```

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

```
7101 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
7102 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
7103   \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}]{%
7104     {\@GLSxtr@fullpl{#1}{#2}[]}%
7105 }
```

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

```
7106 \def\@GLSxtr@fullpl#1#2[#3]{%
  If the record option has been used, the information needs to be written to the aux file regard-
  less of whether the entry exists (unless indexing has been switched off).
7107   \@glsxtr@record{#1}{#2}{glslink}%
7108   \glsdoifexists{#2}%
7109   {%
7110     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7111     \let\glsifplural\@firstoftwo
7112     \let\gls@scapscase\@thirdofthree
7113     \let\glsinsert\@empty
7114     \def\gls@customtext{%
7115       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
7116     \glsxtrsetupfulldefs
```

```

7117 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7118 }%
7119 \glspostlinkhook
7120 }

```

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

`\glsxtrshort`

```

7121 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

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

```

7122 \newcommand*{\ns@glsxtrshort}[2] [] {%
7123 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2} []}%
7124 }

```

Read in the final optional argument:

```

7125 \def\@glsxtrshort#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7126 \glsxtr@record{#1}{#2}{glslink}%
7127 \glsdoifexists{#2}%
7128 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

7129 \glssetabbrvfmt{\glscategory{#2}}%
7130 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7131 \let\glsxtrifwasfirstuse\@secondoftwo
7132 \let\glsifplural\@secondoftwo
7133 \let\glscapscase\@firstofthree
7134 \let\glsinsert\@empty
7135 \def\glscustomtext{%
7136 \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7137 \ifglsxtrininsertinside\else#3\fi
7138 }%
7139 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7140 }%
7141 \glspostlinkhook
7142 }

```

`\Glsxtrshort`

```

7143 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

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

```

7144 \newcommand*{\ns@Glsxtrshort}[2] [] {%
7145 \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
7146 }

```

Read in the final optional argument:

```

7147 \def\@Glsxtrshort#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7148 \@glstr@record{#1}{#2}{glslink}%
7149 \glsoifexists{#2}%
7150 {%
7151 \glsetabbrvfmt{\glscategory{#2}}%
7152 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7153 \let\glstrifwasfirstuse\@secondoftwo
7154 \let\gl@ifplural\@secondoftwo
7155 \let\glscapscase\@secondofthree
7156 \let\glinsert\@empty
7157 \def\glscustomtext{%
7158 \glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
7159 \ifglstrinsertinside\else#3\fi
7160 }%
7161 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7162 }%
7163 \glspostlinkhook
7164 }

```

\GLSxtrshort

```

7165 \newrobustcmd*{\GLSxtrshort}{\@gl@hyp@opt\ns@GLSxtrshort}

```

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

```

7166 \newcommand*{\ns@GLSxtrshort}[2][{}]{%
7167 \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2}[]}%
7168 }

```

Read in the final optional argument:

```

7169 \def\@GLSxtrshort#1#2[#3]{%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7170 \@glstr@record{#1}{#2}{glslink}%
7171 \glsoifexists{#2}%
7172 {%
7173 \glsetabbrvfmt{\glscategory{#2}}%
7174 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7175 \let\glstrifwasfirstuse\@secondoftwo
7176 \let\gl@ifplural\@secondoftwo
7177 \let\glscapscase\@thirdofthree
7178 \let\glinsert\@empty
7179 \def\glscustomtext{%
7180 \mfirstucMakeUppercase
7181 {\glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
7182 \ifglstrinsertinside\else#3\fi
7183 }%
7184 }%
7185 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7186 }%

```

```

7187 \glspostlinkhook
7188 }

```

\glsxtrlong

```

7189 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}

```

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

```

7190 \newcommand*{\ns@glxtrlong}[2] [] {%
7191   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
7192 }

```

Read in the final optional argument:

```

7193 \def\@glxtrlong#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7194   \@glxtr@record{#1}{#2}{glslink}%
7195   \glsoifexists{#2}%
7196   {%
7197     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7198     \let\glxtrifwasfirstuse\@secondoftwo
7199     \let\glxtrifplural\@secondoftwo
7200     \let\glscapscase\@firstofthree
7201     \let\glxtrinsert\@empty
7202     \def\glscustomtext{%
7203       \glslongfont{\glssaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7204       \ifglxtrinsertinside\else#3\fi
7205     }%
7206     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7207   }%
7208   \glspostlinkhook
7209 }

```

\Glsxtrlong

```

7210 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}

```

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

```

7211 \newcommand*{\ns@Glsxtrlong}[2] [] {%
7212   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
7213 }

```

Read in the final optional argument:

```

7214 \def\@Glsxtrlong#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7215   \@glxtr@record{#1}{#2}{glslink}%
7216   \glsoifexists{#2}%
7217   {%
7218     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7219     \let\glxtrifwasfirstuse\@secondoftwo

```

```

7220 \let\glsifplural\@secondoftwo
7221 \let\glsifscapscase\@secondofthree
7222 \let\glsinsert\@empty
7223 \def\glscustomtext{%
7224   \glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7225   \ifglstrinsertinside\else#3\fi
7226 }%
7227 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7228 }%
7229 \glspostlinkhook
7230 }

```

\GLSxtrlong

```

7231 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}

```

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

```

7232 \newcommand*{\ns@GLSxtrlong}[2] [] {%
7233   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
7234 }

```

Read in the final optional argument:

```

7235 \def\@GLSxtrlong#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7236   \@glsxtr@record{#1}{#2}{\glslink}%
7237   \glsdoifexists{#2}%
7238   {%
7239     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7240     \let\glsxtrifwasfirstuse\@secondoftwo
7241     \let\glsifplural\@secondoftwo
7242     \let\glsifscapscase\@thirdofthree
7243     \let\glsinsert\@empty
7244     \def\glscustomtext{%
7245       \mfirstucMakeUppercase
7246       {\glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7247       \ifglstrinsertinside\else#3\fi
7248     }%
7249     }%
7250     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7251   }%
7252   \glspostlinkhook
7253 }

```

Plural short forms:

\glsxtrshortpl

```

7254 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}

```

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

```

7255 \newcommand*{\ns@glsxtrshortpl}[2] [] {%

```

```

7256 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7257 }

```

Read in the final optional argument:

```

7258 \def\@glsxtrshortpl#1#2[#3]{%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7259 \@glsxtr@record{#1}{#2}{glslink}%
7260 \glsdoifexists{#2}%
7261 {%
7262 \glssetabbrvfmt{\glscategory{#2}}%
7263 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7264 \let\glsxtrifwasfirstuse\@secondoftwo
7265 \let\glsifplural\@firstoftwo
7266 \let\glsapscase\@firstofthree
7267 \let\glsinsert\@empty
7268 \def\glscustomtext{%
7269 \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7270 \ifglsxtrininsertinside\else#3\fi
7271 }%
7272 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7273 }%
7274 \glspostlinkhook
7275 }

```

\Glsxtrshortpl

```

7276 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

7277 \newcommand*{\ns@Glsxtrshortpl}[2] []{%
7278 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7279 }

```

Read in the final optional argument:

```

7280 \def\@Glsxtrshortpl#1#2[#3]{%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7281 \@glsxtr@record{#1}{#2}{glslink}%
7282 \glsdoifexists{#2}%
7283 {%
7284 \glssetabbrvfmt{\glscategory{#2}}%
7285 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7286 \let\glsxtrifwasfirstuse\@secondoftwo
7287 \let\glsifplural\@firstoftwo
7288 \let\glsapscase\@secondofthree
7289 \let\glsinsert\@empty
7290 \def\glscustomtext{%
7291 \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7292 \ifglsxtrininsertinside\else#3\fi

```

```

7293 }%
7294 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7295 }%
7296 \glspostlinkhook
7297 }

```

\GLSxtrshortpl

```

7298 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7299 \newcommand*{\ns@GLSxtrshortpl}[2][]{%
7300   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
7301 }

```

Read in the final optional argument:

```

7302 \def\@GLSxtrshortpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7303   \@glsxtr@record{#1}{#2}{glslink}%
7304   \glsdoifexists{#2}%
7305   {%
7306     \glssetabbrvfmt{\glscategory{#2}}%
7307     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7308     \let\glsxtrifwasfirstuse\@secondoftwo
7309     \let\glsifplural\@firstoftwo
7310     \let\glsapscase\@thirdofthree
7311     \let\glsinsert\@empty
7312     \def\glscustomtext{%
7313       \mfirstucMakeUppercase
7314       {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7315       \ifglsxtrininsertinside\else#3\fi
7316     }%
7317   }%
7318   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7319 }%
7320 \glspostlinkhook
7321 }

```

Plural long forms:

\glsxtrlongpl

```

7322 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7323 \newcommand*{\ns@glsxtrlongpl}[2][]{%
7324   \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
7325 }

```

Read in the final optional argument:

```

7326 \def\@glsxtrlongpl#1#2[#3]{%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7327 \@glstr@record{#1}{#2}{glslink}%
7328 \glsoifexists{#2}%
7329 {%
7330 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7331 \let\glstrifwasfirstuse\@secondoftwo
7332 \let\gl@ifplural\@firstoftwo
7333 \let\glscapscase\@firstofthree
7334 \let\glinsert\@empty
7335 \def\glscustomtext{%
7336 \glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7337 \ifglstrinsertinside\else#3\fi
7338 }%
7339 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7340 }%
7341 \glspostlinkhook
7342 }

```

\Glsxtrlongpl

```

7343 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\ns@Glsxtrlongpl}

```

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

```

7344 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7345 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7346 }

```

Read in the final optional argument:

```

7347 \def\@Glsxtrlongpl#1#2[#3] {%

```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

7348 \@glstr@record{#1}{#2}{glslink}%
7349 \glsoifexists{#2}%
7350 {%
7351 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7352 \let\glstrifwasfirstuse\@secondoftwo
7353 \let\gl@ifplural\@firstoftwo
7354 \let\glscapscase\@secondofthree
7355 \let\glinsert\@empty
7356 \def\glscustomtext{%
7357 \glslongfont{\Glsaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7358 \ifglstrinsertinside\else#3\fi
7359 }%
7360 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7361 }%
7362 \glspostlinkhook
7363 }

```

\GLSxtrlongpl

```

7364 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7365 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7366   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7367 }

    Read in the final optional argument:
7368 \def\@GLSxtrlongpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7369   \@glsxtr@record{#1}{#2}{glslink}%
7370   \glsdoifexists{#2}%
7371   {%
7372     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7373     \let\glsxtrifwasfirstuse\@secondoftwo
7374     \let\glsifplural\@firstoftwo
7375     \let\glsapscase\@thirdofthree
7376     \let\glsinsert\@empty
7377     \def\glscustomtext{%
7378       \mfirstucMakeUppercase
7379       {\glslongfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7380       \ifglsxtrininsertinside\else#3\fi
7381     }%
7382     }%
7383     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7384   }%
7385   \glspostlinkhook
7386 }

```

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

```

7387 \newcommand*{\glssetabbrvfmt}[1]{%
7388   \ifcsdef{\glsabbrv@current@#1}%
7389   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7390   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7391 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7392 \newrobustcmd*{\glsuseabbrvfont}[2]{{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```

7393 \newrobustcmd*{\glsuselongfont}[2]{{\glssetabbrvfmt{#2}\glslongfont{#1}}}

```

sxtrgenabbrvfmt Similar to \glsacgenfmt, but for abbreviations.

```

7394 \newcommand*{\glsxtrgenabbrvfmt}{%
7395   \ifdefempty\glsacgenfmt
7396   {%
7397     \ifglsused\glslabel
7398     {%

```

Subsequent use:

7399 \glsifplural
7400 {%

Subsequent plural form:

7401 \glscapscase
7402 {%

Subsequent plural form, don't adjust case:

7403 \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7404 }%
7405 {%

Subsequent plural form, make first letter upper case:

7406 \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7407 }%
7408 {%

Subsequent plural form, all caps:

7409 \mfirstucMakeUppercase
7410 {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7411 }%
7412 }%
7413 {%

Subsequent singular form

7414 \glscapscase
7415 {%

Subsequent singular form, don't adjust case:

7416 \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7417 }%
7418 {%

Subsequent singular form, make first letter upper case:

7419 \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7420 }%
7421 {%

Subsequent singular form, all caps:

7422 \mfirstucMakeUppercase
7423 {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7424 }%
7425 }%
7426 }%
7427 {%

First use:

7428 \glsifplural
7429 {%

First use plural form:

7430 \glscapscase
7431 {%

First use plural form, don't adjust case:

```
7432      \glxtrfullplformat{\glslabel}{\glsinsert}%  
7433      }%  
7434      {%
```

First use plural form, make first letter upper case:

```
7435      \Glsxtrfullplformat{\glslabel}{\glsinsert}%  
7436      }%  
7437      {%
```

First use plural form, all caps:

```
7438      \mfirstucMakeUppercase  
7439      {\glxtrfullplformat{\glslabel}{\glsinsert}}%  
7440      }%  
7441      }%  
7442      {%
```

First use singular form

```
7443      \glscapscase  
7444      {%
```

First use singular form, don't adjust case:

```
7445      \glxtrfullformat{\glslabel}{\glsinsert}%  
7446      }%  
7447      {%
```

First use singular form, make first letter upper case:

```
7448      \Glsxtrfullformat{\glslabel}{\glsinsert}%  
7449      }%  
7450      {%
```

First use singular form, all caps:

```
7451      \mfirstucMakeUppercase  
7452      {\glxtrfullformat{\glslabel}{\glsinsert}}%  
7453      }%  
7454      }%  
7455      }%  
7456      }%  
7457      {%
```

User supplied text.

```
7458      \glscustomtext  
7459      }%  
7460 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7461 \newcommand*{\glxtrsubsequentfmt}[2]{%  
7462   \glsabbrvfont{\glsaccessshort{#1}}\ifglxtrininsertinside #2\fi}%  
7463   \ifglxtrininsertinside \else#2\fi  
7464 }  
7465 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

```

subsequentplfmt Subsequent use format (plural no case change).
7466 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7467   \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7468   \ifglxtrinsertinside \else#2\fi
7469 }
7470 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
7471 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7472   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7473   \ifglxtrinsertinside \else#2\fi
7474 }
7475 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).
7476 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7477   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7478   \ifglxtrinsertinside \else#2\fi
7479 }
7480 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

```

breviationstyle
7481 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7482   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
7483   {%
7484     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7485   }%
7486   {%
7487     Have abbreviations already been defined for this category?
7488     \ifcsstring{@glsabbrv@current@#1}{#2}%
7489     {%
7490       Style already set.
7491       \def\@glxtr@dostylewarn{}%
7492       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7493       {%
7494         \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7495           style has been switched \MessageBreak
7496           for category ‘#1’, \MessageBreak
7497           but there have already been entries \MessageBreak
7498           defined for this category. Unwanted \MessageBreak
7499           side-effects may result}}%
7500         \@endfortrue
7501       }%
7502       \@glxtr@dostylewarn

```

Set up the style for the given category.

```

7503      \csdef{@glsabbrv@current@#1}{#2}%
7504      \glsxtr@applyabbrvstyle{#2}%
7505      }%
7506      }%
7507 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7508 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7509   \csuse{@glsabbrv@dispstyle@setup@#1}%
7510   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7511 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

7512 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7513   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7514 }

```

`\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.

```

7515 \newcommand*{\newabbreviationstyle}[3]{%
7516   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7517   {%
7518     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7519       defined}{}%
7520   }%
7521   {%
7522     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7523     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7524     #2}%
7525     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7526     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7527     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7528     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7529     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7530     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7531     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7532     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7533     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7534     #3}%
7535   }%
7536 }

```

abbreviationstyle

```
7537 \newcommand*{\renewabbreviationstyle}[3]{%
7538   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7539   {%
7540     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7541   }%
7542   {%
7543     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
7544       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7545       #2}%
7546     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
7547       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7548       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7549       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7550       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7551       #3}%
7552   }%
7553 }
```

abbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```
7554 \newcommand*{\letabbreviationstyle}[2]{%
7555   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7556   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7557 }
```

deprecated@abbrstyle

`\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```
7558 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7559   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7560     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7561     \csuse{@glsabbrv@dispstyle@setup@#2}%
7562   }%
7563   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7564 }
```

deprecatedAbbrStyle Generate warning for deprecated style use.

```
7565 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7566   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7567   use ‘#2’ instead}%
7568 }
```

eAbbrStyleSetup

```
7569 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7570   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7571   {%
7572     \PackageError{glossaries-extra}%
7573     {Unknown abbreviation style definitions ‘#1’}{}%
7574   }%
7575   {%
7576     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7577   }%
7578 }
```

seAbbrStyleFmts

```
7579 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7580   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7581   {%
7582     \PackageError{glossaries-extra}%
7583     {Unknown abbreviation style formats ‘#1’}{}%
7584   }%
7585   {%
7586     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7587   }%
7588 }
```

1.7.2 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.

```
7589 \newif\ifglsxtrinsertinside
7590 \glsxtrinsertinsidefalse
```

trlongshortname

```
7591 \newcommand*{\glsxtrlongshortname}{%
7592   \protect\glsabbrvfont{\the\glsshorttok}%
7593 }
```

long-short

```
7594 \newabbreviationstyle{long-short}%
7595 {%
```

```

7596 \renewcommand*{\CustomAbbreviationFields}{%
7597   name={\glxtrlongshortname},
7598   sort={\the\glsshorttok},
7599   first={\protect\glsfirstlongfont{\the\glslongtok}%
7600     \protect\glxtrfullsep{\the\glslabeltok}%
7601     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7602   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7603     \protect\glxtrfullsep{\the\glslabeltok}%
7604     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7605   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
7606   description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7607 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7608   \glshasattribute{\the\glslabeltok}{regular}%
7609   {%
7610     \glssetattribute{\the\glslabeltok}{regular}{false}%
7611   }%
7612   {}%
7613 }%
7614 }%
7615 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7616 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7617 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7618 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7619 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7620 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7621 \renewcommand*{\glxtrfullformat}[2]{%
7622   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7623   \ifglxtrininsertinside\else##2\fi
7624   \glxtrfullsep{##1}%
7625   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7626 }%
7627 \renewcommand*{\glxtrfullplformat}[2]{%
7628   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7629   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7630   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7631 }%
7632 \renewcommand*{\Glsxtrfullformat}[2]{%
7633   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7634   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7635   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7636 }%
7637 \renewcommand*{\Glsxtrfullplformat}[2]{%
7638   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7639   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7640 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7641 }%
7642 }

```

Set this as the default style for general abbreviations:

```

7643 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7644 \newcommand*{\glsxtrlongshortdescsort}{%
7645 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7646 }

```

ngshortdescname

```

7647 \newcommand*{\glsxtrlongshortdescname}{%
7648 \protect\glslongfont{\the\glslongtok}
7649 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7650 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7651 \newabbreviationstyle{long-short-desc}%
7652 {%
7653 \renewcommand*{\CustomAbbreviationFields}{%
7654 name={\glsxtrlongshortdescname},
7655 sort={\glsxtrlongshortdescsort},%
7656 first={\protect\glsfirstlongfont{\the\glslongtok}%
7657 \protect\glsxtrfullsep{\the\glslabeltok}%
7658 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7659 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7660 \protect\glsxtrfullsep{\the\glslabeltok}%
7661 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7662 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7663 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7664 }%

```

Unset the regular attribute if it has been set.

```

7665 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7666 \glshasattribute{\the\glslabeltok}{regular}%
7667 {%
7668 \glissetattribute{\the\glslabeltok}{regular}{false}%
7669 }%
7670 {}%
7671 }%
7672 }%
7673 {%
7674 \GlsXtrUseAbbrStyleFmts{long-short}%
7675 }

```

trshortlongname

```
7676 \newcommand*{\glxtrshortlongname}{%
7677   \protect\glsabbrvfont{\the\glsshorttok}%
7678 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7679 \newabbreviationstyle{short-long}%
7680 {%
7681   \renewcommand*{\CustomAbbreviationFields}{%
7682     name={\glxtrshortlongname},
7683     sort={\the\glsshorttok},
7684     description={\the\glslongtok},%
7685     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7686       \protect\glxtrfullsep{\the\glslabeltok}%
7687       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7688     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7689       \protect\glxtrfullsep{\the\glslabeltok}%
7690       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7691     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7692 }
```

Unset the regular attribute if it has been set.

```
7692 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7693   \glshasattribute{\the\glslabeltok}{regular}%
7694   {%
7695     \glissetattribute{\the\glslabeltok}{regular}{false}%
7696   }%
7697   {}%
7698 }%
7699 }%
7700 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7701 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7702 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7703 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7704 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7705 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7706 \renewcommand*{\glxtrfullformat}[2]{%
7707   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7708   \ifglxtrininsertinside\else##2\fi
7709   \glxtrfullsep{##1}%
7710   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7711 }%
7712 \renewcommand*{\glxtrfullplformat}[2]{%
7713   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7714   \ifglxtrininsertinside\else##2\fi
7715   \glxtrfullsep{##1}%
7716 }
```

```

7716 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7717 }%
7718 \renewcommand*{\Glsxtrfullformat}[2]{%
7719 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
7720 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7721 \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7722 }%
7723 \renewcommand*{\Glsxtrfullplformat}[2]{%
7724 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
7725 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7726 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7727 }%
7728 }

```

ortlongdescsort

```

7729 \newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7730 \newcommand*{\glsxtrshortlongdescname}{%
7731 \protect\glsabbrvfont{\the\glsshorttok}
7732 \glsxtrparen{\protect\glslongfont{\the\glslongtok}}}%
7733 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7734 \newabbreviationstyle{short-long-desc}%
7735 {%
7736 \renewcommand*{\CustomAbbreviationFields}{%
7737 name={\glsxtrshortlongdescname},
7738 sort={\glsxtrshortlongdescsort},
7739 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7740 \protect\glsxtrfullsep{\the\glslabeltok}%
7741 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7742 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7743 \protect\glsxtrfullsep{\the\glslabeltok}%
7744 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7745 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7746 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7747 }%

```

Unset the regular attribute if it has been set.

```

7748 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7749 \glshasattribute{\the\glslabeltok}{regular}%
7750 {%
7751 \glissetattribute{\the\glslabeltok}{regular}{false}%
7752 }%
7753 {}%
7754 }%

```

```

7755 }%
7756 {%
7757   \GlsXtrUseAbbrStyleFmts{short-long}%
7758 }

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7759 \newcommand*{\glfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7760 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`\glsextrabrvfootnote` `\glsextrabrvfootnote{<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`).

```

7761 \newcommand*{\glsextrabrvfootnote}[2]{\footnote{#2}}

```

`\glsextrfootnotename`

```

7762 \newcommand*{\glsextrfootnotename}{%
7763   \protect\glsabbrvfont{\the\glsshorttok}%
7764 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

7765 \newabbreviationstyle{footnote}%
7766 {%
7767   \renewcommand*{\CustomAbbreviationFields}{%
7768     name={\glsextrfootnotename},
7769     sort={\the\glsshorttok},
7770     description={\the\glslongtok},%

7771     first={\protect\glfirstabbrvfont{\the\glsshorttok}}%
7772     \protect\glsextrabrvfootnote{\the\glslabeltok}%
7773     {\protect\glfirstlongfootnotefont{\the\glslongtok}}},%
7774     firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}}%
7775     \protect\glsextrabrvfootnote{\the\glslabeltok}%
7776     {\protect\glfirstlongfootnotefont{\the\glslongpltok}}},%

7777     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

7778 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7779   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7780   \glshasattribute{\the\glslabeltok}{regular}%

```

```

7781   {%
7782     \glssetattribute{\the\glslabeltok}{regular}{false}%
7783   }%
7784   {}%
7785 }%
7786 }%
7787 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7788 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7789 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7790 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7791 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7792 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7793 \renewcommand*\glsxtrfullformat[2]{%
7794   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7795   \ifglsxtrininsertinside\else##2\fi
7796   \protect\glsxtrabbrvfootnote{##1}%
7797   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7798 }%
7799 \renewcommand*\glsxtrfullplformat[2]{%
7800   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7801   \ifglsxtrininsertinside\else##2\fi
7802   \protect\glsxtrabbrvfootnote{##1}%
7803   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7804 }%
7805 \renewcommand*\Glsxtrfullformat[2]{%
7806   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7807   \ifglsxtrininsertinside\else##2\fi
7808   \protect\glsxtrabbrvfootnote{##1}%
7809   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7810 }%
7811 \renewcommand*\Glsxtrfullplformat[2]{%
7812   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7813   \ifglsxtrininsertinside\else##2\fi
7814   \protect\glsxtrabbrvfootnote{##1}%
7815   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7816 }%

```

The first use full form and the inline full form use the short (long) style.

```

7817 \renewcommand*\glsxtrininlinefullformat[2]{%
7818   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7819   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7820   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7821 }%
7822 \renewcommand*\glsxtrininlinefullplformat[2]{%
7823   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7824   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7825   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

7826 }%
7827 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7828   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7829   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7830   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7831 }%
7832 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7833   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7834   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7835   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7836 }%
7837 }

```

short-footnote

```
7838 \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.

```

7839 \newabbreviationstyle{postfootnote}%
7840 {%
7841   \renewcommand*{\CustomAbbreviationFields}{%
7842     name={\glxtrfootnotename},
7843     sort={\the\glsshorttok},
7844     description={\the\glslongtok},%
7845     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7846     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7847     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

7848   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7849     \csdef{glxtrpostlink\glscategorylabel}{%
7850       \glxtrifwasfirstuse
7851       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7852         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7853         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7854       }%
7855     }%
7856   }%
7857   \glshasattribute{\the\glslabeltok}{regular}%
7858   {%
7859     \glssetattribute{\the\glslabeltok}{regular}{false}%
7860   }%
7861   {}%
7862 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7863 \renewcommand*{\glxtrsetupfulldefs}{%
7864   \let\glxtrifwasfirstuse\@secondoftwo
7865 }%
7866 }%
7867 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7868 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7869 \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvdefaultfont{##1}}%
7870 \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvdefaultfont{##1}}%
7871 \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlongfootnotefont{##1}}%
7872 \renewcommand*{\glxtrlongfont}[1]{\glxtrlongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7873 \renewcommand*{\glxtrfullformat}[2]{%
7874   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7875   \ifglxtrininsertinside\else##2\fi
7876 }%
7877 \renewcommand*{\glxtrfullplformat}[2]{%
7878   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7879   \ifglxtrininsertinside\else##2\fi
7880 }%
7881 \renewcommand*{\Glsxtrfullformat}[2]{%
7882   \glxtrfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7883   \ifglxtrininsertinside\else##2\fi
7884 }%
7885 \renewcommand*{\Glsxtrfullplformat}[2]{%
7886   \glxtrfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7887   \ifglxtrininsertinside\else##2\fi
7888 }%
```

The first use full form and the inline full form use the short (long) style.

```
7889 \renewcommand*{\glxtrininlinefullformat}[2]{%
7890   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7891   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7892   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslong{##1}}}%
7893 }%
7894 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7895   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7896   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7897   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslongpl{##1}}}%
7898 }%
7899 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
7900   \glxtrfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7901   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7902   \glxtrparen{\glxtrfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7903 }%
7904 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```

```

7905 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7906 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7907 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7908 }%
7909 }

```

rt-postfootnote

```

7910 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

shortnolongname

```

7911 \newcommand*{\glxtrshortnolongname}{%
7912 \protect\glsabbrvfont{\the\glsshorttok}%
7913 }

```

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.

```

7914 \newabbreviationstyle{short}%
7915 {%
7916 \renewcommand*{\CustomAbbreviationFields}{%
7917 name={\glxtrshortnolongname},
7918 sort={\the\glsshorttok},
7919 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7920 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7921 text={\protect\glsabbrvfont{\the\glsshorttok}},
7922 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7923 description={\the\glslongtok}}%
7924 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7925 \glssetattribute{\the\glslabeltok}{regular}{true}}%
7926 }%
7927 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7928 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7929 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7930 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7931 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7932 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7933 \renewcommand*{\glxtrinlinefullformat}[2]{%
7934 \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7935 \ifglxtrinsertinside##2\fi}%
7936 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7937 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7938 }%
7939 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7940 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%

```

```

7941 \ifglxtrinsertinside##2\fi}%
7942 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7943 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7944 }%
7945 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7946 \protect\glsfirstabbrvfont{\glssaccessshort{##1}%
7947 \ifglxtrinsertinside##2\fi}%
7948 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7949 \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7950 }%
7951 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7952 \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}%
7953 \ifglxtrinsertinside##2\fi}%
7954 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7955 \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7956 }%

```

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.

```

7957 \renewcommand*{\glxtrfullformat}[2]{%
7958 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7959 \ifglxtrinsertinside\else##2\fi
7960 }%
7961 \renewcommand*{\glxtrfullplformat}[2]{%
7962 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7963 \ifglxtrinsertinside\else##2\fi
7964 }%
7965 \renewcommand*{\Glsxtrfullformat}[2]{%
7966 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7967 \ifglxtrinsertinside\else##2\fi
7968 }%
7969 \renewcommand*{\Glsxtrfullplformat}[2]{%
7970 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7971 \ifglxtrinsertinside\else##2\fi
7972 }%
7973 }

```

Set this as the default style for acronyms:

```

7974 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7975 \letabbreviationstyle{short-nolong}{short}

```

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7976 \newabbreviationstyle{short-nolong-noreg}%
7977 {%
7978 \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7979 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

7980 \glshasattribute{\the\glslabeltok}{regular}%
7981 {%
7982 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7983 }%
7984 {}%
7985 }%
7986 }%
7987 {%
7988 \GlsXtrUseAbbrStyleFmts{short-nolong}%
7989 }

```

trshortdescname

```

7990 \newcommand*{\glxtrshortdescname}{%
7991 \protect\glsabbrvfont{\the\glsshorttok}%
7992 \protect\glxtrfullsep{\the\glslabeltok}%
7993 \protect\glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
7994 }

```

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.

```

7995 \newabbreviationstyle{short-desc}%
7996 {%
7997 \renewcommand*{\CustomAbbreviationFields}{%
7998 name={\glxtrshortdescname},
7999 sort={\the\glsshorttok},
8000 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
8001 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
8002 text={\protect\glsabbrvfont{\the\glsshorttok}},
8003 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
8004 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8005 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8006 }%
8007 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8008 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8009 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8010 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8011 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8012 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8013 \renewcommand*{\glxtrinlinefullformat}[2]{%
8014 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8015 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8016 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8017 }%
8018 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8019 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8020 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

8021 \glstrparen{\glfirstlongfont{\glsaccesslongpl{##1}}}%
8022 }%
8023 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8024 \glfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8025 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8026 \glstrparen{\glfirstlongfont{\glsaccesslong{##1}}}%
8027 }%
8028 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8029 \glfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8030 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8031 \glstrparen{\glfirstlongfont{\glsaccesslongpl{##1}}}%
8032 }%

```

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.

```

8033 \renewcommand*{\glxtrfullformat}[2]{%
8034 \glfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8035 \ifglxtrinsertinside\else##2\fi
8036 }%
8037 \renewcommand*{\glxtrfullplformat}[2]{%
8038 \glfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8039 \ifglxtrinsertinside\else##2\fi
8040 }%
8041 \renewcommand*{\Glsxtrfullformat}[2]{%
8042 \glfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8043 \ifglxtrinsertinside\else##2\fi
8044 }%
8045 \renewcommand*{\Glsxtrfullplformat}[2]{%
8046 \glfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8047 \ifglxtrinsertinside\else##2\fi
8048 }%
8049 }

```

ort-nolong-desc

```
8050 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

8051 \newabbreviationstyle{short-nolong-desc-noreg}%
8052 {%
8053 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

8054 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8055 \glshasattribute{\the\glslabeltok}{regular}%
8056 {%
8057 \glissetattribute{\the\glslabeltok}{regular}{false}%
8058 }%
8059 {}%
8060 }%
8061 }%

```

```

8062 {%
8063   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
8064 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

8065 \newabbreviationstyle{nolong-short}%
8066 {%
8067   \GlsXtrUseAbbrStyleSetup{short-nolong}%
8068 }%
8069 {%
8070   \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8071   \renewcommand*{\glxtrinlinefullformat}[2]{%
8072     \protect\glsfirstlongfont{\glsaccesslong{##1}%
8073       \ifglxtrininsertinside##2\fi}%
8074     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8075     \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8076   }%
8077   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8078     \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8079       \ifglxtrininsertinside##2\fi}%
8080     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8081     \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8082   }%
8083   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8084     \protect\glsfirstlongfont{\glsaccesslong{##1}%
8085       \ifglxtrininsertinside##2\fi}%
8086     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8087     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8088   }%
8089   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8090     \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8091       \ifglxtrininsertinside##2\fi}%
8092     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8093     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8094   }%
8095 }

```

nolong-short-noreg Like nolong-short but doesn't set the regular attribute.

```

8096 \newabbreviationstyle{nolong-short-noreg}%
8097 {%
8098   \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

8099   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8100     \glshasattribute{\the\glslabeltok}{regular}%
8101     {%
8102       \glissetattribute{\the\glslabeltok}{regular}{false}%

```

```

8103 }%
8104 {}%
8105 }%
8106 }%
8107 {%
8108 \GlsXtrUseAbbrStyleFmts{nolong-short}%
8109 }

```

noshortdescname

```

8110 \newcommand*{\glxtrlongnoshortdescname}{%
8111 \protect\glslongfont{\the\glslongtok}%
8112 }

```

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.

```

8113 \newabbreviationstyle{long-desc}%
8114 {%
8115 \renewcommand*{\CustomAbbreviationFields}{%
8116 name={\glxtrlongnoshortdescname},
8117 sort={\the\glslongtok},
8118 first={\protect\glslongfont{\the\glslongtok}},
8119 firstplural={\protect\glslongfont{\the\glslongpltok}},
8120 text={\glslongfont{\the\glslongtok}},
8121 plural={\glslongfont{\the\glslongpltok}}}%
8122 }%
8123 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8124 \glsssetattribute{\the\glslabeltok}{regular}{true}}%
8125 }%
8126 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8127 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8128 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
8129 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
8130 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8131 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8132 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8133 \glslongfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8134 \ifglxtrininsertinside \else##2\fi
8135 }%
8136 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8137 \glslongfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8138 \ifglxtrininsertinside \else##2\fi
8139 }%
8140 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8141 \glslongfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8142 \ifglxtrininsertinside \else##2\fi

```

```

8143 }%
8144 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8145   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
8146   \ifglsxtrinsertinside \else##2\fi
8147 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8148 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8149   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8150   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8151   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8152 }%
8153 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8154   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8155   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8156   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8157 }%
8158 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8159   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8160   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8161   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8162 }%
8163 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8164   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8165   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8166   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8167 }%

```

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.

```

8168 \renewcommand*{\glsxtrfullformat}[2]{%
8169   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8170   \ifglsxtrinsertinside\else##2\fi
8171 }%
8172 \renewcommand*{\glsxtrfullplformat}[2]{%
8173   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8174   \ifglsxtrinsertinside\else##2\fi
8175 }%
8176 \renewcommand*{\Glsxtrfullformat}[2]{%
8177   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8178   \ifglsxtrinsertinside\else##2\fi
8179 }%
8180 \renewcommand*{\Glsxtrfullplformat}[2]{%
8181   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8182   \ifglsxtrinsertinside\else##2\fi
8183 }%
8184 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

8185 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```
8186 \newabbreviationstyle{long-noshort-desc-noreg}%
8187 {%
8188   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
      Unset the regular attribute if it has been set.
8189   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8190     \glshasattribute{\the\glslabeltok}{regular}%
8191     {%
8192       \glissetattribute{\the\glslabeltok}{regular}{false}%
8193     }%
8194   }%
8195 }%
8196 }%
8197 {%
8198   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8199 }
```

longnoshortname

```
8200 \newcommand*{\glsxtrlongnoshortname}{%
8201   \protect\glsabbrvfont{\the\glsshorttok}%
8202 }
```

long It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```
8203 \newabbreviationstyle{long}%
8204 {%
8205   \renewcommand*{\CustomAbbreviationFields}{%
8206     name={\glsxtrlongnoshortname},
8207     sort={\the\glsshorttok},
8208     first={\protect\glsfirstlongfont{\the\glslongtok}},
8209     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8210     text={\glslongfont{\the\glslongtok}},
8211     plural={\glslongfont{\the\glslongpltok}},%
8212     description={\the\glslongtok}%
8213   }%
8214   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8215     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8216 }%
8217 {%
8218   \GlsXtrUseAbbrStyleFmts{long-desc}%
8219 }
```

long-noshort Provide a synonym that matches similar styles.

```
8220 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```

8221 \newabbreviationstyle{long-noshort-noreg}%
8222 {%
8223   \GlsXtrUseAbbrStyleSetup{long-noshort}%
      Unset the regular attribute if it has been set.
8224   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8225     \glshasattribute{\the\glslabeltok}{regular}%
8226     {%
8227       \glissetattribute{\the\glslabeltok}{regular}{false}%
8228     }%
8229   }%
8230 }%
8231 }%
8232 {%
8233   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8234 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

```

\glsxtrscfont   Maintained for backward-compatibility.
8235 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}

\glssabrvscfont Added for consistent naming.
8236 \newcommand*{\glssabrvscfont}{\glsxtrscfont}

sxtrfirstscfont Maintained for backward-compatibility.
8237 \newcommand*{\glsxtrfirstscfont}[1]{\glssabrvscfont{#1}}

irstabrvscfont  Added for consistent naming.
8238 \newcommand*{\glsfirstabrvscfont}{\glsxtrfirstscfont}

```

and for the default short form suffix:

```

\glsxtrscsuffix
8239 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabrvpluralsuffix}}

```

```

long-short-sc
8240 \newabbreviationstyle{long-short-sc}%
8241 {%
8242   \renewcommand*{\CustomAbbreviationFields}{%
8243     name={\glsxtrlongshortname},
8244     sort={\the\glsshorttok},
8245     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8246       \protect\glsxtrfullsep{\the\glslabeltok}%
8247       \glsxtrparen{\protect\glsfirstabrvscfont{\the\glsshorttok}}},%
8248     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8249       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

8250     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8251     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
8252     description={\the\glslongtok}}%
8253 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8254     \glsattribute{\the\glslabeltok}{regular}%
8255     {%
8256         \glsssetattribute{\the\glslabeltok}{regular}{false}%
8257     }%
8258     {}%
8259 }%
8260}%
8261{%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8262 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8263 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8264 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

8265 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8266 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8267 \renewcommand*{\glxtrfullformat}[2]{%
8268     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8269     \ifglxtrinsertinside\else##2\fi
8270     \glxtrfullsep{##1}}%
8271 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8272}%
8273 \renewcommand*{\glxtrfullplformat}[2]{%
8274     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8275     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8276 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8277}%
8278 \renewcommand*{\Glsxtrfullformat}[2]{%
8279     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8280     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8281 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8282}%
8283 \renewcommand*{\Glsxtrfullplformat}[2]{%
8284     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8285     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8286 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8287}%
8288}

```

g-short-sc-desc

```

8289 \newabbreviationstyle{long-short-sc-desc}%
8290 {%
8291     \renewcommand*{\CustomAbbreviationFields}{%

```

```

8292     name={\glxtrlongshortdescname},
8293     sort={\glxtrlongshortdescsort},%
8294     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8295       \protect\glxtrfullsep{\the\glslabeltok}%
8296       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8297     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8298       \protect\glxtrfullsep{\the\glslabeltok}%
8299       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8300     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8301     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8302 }%

```

Unset the regular attribute if it has been set.

```

8303 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8304   \glshasattribute{\the\glslabeltok}{regular}%
8305   {%
8306     \glissetattribute{\the\glslabeltok}{regular}{false}%
8307   }%
8308   {}%
8309 }%
8310 }%
8311 {%

```

As long-short-sc style:

```

8312 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8313 }

```

Now the short (long) version

```

8314 \newabbreviationstyle{short-sc-long}%
8315 {%
8316   \renewcommand*{\CustomAbbreviationFields}{%
8317     name={\glxtrshortlongname},
8318     sort={\the\glsshorttok},
8319     description={\the\glslongtok},%
8320     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8321       \protect\glxtrfullsep{\the\glslabeltok}%
8322       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8323     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8324       \protect\glxtrfullsep{\the\glslabeltok}%
8325       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8326     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8327 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8328   \glshasattribute{\the\glslabeltok}{regular}%
8329   {%
8330     \glissetattribute{\the\glslabeltok}{regular}{false}%
8331   }%
8332   {}%
8333 }%
8334 }%

```

8335 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8336 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8337 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8338 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8339 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8340 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8341 \renewcommand*{\glxtrfullformat}[2]{%
8342   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8343   \ifglxtrininsertinside\else##2\fi
8344   \glxtrfullsep{##1}%
8345   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8346 }%
8347 \renewcommand*{\glxtrfullplformat}[2]{%
8348   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8349   \ifglxtrininsertinside\else##2\fi
8350   \glxtrfullsep{##1}%
8351   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8352 }%
8353 \renewcommand*{\Glsxtrfullformat}[2]{%
8354   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8355   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8356   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8357 }%
8358 \renewcommand*{\Glsxtrfullplformat}[2]{%
8359   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8360   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8361   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8362 }%
8363 }
```

As before but user provides description

```
8364 \newabbreviationstyle{short-sc-long-desc}%
8365 {%
8366   \renewcommand*{\CustomAbbreviationFields}{%
8367     name={\glxtrshortlongdescname},
8368     sort={\glxtrshortlongdescsort},
8369     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
8370     \protect\glxtrfullsep{\the\glslabeltok}%
8371     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8372     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
8373     \protect\glxtrfullsep{\the\glslabeltok}%
8374     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8375     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8376     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8377   }%
```

Unset the regular attribute if it has been set.

```

8378 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8379   \glshasattribute{\the\glslabeltok}{regular}}%
8380   {%
8381     \glissetattribute{\the\glslabeltok}{regular}{false}}%
8382   }%
8383   {}%
8384 }%
8385 }%
8386 {%

```

As short-sc-long style:

```

8387 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8388 }

```

short-sc

```

8389 \newabbreviationstyle{short-sc}%
8390 {%
8391   \renewcommand*{\CustomAbbreviationFields}{%
8392     name={\glxtrshortnolongname},
8393     sort={\the\glsshorttok},
8394     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8395     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8396     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8397     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8398     description={\the\glslongtok}}%
8399   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8400     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8401   }%
8402 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8403 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8404 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8405 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8406 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8407 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8408 \renewcommand*{\glxtrinlinefullformat}[2]{%
8409   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8410   \ifglxtrininsertinside##2\fi}%
8411   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8412   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8413 }%
8414 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8415   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8416   \ifglxtrininsertinside##2\fi}%
8417   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8418   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8419 }%

```

```

8420 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8421   \protect\glsfirstabbrvscfont{\Glsaccesssshort{##1}%
8422     \ifglxtrinsertinside##2\fi}%
8423   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8424   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8425 }%
8426 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8427   \protect\glsfirstabbrvscfont{\Glsaccesssshortpl{##1}%
8428     \ifglxtrinsertinside##2\fi}%
8429   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8430   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8431 }%

```

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.

```

8432 \renewcommand*{\glxtrfullformat}[2]{%
8433   \glsfirstabbrvscfont{\Glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
8434   \ifglxtrinsertinside\else##2\fi
8435 }%
8436 \renewcommand*{\glxtrfullplformat}[2]{%
8437   \glsfirstabbrvscfont{\Glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
8438   \ifglxtrinsertinside\else##2\fi
8439 }%
8440 \renewcommand*{\Glsxtrfullformat}[2]{%
8441   \glsfirstabbrvscfont{\Glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
8442   \ifglxtrinsertinside\else##2\fi
8443 }%
8444 \renewcommand*{\Glsxtrfullplformat}[2]{%
8445   \glsfirstabbrvscfont{\Glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
8446   \ifglxtrinsertinside\else##2\fi
8447 }%
8448 }

```

short-sc-nolong

```

8449 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8450 \newabbreviationstyle{short-sc-desc}{%
8451 {%
8452   \renewcommand*{\CustomAbbreviationFields}{%
8453     name={\glxtrshortdescname},
8454     sort={\the\glsshorttok},
8455     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8456     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8457     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8458     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8459   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8460     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8461 }%
8462 }%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8463 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8464 \renewcommand*\glabbrvfont[1]{\glabbrvscfont{##1}}%
8465 \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvscfont{##1}}%
8466 \renewcommand*\glfirstlongfont[1]{\glfirstlongdefaultfont{##1}}%
8467 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8468 \renewcommand*\glxtrinlinefullformat}[2]{%
8469   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8470   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8471   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8472 }%
8473 \renewcommand*\glxtrinlinefullplformat}[2]{%
8474   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8475   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8476   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8477 }%
8478 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8479   \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8480   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8481   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8482 }%
8483 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8484   \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8485   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8486   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8487 }%

```

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.

```

8488 \renewcommand*\glxtrfullformat}[2]{%
8489   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8490   \ifglxtrininsertinside\else##2\fi
8491 }%
8492 \renewcommand*\glxtrfullplformat}[2]{%
8493   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8494   \ifglxtrininsertinside\else##2\fi
8495 }%
8496 \renewcommand*\Glsxtrfullformat}[2]{%
8497   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8498   \ifglxtrininsertinside\else##2\fi
8499 }%
8500 \renewcommand*\Glsxtrfullplformat}[2]{%
8501   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8502   \ifglxtrininsertinside\else##2\fi
8503 }%
8504 }

```

-sc-nolong-desc

```
8505 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
8506 \newabbreviationstyle{nolong-short-sc}%
8507 {%
8508   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8509 }%
8510 {%
8511   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8512 \renewcommand*{\glxtrinlinefullformat}[2]{%
8513   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}}%
8514   \ifglxtrininsertinside##2\fi}%
8515   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8516   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8517 }%
8518 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8519   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}%
8520   \ifglxtrininsertinside##2\fi}%
8521   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8522   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8523 }%
8524 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8525   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}}%
8526   \ifglxtrininsertinside##2\fi}%
8527   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8528   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8529 }%
8530 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8531   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}%
8532   \ifglxtrininsertinside##2\fi}%
8533   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8534   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8535 }%
8536 }
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```
8537 \newabbreviationstyle{long-noshort-sc}%
8538 {%
8539   \renewcommand*{\CustomAbbreviationFields}{%
8540     name={\glxtrlongnoshortname},
8541     sort={\the\glsshorttok},
8542     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8543     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8544     text={\protect\glslongdefaultfont{\the\glslongtok}},
8545     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8546     description={\the\glslongtok}%
8547   }
```

```

8547 }%
8548 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8549   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8550 }%
8551 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8552 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8553 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8554 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8555 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8556 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8557 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8558   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8559   \ifglxtrininsertinside \else##2\fi
8560 }%
8561 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8562   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8563   \ifglxtrininsertinside \else##2\fi
8564 }%
8565 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8566   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8567   \ifglxtrininsertinside \else##2\fi
8568 }%
8569 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8570   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8571   \ifglxtrininsertinside \else##2\fi
8572 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8573 \renewcommand*{\glxtrinlinefullformat}[2]{%
8574   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8575   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8576   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8577 }%
8578 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8579   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8580   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8581   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8582 }%
8583 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8584   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8585   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8586   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8587 }%
8588 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8589   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8590   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8591   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%

```

8592 }%

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.

```
8593 \renewcommand*{\glxtrfullformat}[2]{%
8594   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8595   \ifglxtrinsertinside\else##2\fi
8596 }%
8597 \renewcommand*{\glxtrfullplformat}[2]{%
8598   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8599   \ifglxtrinsertinside\else##2\fi
8600 }%
8601 \renewcommand*{\Glsxtrfullformat}[2]{%
8602   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8603   \ifglxtrinsertinside\else##2\fi
8604 }%
8605 \renewcommand*{\Glsxtrfullplformat}[2]{%
8606   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8607   \ifglxtrinsertinside\else##2\fi
8608 }%
8609 }
```

long-sc Backward compatibility:

```
8610 \@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`.

```
8611 \newabbreviationstyle{long-noshort-sc-desc}%
8612 {%
8613   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8614 }%
8615 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8616 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8617 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8618 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8619 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8620 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8621 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8622   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8623   \ifglxtrinsertinside \else##2\fi
8624 }%
8625 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8626   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8627   \ifglxtrinsertinside \else##2\fi
8628 }%
8629 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
```

```

8630 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8631 \ifglxtrinsertinside \else##2\fi
8632 }%
8633 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8634 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8635 \ifglxtrinsertinside \else##2\fi
8636 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8637 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8638 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8639 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8640 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8641 }%
8642 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8643 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8644 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8645 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8646 }%
8647 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8648 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8649 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8650 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8651 }%
8652 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8653 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8654 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8655 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8656 }%

```

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.

```

8657 \renewcommand*{\glsxtrfullformat}[2]{%
8658 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8659 \ifglxtrinsertinside\else##2\fi
8660 }%
8661 \renewcommand*{\glsxtrfullplformat}[2]{%
8662 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8663 \ifglxtrinsertinside\else##2\fi
8664 }%
8665 \renewcommand*{\Glsxtrfullformat}[2]{%
8666 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8667 \ifglxtrinsertinside\else##2\fi
8668 }%
8669 \renewcommand*{\Glsxtrfullplformat}[2]{%
8670 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8671 \ifglxtrinsertinside\else##2\fi
8672 }%
8673 }

```

long-desc-sc Backward compatibility:

```
8674 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
8675 \newabbreviationstyle{short-sc-footnote}%
8676 {%
8677   \renewcommand*{\CustomAbbreviationFields}{%
8678     name={\glxtrfootnotename},
8679     sort={\the\glsshorttok},
8680     description={\the\glslongtok},%
8681     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8682       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8683       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8684     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8685       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8686       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8687     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8688   }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
8688   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8689     \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8690     \glshasattribute{\the\glslabeltok}{regular}%
8691     {%
8692       \glsssetattribute{\the\glslabeltok}{regular}{false}%
8693     }%
8694   }%
8695 }%
8696 }%
8697 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8698 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8699 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8700 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8701 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8702 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
8703 \renewcommand*{\glxtrfullformat}[2]{%
8704   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8705   \ifglxtrininsertinside\else##2\fi
8706   \protect\glxtrabbrvfootnote{##1}%
8707   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8708 }%
8709 \renewcommand*{\glxtrfullplformat}[2]{%
8710   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8711   \ifglxtrininsertinside\else##2\fi
8712   \protect\glxtrabbrvfootnote{##1}%
8713   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8714 }
```

```

8714 }%
8715 \renewcommand*{\Glsxtrfullformat}[2]{%
8716   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8717   \ifglxtrinsertinside\else##2\fi
8718   \protect\glxtrabbrvfootnote{##1}%
8719   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8720 }%
8721 \renewcommand*{\Glsxtrfullplformat}[2]{%
8722   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8723   \ifglxtrinsertinside\else##2\fi
8724   \protect\glxtrabbrvfootnote{##1}%
8725   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8726 }%

```

The first use full form and the inline full form use the short (long) style.

```

8727 \renewcommand*{\glxtrinlinefullformat}[2]{%
8728   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8729   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8730   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8731 }%
8732 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8733   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8734   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8735   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8736 }%
8737 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8738   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8739   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8740   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8741 }%
8742 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8743   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8744   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8745   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8746 }%
8747 }

```

footnote-sc Backward compatibility:

```

8748 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8749 \newabbreviationstyle{short-sc-postfootnote}%
8750 {%
8751   \renewcommand*{\CustomAbbreviationFields}{%
8752     name={\glxtrfootnotename},
8753     sort={\the\glsshorttok},
8754     description={\the\glslongtok},%
8755     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8756     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8757     plural={\protect\glsabbrvscfont{\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.

```
8758 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8759 \csdef{glxtrpostlink\glscategorylabel}{%
8760 \glxtrifwasfirstuse
8761 {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
8762 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8763 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
8764 }%
8765 {%
8766 }%
8767 \glshasattribute{\the\glslabeltok}{regular}%
8768 {%
8769 \glissetattribute{\the\glslabeltok}{regular}{false}%
8770 }%
8771 {%
8772 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8773 \renewcommand*{\glxtrsetupfulldefs}{%
8774 \let\glxtrifwasfirstuse\@secondoftwo
8775 }%
8776 }%
8777 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8778 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8779 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8780 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8781 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8782 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8783 \renewcommand*{\glxtrfullformat}[2]{%
8784 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8785 \ifglxtrininsertinside\else##2\fi
8786 }%
8787 \renewcommand*{\glxtrfullplformat}[2]{%
8788 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8789 \ifglxtrininsertinside\else##2\fi
8790 }%
8791 \renewcommand*{\Glsxtrfullformat}[2]{%
8792 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8793 \ifglxtrininsertinside\else##2\fi
8794 }%
8795 \renewcommand*{\Glsxtrfullplformat}[2]{%
```

```

8796 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8797 \ifglxtrinsertinside\else##2\fi
8798 }%

```

The first use full form and the inline full form use the short (long) style.

```

8799 \renewcommand*{\glxtrinlinefullformat}[2]{%
8800 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8801 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8802 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8803 }%
8804 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8805 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8806 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8807 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8808 }%
8809 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8810 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8811 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8812 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8813 }%
8814 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8815 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8816 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8817 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8818 }%
8819 }

```

postfootnote-sc Backward compatibility:

```

8820 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

```

\glxtrsmfont Maintained for backward compatibility.
8821 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
8822 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8823 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
8824 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

```

and for the default short form suffix:

\glxtrrmsuffix

```
8825 \newcommand*{\glxtrrmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8826 \newabbreviationstyle{long-short-sm}%
8827 {%
8828   \renewcommand*{\CustomAbbreviationFields}{%
8829     name={\glxtrlongshortname},
8830     sort={\the\glsshorttok},
8831     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8832       \protect\glxtrfullsep{\the\glslabeltok}%
8833       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8834     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8835       \protect\glxtrfullsep{\the\glslabeltok}%
8836       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8837     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8838     description={\the\glslongtok}}%
8839   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8840     \glshasattribute{\the\glslabeltok}{regular}%
8841     {%
8842       \glissetattribute{\the\glslabeltok}{regular}{false}%
8843     }%
8844   }%
8845 }%
8846 }%
8847 {%
8848   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8849   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8850   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrrmsuffix}%

```

Use the default long fonts.

```
8851 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8852 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8853 \renewcommand*{\glxtrfullformat}[2]{%
8854   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8855   \ifglxtrininsertinside\else##2\fi
8856   \glxtrfullsep{##1}%
8857   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8858 }%
8859 \renewcommand*{\glxtrfullplformat}[2]{%
8860   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8861   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8862   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8863 }%
8864 \renewcommand*{\Glsxtrfullformat}[2]{%
8865   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8866   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8867   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%

```

```

8868 }%
8869 \renewcommand*{\Glsxtrfullplformat}[2]{%
8870   \glstfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglstxtrinsertinside##2\fi}%
8871   \ifglstxtrinsertinside\else##2\fi\glstxtrfullsep{##1}%
8872   \glstxtrparen{\glstfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8873 }%
8874 }

```

g-short-sm-desc

```

8875 \newabbreviationstyle{long-short-sm-desc}%
8876 {%
8877   \renewcommand*{\CustomAbbreviationFields}{%
8878     name={\glstxtrlongshortdescname},
8879     sort={\glstxtrlongshortdescsort},%
8880     first={\protect\glstfirstlongdefaultfont{\the\glstlongtok}%
8881       \protect\glstxtrfullsep{\the\glstlabeltok}%
8882       \glstxtrparen{\protect\glstfirstabbrvsmfont{\the\glstshorttok}}},%
8883     firstplural={\protect\glstfirstlongdefaultfont{\the\glstlongpltok}%
8884       \protect\glstxtrfullsep{\the\glstlabeltok}%
8885       \glstxtrparen{\protect\glstfirstabbrvsmfont{\the\glstshortpltok}}},%
8886     text={\protect\glstabbrvsmfont{\the\glstshorttok}},%
8887     plural={\protect\glstabbrvsmfont{\the\glstshortpltok}}%
8888   }%

```

Unset the regular attribute if it has been set.

```

8889 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8890   \glsthasattribute{\the\glstlabeltok}{regular}%
8891   {%
8892     \glstsetattribute{\the\glstlabeltok}{regular}{false}%
8893   }%
8894   {}%
8895 }%
8896 }%
8897 {%

```

As long-short-sm style:

```

8898 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8899 }

```

short-sm-long Now the short (long) version

```

8900 \newabbreviationstyle{short-sm-long}%
8901 {%
8902   \renewcommand*{\CustomAbbreviationFields}{%
8903     name={\glstxtrshortlongname},
8904     sort={\the\glstshorttok},
8905     description={\the\glstlongtok},%
8906     first={\protect\glstfirstabbrvsmfont{\the\glstshorttok}%
8907       \protect\glstxtrfullsep{\the\glstlabeltok}%
8908       \glstxtrparen{\protect\glstfirstlongdefaultfont{\the\glstlongtok}}},%
8909     firstplural={\protect\glstfirstabbrvsmfont{\the\glstshortpltok}%

```

```

8910 \protect\glxtrfullsep{\the\glslabeltok}%
8911 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8912 plural={\protect\glssabrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8913 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8914 \glshasattribute{\the\glslabeltok}{regular}%
8915 {%
8916 \glissetattribute{\the\glslabeltok}{regular}{false}%
8917 }%
8918 {}%
8919 }%
8920}%
8921{%
8922 \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
8923 \renewcommand*\glsfirstabrvfont[1]{\glsfirstabrvsmfont{##1}}%
8924 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8925 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8926 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8927 \renewcommand*{\glxtrfullformat}[2]{%
8928 \glsfirstabrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8929 \ifglxtrininsertinside\else##2\fi
8930 \glxtrfullsep{##1}%
8931 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8932}%
8933 \renewcommand*{\glxtrfullplformat}[2]{%
8934 \glsfirstabrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8935 \ifglxtrininsertinside\else##2\fi
8936 \glxtrfullsep{##1}%
8937 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8938}%
8939 \renewcommand*{\Glsxtrfullformat}[2]{%
8940 \glsfirstabrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8941 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8942 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8943}%
8944 \renewcommand*{\Glsxtrfullplformat}[2]{%
8945 \glsfirstabrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8946 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8947 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8948}%
8949}

```

rt-sm-long-desc As before but user provides description

```

8950 \newabbreviationstyle{short-sm-long-desc}%
8951 {%
8952 \renewcommand*{\CustomAbbreviationFields}{%
8953 name={\glxtrshortlongdescname},

```

```

8954     sort={\glxtrshortlongdescsort},
8955     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8956       \protect\glxtrfullsep{\the\glslabeltok}%
8957       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8958     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8959       \protect\glxtrfullsep{\the\glslabeltok}%
8960       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8961     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8962     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8963   }%

```

Unset the regular attribute if it has been set.

```

8964   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8965     \glshasattribute{\the\glslabeltok}{regular}%
8966     {%
8967       \glssetattribute{\the\glslabeltok}{regular}{false}%
8968     }%
8969   }%
8970 }%
8971 }%
8972 {%

```

As short-sm-long style:

```

8973   \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8974 }

```

short-sm

```

8975 \newabbreviationstyle{short-sm}%
8976 {%
8977   \renewcommand*{\CustomAbbreviationFields}{%
8978     name={\glxtrshortnolongname},
8979     sort={\the\glsshorttok},
8980     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8981     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8982     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8983     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8984     description={\the\glslongtok}}%
8985   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8986     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8987 }%
8988 {%
8989   \renewcommand*{\glsabbrvfont[1]}{\glsabbrvsmfont{##1}}%
8990   \renewcommand*{\glsfirstabbrvfont[1]}{\glsfirstabbrvsmfont{##1}}%
8991   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8992   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8993   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8994   \renewcommand*{\glxtrinlinefullformat}[2]{%
8995     \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%

```

```

8996     \ifglxtrinsertinside##2\fi}%
8997     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8998     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8999 }%
9000 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9001     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
9002         \ifglxtrinsertinside##2\fi}%
9003     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9004     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9005 }%

9006 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9007     \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
9008         \ifglxtrinsertinside##2\fi}%
9009     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9010     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9011 }%
9012 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9013     \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
9014         \ifglxtrinsertinside##2\fi}%
9015     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9016     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9017 }%

```

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.

```

9018 \renewcommand*{\glxtrfullformat}[2]{%
9019     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9020     \ifglxtrinsertinside\else##2\fi
9021 }%
9022 \renewcommand*{\glxtrfullplformat}[2]{%
9023     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9024     \ifglxtrinsertinside\else##2\fi
9025 }%
9026 \renewcommand*{\Glsxtrfullformat}[2]{%
9027     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9028     \ifglxtrinsertinside\else##2\fi
9029 }%
9030 \renewcommand*{\Glsxtrfullplformat}[2]{%
9031     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9032     \ifglxtrinsertinside\else##2\fi
9033 }%
9034 }

```

short-sm-nolong

```
9035 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
9036 \newabbreviationstyle{short-sm-desc}%

```

```

9037 {%
9038   \renewcommand*{\CustomAbbreviationFields}{%
9039     name={\glxtrshortdescname},
9040     sort={\the\glsshorttok},
9041     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
9042     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
9043     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
9044     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
9045   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9046     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9047 }%
9048 {%
9049   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9050   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9051   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9052   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9053   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9054   \renewcommand*{\glxtrinlinefullformat}[2]{%
9055     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9056     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9057     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9058   }%
9059   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9060     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9061     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9062     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9063   }%
9064   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9065     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9066     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9067     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9068   }%
9069   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9070     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9071     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9072     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9073   }%

```

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.

```

9074   \renewcommand*{\glxtrfullformat}[2]{%
9075     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9076     \ifglxtrininsertinside\else##2\fi
9077   }%
9078   \renewcommand*{\glxtrfullplformat}[2]{%
9079     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9080     \ifglxtrininsertinside\else##2\fi
9081   }%

```

```

9082 \renewcommand*{\Glsxtrfullformat}[2]{%
9083   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9084   \ifglxtrinsertinside\else##2\fi
9085 }%
9086 \renewcommand*{\Glsxtrfullplformat}[2]{%
9087   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9088   \ifglxtrinsertinside\else##2\fi
9089 }%
9090 }

```

-sm-nolong-desc

```

9091 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

9092 \newabbreviationstyle{nolong-short-sm}%
9093 {%
9094   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9095 }%
9096 {%
9097   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9098 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9099   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9100     \ifglxtrinsertinside##2\fi}%
9101   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9102   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9103 }%
9104 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9105   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9106     \ifglxtrinsertinside##2\fi}%
9107   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9108   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9109 }%
9110 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9111   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9112     \ifglxtrinsertinside##2\fi}%
9113   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9114   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9115 }%
9116 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9117   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9118     \ifglxtrinsertinside##2\fi}%
9119   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9120   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9121 }%
9122 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsshort.

```

9123 \newabbreviationstyle{long-noshort-sm}%
9124 {%
9125   \renewcommand*{\CustomAbbreviationFields}{%
9126     name={\glxtrlongnoshortname},
9127     sort={\the\glsshorttok},
9128     first={\protect\glfirstlongdefaultfont{\the\glslongtok}},
9129     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}},
9130     text={\protect\glslongdefaultfont{\the\glslongtok}},
9131     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9132     description={\the\glslongtok}%
9133   }%
9134   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9135     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9136 }%
9137 {%
9138   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
9139   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvsmfont{##1}}%
9140   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9141   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9142   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9143   \renewcommand*{\glxtrsubsequentfmt}[2]{%
9144     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9145     \ifglxtrininsertinside \else##2\fi
9146   }%
9147   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9148     \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9149     \ifglxtrininsertinside \else##2\fi
9150   }%
9151   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9152     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9153     \ifglxtrininsertinside \else##2\fi
9154   }%
9155   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9156     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9157     \ifglxtrininsertinside \else##2\fi
9158   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9159   \renewcommand*{\glxtrinlinefullformat}[2]{%
9160     \glfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9161     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9162     \glxtrparen{\protect\glfirstabbrvsmfont{\glssaccessshort{##1}}}%
9163   }%
9164   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9165     \glfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9166     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9167     \glxtrparen{\protect\glfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
9168   }%

```

```

9169 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9170   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9171   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9172   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
9173 }%
9174 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9175   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9176   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9177   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
9178 }%

```

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.

```

9179 \renewcommand*{\glxtrfullformat}[2]{%
9180   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9181   \ifglxtrinsertinside\else##2\fi
9182 }%
9183 \renewcommand*{\glxtrfullplformat}[2]{%
9184   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9185   \ifglxtrinsertinside\else##2\fi
9186 }%
9187 \renewcommand*{\Glsxtrfullformat}[2]{%
9188   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9189   \ifglxtrinsertinside\else##2\fi
9190 }%
9191 \renewcommand*{\Glsxtrfullplformat}[2]{%
9192   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9193   \ifglxtrinsertinside\else##2\fi
9194 }%
9195 }

```

long-sm Backward compatibility:

```

9196 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9197 \newabbreviationstyle{long-noshort-sm-desc}%
9198 {%
9199   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9200 }%
9201 {%
9202   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9203   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9204   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
9205   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9206   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9207 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9208   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%

```

```

9209 \ifglxtrinsertinside \else##2\fi
9210 }%
9211 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9212 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9213 \ifglxtrinsertinside \else##2\fi
9214 }%
9215 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9216 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9217 \ifglxtrinsertinside \else##2\fi
9218 }%
9219 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9220 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9221 \ifglxtrinsertinside \else##2\fi
9222 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9223 \renewcommand*{\glxtrinlinefullformat}[2]{%
9224 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9225 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9226 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9227 }%
9228 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9229 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9230 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9231 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9232 }%
9233 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9234 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9235 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9236 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9237 }%
9238 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9239 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9240 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9241 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9242 }%

```

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.

```

9243 \renewcommand*{\glxtrfullformat}[2]{%
9244 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9245 \ifglxtrinsertinside\else##2\fi
9246 }%
9247 \renewcommand*{\glxtrfullplformat}[2]{%
9248 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9249 \ifglxtrinsertinside\else##2\fi
9250 }%
9251 \renewcommand*{\Glsxtrfullformat}[2]{%
9252 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9253 \ifglxtrinsertinside\else##2\fi

```

```

9254 }%
9255 \renewcommand*{\Glsxtrfullplformat}[2]{%
9256   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9257   \ifglxtrinsertinside\else##2\fi
9258 }%
9259 }

```

long-desc-sm Backward compatibility:

```

9260 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9261 \newabbreviationstyle{short-sm-footnote}%
9262 {%
9263   \renewcommand*{\CustomAbbreviationFields}{%
9264     name={\glxtrfootnotename},
9265     sort={\the\glsshorttok},
9266     description={\the\glslongtok},%
9267     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9268       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9269       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9270     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9271       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9272       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9273     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

9274 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9275   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9276   \glsattribute{\the\glslabeltok}{regular}%
9277   {%
9278     \glssetattribute{\the\glslabeltok}{regular}{false}%
9279   }%
9280   {}%
9281 }%
9282 }%
9283 {%
9284   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9285   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9286   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
9287   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9288   \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9289 \renewcommand*{\glxtrfullformat}[2]{%
9290   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9291   \ifglxtrinsertinside\else##2\fi
9292   \protect\glxtrabbrvfootnote{##1}%
9293   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9294 }%

```

```

9295 \renewcommand*{\glxtrfullplformat}[2]{%
9296   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9297   \ifglxtrininsertinside\else##2\fi
9298   \protect\glxtrabbrvfootnote{##1}%
9299   {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9300 }%
9301 \renewcommand*{\Glsxtrfullformat}[2]{%
9302   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9303   \ifglxtrininsertinside\else##2\fi
9304   \protect\glxtrabbrvfootnote{##1}%
9305   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9306 }%
9307 \renewcommand*{\Glsxtrfullplformat}[2]{%
9308   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9309   \ifglxtrininsertinside\else##2\fi
9310   \protect\glxtrabbrvfootnote{##1}%
9311   {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9312 }%

```

The first use full form and the inline full form use the short (long) style.

```

9313 \renewcommand*{\glxtrininlinefullformat}[2]{%
9314   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9315   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9316   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9317 }%
9318 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9319   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9320   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9321   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9322 }%
9323 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9324   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9325   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9326   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9327 }%
9328 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9329   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9330   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9331   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9332 }%
9333 }

```

footnote-sm Backward compatibility:

```

9334 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

9335 \newabbreviationstyle{short-sm-postfootnote}%
9336 {%
9337   \renewcommand*{\CustomAbbreviationFields}{%
9338     name={\glxtrfootnotename},

```

```

9339     sort={\the\glsshorttok},
9340     description={\the\glslongtok},%
9341     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9342     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9343     plural={\protect\glsabbrvsmfont{\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.

```

9344 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9345   \csdef{glsxtrpostlink\glscategorylabel}{%
9346     \glsxtrifwasfirstuse
9347     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9348       \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
9349       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
9350   }%
9351   {%
9352   }%
9353   \glsattribute{\the\glslabeltok}{regular}%
9354   {%
9355     \glssetattribute{\the\glslabeltok}{regular}{false}%
9356   }%
9357   {}%
9358 }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

9359 \renewcommand*{\glsxtrsetupfulldefs}{%
9360   \let\glsxtrifwasfirstuse\@secondoftwo
9361 }%
9362 }%
9363 {%
9364 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9365 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9366 \renewcommand*\glsabbrvpluralsuffix{\protect\glsxtrrmsuffix}%
9367 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9368 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9369 \renewcommand*\glsxtrfullformat[2]{%
9370   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9371   \ifglsxtrininsertinside\else##2\fi
9372 }%
9373 \renewcommand*\glsxtrfullplformat[2]{%
9374   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9375   \ifglsxtrininsertinside\else##2\fi
9376 }%
9377 \renewcommand*\Glsxtrfullformat[2]{%
9378   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%

```

```

9379 \ifglxtrinsertinside\else##2\fi
9380 }%
9381 \renewcommand*{\Glsxtrfullplformat}[2]{%
9382 \glstfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9383 \ifglxtrinsertinside\else##2\fi
9384 }%

```

The first use full form and the inline full form use the short (long) style.

```

9385 \renewcommand*{\glxtrinlinefullformat}[2]{%
9386 \glstfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9387 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9388 \glxtrparen{\glstfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9389 }%
9390 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9391 \glstfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9392 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9393 \glxtrparen{\glstfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9394 }%
9395 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9396 \glstfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9397 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9398 \glxtrparen{\glstfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9399 }%
9400 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9401 \glstfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9402 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9403 \glxtrparen{\glstfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9404 }%
9405 }

```

postfootnote-sm Backward compatibility:

```

9406 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glssabbrvemfont`

```

9407 \newcommand*{\glssabbrvemfont}[1]{\emph{##1}}%

```

`\glstfirstabbrvemfont`

```

9408 \newcommand*{\glstfirstabbrvemfont}[1]{\glssabbrvemfont{##1}}%

```

The default short form suffix:

`\glstxtremsuffix`

```

9409 \newcommand*{\glstxtremsuffix}{\glstxttrabbrvpluralsuffix}

```

`\glstfirstlongemfont` Only used by the “long-em” styles.

```

9410 \newcommand*{\glstfirstlongemfont}[1]{\glstlongemfont{##1}}%

```

`\glslongemfont` Only used by the “long-em” styles.

```
9411 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9412 \newabbreviationstyle{long-short-em}%
9413 {%
9414   \renewcommand*{\CustomAbbreviationFields}{%
9415     name={\glsxtrlongshortname},
9416     sort={\the\glsshorttok},
9417     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9418       \protect\glsxtrfullsep{\the\glslabeltok}%
9419       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9420     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9421       \protect\glsxtrfullsep{\the\glslabeltok}%
9422       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9423     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9424     description={\the\glslongtok}}%
9425   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9426     \glsattribute{\the\glslabeltok}{regular}%
9427     {%
9428       \glssetattribute{\the\glslabeltok}{regular}{false}%
9429     }%
9430   }%
9431 }%
9432 }%
9433 {%
9434   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9435   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9436   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9437   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9438   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

Use the default long fonts.

```
9437 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9438 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9439 \renewcommand*{\glsxtrfullformat}[2]{%
9440   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9441   \ifglsxtrininsertinside\else##2\fi
9442   \glsxtrfullsep{##1}%
9443   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9444 }%
9445 \renewcommand*{\glsxtrfullplformat}[2]{%
9446   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9447   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9448   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9449 }%
9450 \renewcommand*{\Glsxtrfullformat}[2]{%
9451   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9452   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9453   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9454 }
```

```

9454 }%
9455 \renewcommand*{\Glsxtrfullplformat}[2]{%
9456   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9457   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9458   \glxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9459 }%
9460 }

```

g-short-em-desc

```

9461 \newabbreviationstyle{long-short-em-desc}%
9462 {%
9463   \renewcommand*{\CustomAbbreviationFields}{%
9464     name={\glxtrlongshortdescname},
9465     sort={\glxtrlongshortdescsort},%
9466     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9467       \protect\glxtrfullsep{\the\glslabeltok}%
9468       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9469     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9470       \protect\glxtrfullsep{\the\glslabeltok}%
9471       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9472     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9473     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9474   }%

```

Unset the regular attribute if it has been set.

```

9475   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9476     \glshasattribute{\the\glslabeltok}{regular}%
9477     {%
9478       \glissetattribute{\the\glslabeltok}{regular}{false}%
9479     }%
9480   }%
9481 }%
9482 }%
9483 {%

```

As long-short-em style:

```

9484   \GlsXtrUseAbbrStyleFmts{long-short-em}%
9485 }

```

ong-em-short-em

```

9486 \newabbreviationstyle{long-em-short-em}%
9487 {%
9488   \glslongemfont is used in the description since \glsdesc doesn't set the style.
9488   \renewcommand*{\CustomAbbreviationFields}{%
9489     name={\glxtrlongshortname},
9490     sort={\the\glsshorttok},
9491     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9492       \protect\glxtrfullsep{\the\glslabeltok}%
9493       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%

```

```

9494   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}}%
9495   \protect\glsxtrfullsep{\the\glslabeltok}}%
9496   \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9497   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9498   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9499   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9500     \glsattribute{\the\glslabeltok}{regular}}%
9501   {%
9502     \glssetattribute{\the\glslabeltok}{regular}{false}}%
9503   }%
9504   {}%
9505 }%
9506 }%
9507 {%
9508   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9509   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9510   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9511   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9512   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9513   \renewcommand*{\glsxtrfullformat}[2]{%
9514     \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9515     \ifglsxtrininsertinside\else##2\fi
9516     \glsxtrfullsep{##1}}%
9517     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9518   }%
9519   \renewcommand*{\glsxtrfullplformat}[2]{%
9520     \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9521     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
9522     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9523   }%
9524   \renewcommand*{\Glsxtrfullformat}[2]{%
9525     \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9526     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
9527     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9528   }%
9529   \renewcommand*{\Glsxtrfullplformat}[2]{%
9530     \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9531     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
9532     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9533   }%
9534 }

```

m-short-em-desc

```

9535 \newabbreviationstyle{long-em-short-em-desc}%
9536 {%

```

```

9537 \renewcommand*{\CustomAbbreviationFields}{%
9538   name={\glxtrlongshortdescname},
9539   sort={\glxtrlongshortdescsort},%
9540   first={\protect\glsfirstlongemfont{\the\glslongtok}%
9541     \protect\glxtrfullsep{\the\glslabeltok}%
9542     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9543   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9544     \protect\glxtrfullsep{\the\glslabeltok}%
9545     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9546   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9547   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9548 }%

```

Unset the regular attribute if it has been set.

```

9549 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9550   \glshasattribute{\the\glslabeltok}{regular}%
9551   {%
9552     \glissetattribute{\the\glslabeltok}{regular}{false}%
9553   }%
9554   {}%
9555 }%
9556 }%
9557 {%
9558   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9559 }

```

short-em-long Now the short (long) version

```

9560 \newabbreviationstyle{short-em-long}%
9561 {%
9562   \renewcommand*{\CustomAbbreviationFields}{%
9563     name={\glxtrshortlongname},
9564     sort={\the\glsshorttok},
9565     description={\the\glslongtok},%
9566     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9567       \protect\glxtrfullsep{\the\glslabeltok}%
9568       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9569     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9570       \protect\glxtrfullsep{\the\glslabeltok}%
9571       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9572     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9573 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9574   \glshasattribute{\the\glslabeltok}{regular}%
9575   {%
9576     \glissetattribute{\the\glslabeltok}{regular}{false}%
9577   }%
9578   {}%
9579 }%
9580 }%

```

9581 {%

Mostly as short-long style:

```
9582 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9583 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9584 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9585 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9586 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9587 \renewcommand*{\glsxtrfullformat}[2]{%
9588   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9589   \ifglsxtrininsertinside\else##2\fi
9590   \glsxtrfullsep{##1}%
9591   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9592 }%
9593 \renewcommand*{\glsxtrfullplformat}[2]{%
9594   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9595   \ifglsxtrininsertinside\else##2\fi
9596   \glsxtrfullsep{##1}%
9597   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9598 }%
9599 \renewcommand*{\Glsxtrfullformat}[2]{%
9600   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9601   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9602   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9603 }%
9604 \renewcommand*{\Glsxtrfullplformat}[2]{%
9605   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9606   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9607   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9608 }%
9609 }
```

rt-em-long-desc As before but user provides description

```
9610 \newabbreviationstyle{short-em-long-desc}%
9611 {%
9612   \renewcommand*{\CustomAbbreviationFields}{%
9613     name={\glsxtrshortlongdescname},
9614     sort={\glsxtrshortlongdescsort},
9615     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9616     \protect\glsxtrfullsep{\the\glslabeltok}%
9617     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9618     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9619     \protect\glsxtrfullsep{\the\glslabeltok}%
9620     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9621     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9622     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9623   }%
```

Unset the regular attribute if it has been set.

```

9624 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9625   \glshasattribute{\the\glslabeltok}{regular}%
9626   {%
9627     \glissetattribute{\the\glslabeltok}{regular}{false}%
9628   }%
9629   {}%
9630 }%
9631 }%
9632 {%
9633   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9634 }

```

hort-em-long-em

```

9635 \newabbreviationstyle{short-em-long-em}%
9636 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9637 \renewcommand*{\CustomAbbreviationFields}{%
9638   name={\glsxtrshortlongname},
9639   sort={\the\glsshorttok},
9640   description={\protect\glslongemfont{\the\glslongtok}},%
9641   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9642     \protect\glsxtrfullsep{\the\glslabeltok}%
9643     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9644   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9645     \protect\glsxtrfullsep{\the\glslabeltok}%
9646     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9647   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9648 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9649   \glshasattribute{\the\glslabeltok}{regular}%
9650   {%
9651     \glissetattribute{\the\glslabeltok}{regular}{false}%
9652   }%
9653   {}%
9654 }%
9655 }%
9656 {%
9657   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9658   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9659   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9660   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9661   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9662 \renewcommand*{\glsxtrfullformat}[2]{%
9663   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9664   \ifglsxtrininsertinside\else##2\fi
9665   \glsxtrfullsep{##1}%

```

```

9666 \glstrparen{\glfirstlongemfont{\glsaccesslong{##1}}}%
9667 }%
9668 \renewcommand*{\glstrfullplformat}[2]{%
9669 \glfirstabbrvemfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9670 \ifglstrinsertinside\else##2\fi
9671 \glstrfullsep{##1}%
9672 \glstrparen{\glfirstlongemfont{\glsaccesslongpl{##1}}}%
9673 }%
9674 \renewcommand*{\Glsxtrfullformat}[2]{%
9675 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9676 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9677 \glstrparen{\glfirstlongemfont{\glsaccesslong{##1}}}%
9678 }%
9679 \renewcommand*{\Glsxtrfullplformat}[2]{%
9680 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9681 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9682 \glstrparen{\glfirstlongemfont{\glsaccesslongpl{##1}}}%
9683 }%
9684 }

```

em-long-em-desc

```

9685 \newabbreviationstyle{short-em-long-em-desc}%
9686 {%
9687 \renewcommand*{\CustomAbbreviationFields}{%
9688 name={\glstrshortlongdescname},%
9689 sort={\glstrshortlongdescsort},%
9690 first={\protect\glfirstabbrvemfont{\the\glsshorttok}%
9691 \protect\glstrfullsep{\the\glslabeltok}%
9692 \glstrparen{\protect\glfirstlongemfont{\the\glslongtok}}},%
9693 firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}%
9694 \protect\glstrfullsep{\the\glslabeltok}%
9695 \glstrparen{\protect\glfirstlongemfont{\the\glslongpltok}}},%
9696 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9697 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9698 }%

```

Unset the regular attribute if it has been set.

```

9699 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9700 \glshasattribute{\the\glslabeltok}{regular}%
9701 {%
9702 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9703 }%
9704 {}%
9705 }%
9706 }%
9707 {%
9708 \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9709 }

```

short-em

```

9710 \newabbreviationstyle{short-em}%
9711 {%
9712   \renewcommand*{\CustomAbbreviationFields}{%
9713     name={\glxtrshortnolongname},
9714     sort={\the\glsshorttok},
9715     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9716     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9717     text={\protect\glsabbrvemfont{\the\glsshorttok}},
9718     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9719     description={\the\glslongtok}}%
9720   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9721     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9722 }%
9723 {%
9724   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9725   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9726   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9727   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9728   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9729   \renewcommand*{\glxtrinlinefullformat}[2]{%
9730     \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
9731     \ifglxtrininsertinside##2\fi}%
9732   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9733   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9734 }%
9735   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9736     \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}%
9737     \ifglxtrininsertinside##2\fi}%
9738   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9739   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9740 }%

9741   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9742     \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}%
9743     \ifglxtrininsertinside##2\fi}%
9744   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9745   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9746 }%
9747   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9748     \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}%
9749     \ifglxtrininsertinside##2\fi}%
9750   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9751   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9752 }%

```

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.

```

9753   \renewcommand*{\glxtrfullformat}[2]{%

```

```

9754 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9755 \ifglxtrinsertinside\else##2\fi
9756 }%
9757 \renewcommand*{\glxtrfullplformat}[2]{%
9758 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9759 \ifglxtrinsertinside\else##2\fi
9760 }%
9761 \renewcommand*{\Glsxtrfullformat}[2]{%
9762 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9763 \ifglxtrinsertinside\else##2\fi
9764 }%
9765 \renewcommand*{\Glsxtrfullplformat}[2]{%
9766 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9767 \ifglxtrinsertinside\else##2\fi
9768 }%
9769 }

```

short-em-nolong

```

9770 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9771 \newabbreviationstyle{short-em-desc}%
9772 {%
9773 \renewcommand*{\CustomAbbreviationFields}{%
9774 name={\glxtrshortdescname},
9775 sort={\the\glsshorttok},
9776 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9777 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9778 text={\protect\glsabbrvemfont{\the\glsshorttok}},
9779 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9780 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9781 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9782 }%
9783 {%
9784 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9785 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9786 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9787 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9788 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9789 \renewcommand*{\glxtrinlinefullformat}[2]{%
9790 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9791 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9792 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9793 }%
9794 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9795 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9796 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9797 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%

```

```

9798 }%
9799 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9800   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9801   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9802   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9803 }%
9804 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9805   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9806   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9807   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9808 }%

```

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.

```

9809 \renewcommand*{\glxtrfullformat}[2]{%
9810   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9811   \ifglxtrinsertinside\else##2\fi
9812 }%
9813 \renewcommand*{\glxtrfullplformat}[2]{%
9814   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9815   \ifglxtrinsertinside\else##2\fi
9816 }%
9817 \renewcommand*{\Glsxtrfullformat}[2]{%
9818   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9819   \ifglxtrinsertinside\else##2\fi
9820 }%
9821 \renewcommand*{\Glsxtrfullplformat}[2]{%
9822   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9823   \ifglxtrinsertinside\else##2\fi
9824 }%
9825 }

```

-em-nolong-desc

```

9826 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9827 \newabbreviationstyle{nolong-short-em}%
9828 {%
9829   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9830 }%
9831 {%
9832   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9833 \renewcommand*{\glxtrinlinefullformat}[2]{%
9834   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9835     \ifglxtrinsertinside##2\fi}%
9836   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9837   \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
9838 }%

```

```

9839 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9840   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9841     \ifglxtrinsertinside##2\fi}%
9842   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9843   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9844 }%
9845 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9846   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9847     \ifglxtrinsertinside##2\fi}%
9848   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9849   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9850 }%
9851 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9852   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9853     \ifglxtrinsertinside##2\fi}%
9854   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9855   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9856 }%
9857 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9858 \newabbreviationstyle{long-noshort-em}%
9859 {%
9860   \renewcommand*{\CustomAbbreviationFields}{%
9861     name={\glxtrlongnoshortname},
9862     sort={\the\glsshorttok},
9863     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9864     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9865     text={\protect\glslongdefaultfont{\the\glslongtok}},
9866     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9867     description={\the\glslongtok}%
9868   }%
9869   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9870     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9871 }%
9872 {%
9873   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9874   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9875   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9876   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9877   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9878 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9879   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9880   \ifglxtrinsertinside \else##2\fi
9881 }%
9882 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9883   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9884   \ifglxtrinsertinside \else##2\fi

```

```

9885 }%
9886 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9887   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9888   \ifglxtrinsertinside \else##2\fi
9889 }%
9890 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9891   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9892   \ifglxtrinsertinside \else##2\fi
9893 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9894 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9895   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9896   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9897   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9898 }%
9899 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9900   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9901   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9902   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9903 }%
9904 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9905   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9906   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9907   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9908 }%
9909 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9910   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9911   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9912   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9913 }%

```

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.

```

9914 \renewcommand*{\Glsxtrfullformat}[2]{%
9915   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9916   \ifglxtrinsertinside\else##2\fi
9917 }%
9918 \renewcommand*{\Glsxtrfullplformat}[2]{%
9919   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9920   \ifglxtrinsertinside\else##2\fi
9921 }%
9922 \renewcommand*{\Glsxtrfullformat}[2]{%
9923   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9924   \ifglxtrinsertinside\else##2\fi
9925 }%
9926 \renewcommand*{\Glsxtrfullplformat}[2]{%
9927   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9928   \ifglxtrinsertinside\else##2\fi
9929 }%

```

9930 }

long-em Backward compatibility:

9931 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
9932 \newabbreviationstyle{long-em-noshort-em}%
9933 {%
9934   \renewcommand*{\CustomAbbreviationFields}{%
9935     name={\glxtrlongnoshortname},
9936     sort={\the\glsshorttok},
9937     first={\protect\glslongemfont{\the\glslongtok}},
9938     firstplural={\protect\glslongemfont{\the\glslongpltok}},
9939     text={\protect\glslongemfont{\the\glslongtok}},
9940     plural={\protect\glslongemfont{\the\glslongpltok}},%
9941     description={\protect\glslongemfont{\the\glslongtok}}%
9942   }%
9943   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9944     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9945 }%
9946 {%
9947   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9948   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9949   \renewcommand*{\glslfirstabbrvfont}[1]{\glslfirstabbrvemfont{##1}}%
9950   \renewcommand*{\glslfirstlongfont}[1]{\glslfirstlongemfont{##1}}%
9951   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9952 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9953   \glslongemfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9954   \ifglxtrininsertinside \else##2\fi
9955 }%
9956 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9957   \glslongemfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9958   \ifglxtrininsertinside \else##2\fi
9959 }%
9960 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9961   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9962   \ifglxtrininsertinside \else##2\fi
9963 }%
9964 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9965   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9966   \ifglxtrininsertinside \else##2\fi
9967 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9968 \renewcommand*{\glxtrinlinefullformat}[2]{%
9969   \glslfirstlongemfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9970   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9971   \glxtrparen{\protect\glslfirstabbrvemfont{\glssaccessshort{##1}}}%

```

```

9972 }%
9973 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9974   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9975   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9976   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9977 }%
9978 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9979   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9980   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9981   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9982 }%
9983 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9984   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9985   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9986   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9987 }%

```

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.

```

9988 \renewcommand*{\glxstrfullformat}[2]{%
9989   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9990   \ifglxstrinsertinside\else##2\fi
9991 }%
9992 \renewcommand*{\glxstrfullplformat}[2]{%
9993   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9994   \ifglxstrinsertinside\else##2\fi
9995 }%
9996 \renewcommand*{\Glsxtrfullformat}[2]{%
9997   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9998   \ifglxstrinsertinside\else##2\fi
9999 }%
10000 \renewcommand*{\Glsxtrfullplformat}[2]{%
10001   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
10002   \ifglxstrinsertinside\else##2\fi
10003 }%
10004 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

10005 \newabbreviationstyle{long-em-noshort-em-noreg}%
10006 {%
10007   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%

```

Unset the regular attribute if it has been set.

```

10008 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10009   \glshasattribute{\the\glslabeltok}{regular}%
10010   {%
10011     \glssetattribute{\the\glslabeltok}{regular}{false}%
10012   }%
10013   {}%
10014 }%

```

```

10015 }%
10016 {%
10017   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
10018 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glssshort`.

```

10019 \newabbreviationstyle{long-noshort-em-desc}%
10020 {%
10021   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
10022 }%
10023 {%
10024   \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
10025   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
10026   \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
10027   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
10028   \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10029 \renewcommand*{\glssxtrsubsequentfmt}[2]{%
10030   \glsslongdefaultfont{\glssaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
10031   \ifglssxtrininsertinside \else##2\fi
10032 }%
10033 \renewcommand*{\glssxtrsubsequentplfmt}[2]{%
10034   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
10035   \ifglssxtrininsertinside \else##2\fi
10036 }%
10037 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10038   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
10039   \ifglssxtrininsertinside \else##2\fi
10040 }%
10041 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10042   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
10043   \ifglssxtrininsertinside \else##2\fi
10044 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10045 \renewcommand*{\glssxtrinlinefullformat}[2]{%
10046   \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
10047   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10048   \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
10049 }%
10050 \renewcommand*{\glssxtrinlinefullplformat}[2]{%
10051   \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
10052   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10053   \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10054 }%
10055 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10056   \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
10057   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%

```

```

10058 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10059 }%
10060 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10061 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10062 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10063 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10064 }%

```

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.

```

10065 \renewcommand*{\glxtrfullformat}[2]{%
10066 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10067 \ifglxtrinsertinside\else##2\fi
10068 }%
10069 \renewcommand*{\glxtrfullplformat}[2]{%
10070 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10071 \ifglxtrinsertinside\else##2\fi
10072 }%
10073 \renewcommand*{\Glsxtrfullformat}[2]{%
10074 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10075 \ifglxtrinsertinside\else##2\fi
10076 }%
10077 \renewcommand*{\Glsxtrfullplformat}[2]{%
10078 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10079 \ifglxtrinsertinside\else##2\fi
10080 }%
10081 }

```

long-desc-em Backward compatibility:

```

10082 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}

```

noshort-em-desc The short form is explicitly invoked through commands like `\glxtrshort`. The long form is emphasized.

```

10083 \newabbreviationstyle{long-em-noshort-em-desc}%
10084 {%
10085 \renewcommand*{\CustomAbbreviationFields}{%
10086 name={\glxtrlongnoshortdescname},
10087 sort={\the\glslongtok},
10088 first={\protect\glsfirstlongemfont{\the\glslongtok}},
10089 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10090 text={\glslongemfont{\the\glslongtok}},
10091 plural={\glslongemfont{\the\glslongpltok}}}%
10092 }%
10093 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10094 \glssetattribute{\the\glslabeltok}{regular}{true}}%
10095 }%
10096 {%
10097 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
10098 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%

```

```

10099 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10100 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10101 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10102 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10103   \glslongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
10104   \ifglsxtrinsertinside \else##2\fi
10105 }%
10106 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10107   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
10108   \ifglsxtrinsertinside \else##2\fi
10109 }%
10110 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10111   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
10112   \ifglsxtrinsertinside \else##2\fi
10113 }%
10114 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10115   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
10116   \ifglsxtrinsertinside \else##2\fi
10117 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10118 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10119   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10120   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10121   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10122 }%
10123 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10124   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10125   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10126   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10127 }%
10128 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10129   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10130   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10131   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10132 }%
10133 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10134   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10135   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10136   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10137 }%

```

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.

```

10138 \renewcommand*{\glsxtrfullformat}[2]{%
10139   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10140   \ifglsxtrinsertinside\else##2\fi
10141 }%
10142 \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

10143 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10144 \ifglxtrinsertinside\else##2\fi
10145 }%
10146 \renewcommand*{\Glsxtrfullformat}[2]{%
10147 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10148 \ifglxtrinsertinside\else##2\fi
10149 }%
10150 \renewcommand*{\Glsxtrfullplformat}[2]{%
10151 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10152 \ifglxtrinsertinside\else##2\fi
10153 }%
10154 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

10155 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10156 {%
10157 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
10158 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10159 \glshasattribute{\the\glslabeltok}{regular}%
10160 {%
10161 \glssetattribute{\the\glslabeltok}{regular}{false}%
10162 }%
10163 {}%
10164 }%
10165 }%
10166 {%
10167 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10168 }

```

ort-em-footnote

```

10169 \newabbreviationstyle{short-em-footnote}%
10170 {%
10171 \renewcommand*{\CustomAbbreviationFields}{%
10172 name={\glxtrfootnotename},
10173 sort={\the\glsshorttok},
10174 description={\the\gslongtok},%
10175 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
10176 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10177 {\protect\glsfirstlongfootnotefont{\the\gslongtok}}},%
10178 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
10179 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10180 {\protect\glsfirstlongfootnotefont{\the\gslongpltok}}},%
10181 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10182 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10183 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

10184 \glshasattribute{\the\glslabelltok}{regular}%
10185 {%
10186 \glissetattribute{\the\glslabelltok}{regular}{false}%
10187 }%
10188 {}%
10189 }%
10190 }%
10191 {%
10192 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
10193 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
10194 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
10195 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongfootnotefont{##1}}%
10196 \renewcommand*{\glsslongfont}[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10197 \renewcommand*{\glsxtrfullformat}[2]{%
10198 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10199 \ifglsxtrininsertinside\else##2\fi
10200 \protect\glsxtrabbrvfootnote{##1}%
10201 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10202 }%
10203 \renewcommand*{\glsxtrfullplformat}[2]{%
10204 \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10205 \ifglsxtrininsertinside\else##2\fi
10206 \protect\glsxtrabbrvfootnote{##1}%
10207 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10208 }%
10209 \renewcommand*{\Glsxtrfullformat}[2]{%
10210 \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10211 \ifglsxtrininsertinside\else##2\fi
10212 \protect\glsxtrabbrvfootnote{##1}%
10213 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10214 }%
10215 \renewcommand*{\Glsxtrfullplformat}[2]{%
10216 \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10217 \ifglsxtrininsertinside\else##2\fi
10218 \protect\glsxtrabbrvfootnote{##1}%
10219 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10220 }%

```

The first use full form and the inline full form use the short (long) style.

```

10221 \renewcommand*{\glsxtrininlinefullformat}[2]{%
10222 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10223 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10224 \glsxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10225 }%
10226 \renewcommand*{\glsxtrininlinefullplformat}[2]{%
10227 \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10228 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10229 \glsxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%

```

```

10230 }%
10231 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10232   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10233   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10234   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10235 }%
10236 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10237   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10238   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10239   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10240 }%
10241 }

```

footnote-em Backward compatibility:

```

10242 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

10243 \newabbreviationstyle{short-em-postfootnote}%
10244 {%
10245   \renewcommand*{\CustomAbbreviationFields}{%
10246     name={\glxtrfootnotename},
10247     sort={\the\glsshorttok},
10248     description={\the\glslongtok},%
10249     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
10250     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
10251     plural={\protect\glsabbrvemfont{\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.

```

10252 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10253   \csdef{glxtrpostlink\glscategorylabel}{%
10254     \glxtrifwasfirstuse
10255     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

10256       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10257       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
10258       }%
10259     }%
10260   }%
10261   \glshasattribute{\the\glslabeltok}{regular}%
10262   {%
10263     \glissetattribute{\the\glslabeltok}{regular}{false}%
10264   }%
10265   {}%
10266 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

10267 \renewcommand*{\glxtrsetupfulldefs}{%
10268   \let\glxtrifwasfirstuse\@secondoftwo
10269 }%
10270 }%
10271 {%
10272   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
10273   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10274   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10275   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
10276   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

10277 \renewcommand*{\glxtrfullformat}[2]{%
10278   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10279   \ifglxtrininsertinside\else##2\fi
10280 }%
10281 \renewcommand*{\glxtrfullplformat}[2]{%
10282   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10283   \ifglxtrininsertinside\else##2\fi
10284 }%
10285 \renewcommand*{\Glsxtrfullformat}[2]{%
10286   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10287   \ifglxtrininsertinside\else##2\fi
10288 }%
10289 \renewcommand*{\Glsxtrfullplformat}[2]{%
10290   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10291   \ifglxtrininsertinside\else##2\fi
10292 }%

```

The first use full form and the inline full form use the short (long) style.

```

10293 \renewcommand*{\glxtrininlinefullformat}[2]{%
10294   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10295   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10296   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10297 }%
10298 \renewcommand*{\glxtrininlinefullplformat}[2]{%
10299   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10300   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10301   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10302 }%
10303 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
10304   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10305   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10306   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10307 }%
10308 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
10309   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10310   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10311   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10312 }%

```

10313 }

postfootnote-em Backward compatibility:

10314 \@glxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}

1.7.6 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:

glxtruserfield Default is the useri field.

10315 \newcommand*{\glxtruserfield}{useri}

glxtruserparen 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.

10316 \ifdef\glscurrentfieldvalue

10317 {

10318 \newcommand*{\glxtruserparen}[2]{%

10319 \glxtrfullsep{#2}%

10320 \glxtrparen

10321 {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%

10322 }

10323 }

10324 {

10325 \newcommand*{\glxtruserparen}[2]{%

10326 \glxtrfullsep{#2}%

10327 \glxtrparen

10328 {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%

10329 }

10330 }

Font used for short form:

lsabbrvuserfont

10331 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

Font used for short form on first use:

stabbrvuserfont

10332 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

Font used for long form:

glslonguserfont

10333 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}

Font used for long form on first use:

rstlonguserfont

```
10334 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10335 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10336 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10337 \newabbreviationstyle{long-short-user}%
10338 {%
10339   \renewcommand*{\CustomAbbreviationFields}{%
10340     name={\glsxtrlongshortname},
10341     sort={\the\glsshorttok},
10342     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10343       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10344     {\the\glslabeltok}},%
10345     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10346       \protect\glsxtruserparen
10347       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10348     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10349     description={\protect\glsuserdescription{\the\glslongtok}%
10350       {\the\glslabeltok}}}%
10350 }
```

Unset the regular attribute if it has been set.

```
10351 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10352   \glshasattribute{\the\glslabeltok}{regular}%
10353   {%
10354     \glissetattribute{\the\glslabeltok}{regular}{false}%
10355   }%
10356   {}%
10357 }%
10358 }%
10359 }
```

In case the user wants to mix and match font styles, these are redefined here.

```
10360 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10361 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10362 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10363 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10364 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
10365 \renewcommand*{\glsxtrfullformat}[2]{%
10366   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsetinside##2\fi}}%
```

```

10367     \ifglxtrinsertinside\else##2\fi
10368     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10369 }%
10370 \renewcommand*{\glxtrfullplformat}[2]{%
10371     \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10372     \ifglxtrinsertinside\else##2\fi
10373     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10374 }%
10375 \renewcommand*{\Glsxtrfullformat}[2]{%
10376     \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10377     \ifglxtrinsertinside\else##2\fi
10378     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10379 }%
10380 \renewcommand*{\Glsxtrfullplformat}[2]{%
10381     \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10382     \ifglxtrinsertinside\else##2\fi
10383     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10384 }%
10385 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10386 \newabbreviationstyle{long-postshort-user}%
10387 {%
10388     \renewcommand*{\CustomAbbreviationFields}{%
10389         name={\glxtrlongshortname},
10390         sort={\the\glsshorttok},
10391         first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10392         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10393         plural={\protect\glssabbrvuserfont{\the\glssshortpltok}},%
10394         description={\protect\glssuserdescription{\the\glslongtok}%
10395             {\the\glslabeltok}}}%
10396     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10397         \csdef{glxtrpostlink\glscategorylabel}{%
10398             \glxtrifwasfirstuse
10399             {%
10400                 \glxtruserparen
10401                 {\glsfirstabbrvuserfont{\glssentryshort{\glslabel}}}%
10402                 {\glslabel}%
10403             }%
10404             {}%
10405         }%
10406         \glshasattribute{\the\glslabeltok}{regular}%
10407         {%
10408             \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10409         }%
10410         {}%
10411     }%
10412 }%
10413 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```
10414 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10415 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10416 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10417 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10418 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
10419 \renewcommand*{\glxtrfullformat}[2]{%
10420   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10421   \ifglxtrininsertinside\else##2\fi
10422 }%
10423 \renewcommand*{\glxtrfullplformat}[2]{%
10424   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10425   \ifglxtrininsertinside\else##2\fi
10426 }%
10427 \renewcommand*{\Glsxtrfullformat}[2]{%
10428   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10429   \ifglxtrininsertinside\else##2\fi
10430 }%
10431 \renewcommand*{\Glsxtrfullplformat}[2]{%
10432   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10433   \ifglxtrininsertinside\else##2\fi
10434 }%
```

In-line format:

```
10435 \renewcommand*{\glxtrinlinefullformat}[2]{%
10436   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10437   \ifglxtrininsertinside\else##2\fi
10438   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10439 }%
10440 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10441   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10442   \ifglxtrininsertinside\else##2\fi
10443   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10444 }%
10445 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10446   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10447   \ifglxtrininsertinside\else##2\fi
10448   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10449 }%
10450 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10451   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10452   \ifglxtrininsertinside\else##2\fi
10453   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10454 }%
10455 }
```

ortuserdesname

```
10456 \newcommand*{\glxtrlongshortuserdesname}{%
```

```

10457 \protect\glslonguserfont{\the\glslongtok}%
10458 \protect\glsxtruserparen
10459 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10460 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10461 \newabbreviationstyle{long-postshort-user-desc}%
10462 {%
10463   \renewcommand*{\CustomAbbreviationFields}{%
10464     name={\glsxtrlongshortuserdescname},
10465     sort={\the\glslongtok},
10466     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10467     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10468     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10469     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
10470   }%
10471   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10472     \csdef{glsxtrpostlink\glscategorylabel}{%
10473       \glsxtrifwasfirstuse
10474       {%
10475         \glsxtruserparen
10476         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10477         {\glslabel}%
10478       }%
10479     }%
10480   }%
10481   \glsattribute{\the\glslabeltok}{regular}%
10482   {%
10483     \glssetattribute{\the\glslabeltok}{regular}{false}%
10484   }%
10485   {}%
10486 }%
10487 }%
10488 {%
10489   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10490 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10491 \newabbreviationstyle{short-postlong-user}%
10492 {%
10493   \renewcommand*{\CustomAbbreviationFields}{%
10494     name={\glsxtrshortlongname},
10495     sort={\the\glsshorttok},
10496     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10497     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10498     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10499     description={\protect\glsuserdescription{\the\glslongtok}}%
10500     {\the\glslabeltok}}%

```

```

10501 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10502   \csdef{glxtrpostlink\glscategorylabel}{%
10503     \glxtrifwasfirstuse
10504     {%
10505       \glxtruserparen
10506       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10507       {\glslabel}%
10508     }%
10509   }%
10510 }%
10511 \glshasattribute{\the\glslabeltok}{regular}%
10512 {%
10513   \glissetattribute{\the\glslabeltok}{regular}{false}%
10514 }%
10515 {}%
10516 }%
10517 }%
10518 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10519 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10520 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10521 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10522 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10523 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10524 \renewcommand*{\glxtrfullformat}[2]{%
10525   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10526   \ifglxtrininsertinside\else##2\fi
10527 }%
10528 \renewcommand*{\glxtrfullplformat}[2]{%
10529   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10530   \ifglxtrininsertinside\else##2\fi
10531 }%
10532 \renewcommand*{\Glsxtrfullformat}[2]{%
10533   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10534   \ifglxtrininsertinside\else##2\fi
10535 }%
10536 \renewcommand*{\Glsxtrfullplformat}[2]{%
10537   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10538   \ifglxtrininsertinside\else##2\fi
10539 }%

```

In-line format:

```

10540 \renewcommand*{\glxtrinlinefullformat}[2]{%
10541   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10542   \ifglxtrininsertinside\else##2\fi
10543   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10544 }%

```

```

10545 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10546   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10547   \ifglxtrinsertinside\else##2\fi
10548   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10549 }%
10550 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10551   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10552   \ifglxtrinsertinside\else##2\fi
10553   \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10554 }%
10555 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10556   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10557   \ifglxtrinsertinside\else##2\fi
10558   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10559 }%
10560 }

```

onguserdescname

```

10561 \newcommand*{\glxtrshortlonguserdescname}{%
10562   \protect\glssabbrvuserfont{\the\glssshorttok}%
10563   \protect\glsxtruserparen
10564     {\protect\glslonguserfont{\the\glslongpltok}}}%
10565     {\the\glslabelltok}%
10566 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10567 \newabbreviationstyle{short-postlong-user-desc}%
10568 {%
10569   \renewcommand*{\CustomAbbreviationFields}{%
10570     name={\glxtrshortlonguserdescname},
10571     sort={\the\glssshorttok},
10572     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10573     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10574     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10575     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10576 }%
10577 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10578   \csdef{glxtrpostlink\glscategorylabel}{%
10579     \glsxtrifwasfirstuse
10580     {%
10581       \glsxtruserparen
10582         {\glsfirstlonguserfont{\glssentrylong{\glslabel}}}%
10583         {\glslabel}%
10584       }%
10585     }%
10586   }%
10587   \glshasattribute{\the\glslabelltok}{regular}%
10588   {%

```

```

10589     \glsetattribute{\the\glslabeltok}{regular}{false}%
10590   }%
10591   {}%
10592 }%
10593 }%
10594 {%
10595   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10596 }

```

short-user-desc

```

10597 \newabbreviationstyle{long-short-user-desc}%
10598 {%
10599   \renewcommand*{\CustomAbbreviationFields}{%
10600     name={\glxtrlongshortuserdescname},
10601     sort={\glxtrlongshortdescsort},%

10602     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10603       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10604     {\the\glslabeltok}},%
10605     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10606       \protect\glsxtruserparen
10607       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10608     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10609     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10610   }%

```

Unset the regular attribute if it has been set.

```

10611   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10612     \glshasattribute{\the\glslabeltok}{regular}%
10613     {%
10614       \glsetattribute{\the\glslabeltok}{regular}{false}%
10615     }%
10616     {}%
10617   }%
10618 }%
10619 {%
10620   \GlsXtrUseAbbrStyleFmts{long-short-user}%
10621 }

```

short-long-user

```

10622 \newabbreviationstyle{short-long-user}%
10623 {%

  \glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in
  \glsuserdescription.)

10624   \renewcommand*{\CustomAbbreviationFields}{%
10625     name={\glxtrshortlongname},
10626     sort={\the\glsshorttok},
10627     description={\protect\glsuserdescription{\the\glslongtok}%
10628       {\the\glslabeltok}},%

```

```

10629 first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10630 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10631 {\the\glslabeltok}},%
10632 firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10633 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10634 {\the\glslabeltok}},%

```

```

10635 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10636 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10637 \glshasattribute{\the\glslabeltok}{regular}%
10638 {%
10639 \glssetattribute{\the\glslabeltok}{regular}{false}%
10640 }%
10641 {}%
10642 }%
10643 }%
10644 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10645 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10646 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10647 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10648 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10649 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10650 \renewcommand*{\glsxtrfullformat}[2]{%
10651 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10652 \ifglsxtrinsertinside\else##2\fi
10653 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10654 }%
10655 \renewcommand*{\glsxtrfullplformat}[2]{%
10656 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10657 \ifglsxtrinsertinside\else##2\fi
10658 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10659 }%
10660 \renewcommand*{\Glsxtrfullformat}[2]{%
10661 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10662 \ifglsxtrinsertinside\else##2\fi
10663 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10664 }%
10665 \renewcommand*{\Glsxtrfullplformat}[2]{%
10666 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10667 \ifglsxtrinsertinside\else##2\fi
10668 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10669 }%
10670 }

```

-long-user-desc

```
10671 \newabbreviationstyle{short-long-user-desc}%
10672 {%
10673   \renewcommand*{\CustomAbbreviationFields}{%
10674     name={\glxtrshortlonguserdescname},
10675     sort={\glxtrshortlongdescsort},%

10676     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10677       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10678     {\the\glslabeltok}},%
10679     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10680       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10681     {\the\glslabeltok}},%
10682     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10683     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10684   }%

   Unset the regular attribute if it has been set.

10685   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10686     \glshasattribute{\the\glslabeltok}{regular}%
10687     {%
10688       \glissetattribute{\the\glslabeltok}{regular}{false}%
10689     }%
10690   }%
10691 }%
10692 }%
10693 {%
10694   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10695 }
```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glxtrlong` set `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

`trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```
10696 \newrobustcmd*{\glxtrifhyphenstart}[3]{%
10697   \ifx\glsinsert#1\relax
10698     \expandafter\@glxtrifhyphenstart#1\relax\relax
10699     \@end@glxtrifhyphenstart{#2}{#3}%
10700   \else
10701     \@glxtrifhyphenstart#1\relax\relax\@end@glxtrifhyphenstart{#2}{#3}%
10702   \fi
10703 }
```

trifhyphenstart

```
10704 \def\@glxtrifhyphenstart#1#2\@end@glxtrifhyphenstart#3#4{%
10705   \ifx-#1\relax#3\else #4\fi
10706 }
```

rlonghyphenshort

```
\glxtrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10707 \newcommand*\glxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10708   {%
```

If *<insert>* starts with a hyphen, redefine `\glxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxtrwordsep` if *<insert>* doesn't start with a hyphen.

```
10709   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
10710   \glsfirslonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10711   \ifglxtrininsertinside\else{#4}\fi
10712   \glxtrfullsep{#1}%
10713   \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10714     \ifglxtrininsertinside\else{#4}\fi}%
10715   }%
10716 }
```

abbrvhyphenfont

```
10717 \newcommand*\glsabbrvhyphenfont{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10718 \newcommand*\glsfirstabbrvhyphenfont{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10719 \newcommand*\glslonghyphenfont{\glslongdefaultfont}%
```

tlonghyphenfont

```
10720 \newcommand*\glsfirslonghyphenfont{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10721 \newcommand*\glxtrhyphensuffix{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the `markwords` attribute.

```
10722 \newabbreviationstyle{long-hyphen-short-hyphen}%
10723 {%
```

```

10724 \renewcommand*{\CustomAbbreviationFields}{%
10725     name={\glxtrlongshortname},
10726     sort={\the\glsshorttok},
10727     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10728         \protect\glxtrfullsep{\the\glslabeltok}%
10729         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10730     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10731         \protect\glxtrfullsep{\the\glslabeltok}%
10732         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10733     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10734     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10735 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10736     \glshasattribute{\the\glslabeltok}{regular}%
10737     {%
10738         \glissetattribute{\the\glslabeltok}{regular}{false}%
10739     }%
10740     {}%
10741 }%
10742 }%
10743 {%
10744     \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10745     \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
10746     \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10747     \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10748     \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10749 \renewcommand*{\glxtrfullformat}[2]{%
10750     \glxtrlonghyphenshort{##1}{\glssaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
10751 }%
10752 \renewcommand*{\glxtrfullplformat}[2]{%
10753     \glxtrlonghyphenshort{##1}{\glssaccesslongpl{##1}}%
10754     {\glssaccessshortpl{##1}}{##2}%
10755 }%
10756 \renewcommand*{\Glsxtrfullformat}[2]{%
10757     \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\Glsaccessshort{##1}}{##2}%
10758 }%
10759 \renewcommand*{\Glsxtrfullplformat}[2]{%
10760     \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10761     {\Glsaccessshortpl{##1}}{##2}%
10762 }%
10763 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10764 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10765 {%
10766     \renewcommand*{\CustomAbbreviationFields}{%
10767         name={\glxtrlongshortdescname},

```

```

10768 sort={\glxtrlongshortdescsort},
10769 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10770 \protect\glxtrfullsep{\the\glslabeltok}%
10771 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10772 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10773 \protect\glxtrfullsep{\the\glslabeltok}%
10774 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10775 text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
10776 plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
10777 }%

```

Unset the regular attribute if it has been set.

```

10778 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10779 \glshasattribute{\the\glslabeltok}{regular}%
10780 {%
10781 \glissetattribute{\the\glslabeltok}{regular}{false}%
10782 }%
10783 {}%
10784 }%
10785 }%
10786 {%
10787 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10788 }

```

onghyphennoshort `\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}`

```

10789 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10790 {%

```

If *<insert>* starts with a hyphen, redefine `\glxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxtrwordsep` if *<insert>* doesn't start with a hyphen.

```

10791 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10792 \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10793 \ifglxtrininsertinside\else{#3}\fi
10794 }%
10795 }

```

hort-desc-noreg This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```

10796 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10797 {%
10798 \renewcommand*{\CustomAbbreviationFields}{%

```

```

10799     name={\glxtrlongnoshortdescname},
10800     sort={\expandonce\glxtrorglong},
10801     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10802     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10803     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10804 }%

```

Unset the regular attribute if it has been set.

```

10805 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10806   \glshasattribute{\the\glslabeltok}{regular}%
10807   {%
10808     \glissetattribute{\the\glslabeltok}{regular}{false}%
10809   }%
10810   {}}%
10811 }%
10812 }%
10813 {%
10814   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10815 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10816 \renewcommand*{\glssabrvfont[1]}{\glssabrvdefaultfont{##1}}%
10817 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10818 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10819 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10820 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10821   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10822 }%
10823 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10824   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10825 }%
10826 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10827   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10828 }%
10829 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10830   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10831 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10832 \renewcommand*{\glxtrinlinefullformat}[2]{%
10833   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10834   \glxtrfullsep{##1}%
10835   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10836 }%
10837 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10838   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10839   \glxtrfullsep{##1}%
10840   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
10841 }%

```

```

10842 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10843   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10844   \glsxtrfullsep{##1}%
10845   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
10846 }%
10847 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10848   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10849   \glsxtrfullsep{##1}%
10850   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
10851 }%

```

The first use full form only displays the long form.

```

10852 \renewcommand*{\glsxtrfullformat}[2]{%
10853   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10854 }%
10855 \renewcommand*{\glsxtrfullplformat}[2]{%
10856   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10857 }%
10858 \renewcommand*{\Glsxtrfullformat}[2]{%
10859   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10860 }%
10861 \renewcommand*{\Glsxtrfullplformat}[2]{%
10862   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10863 }%
10864 }

```

n-noshort-noreg It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```

10865 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10866 {%
10867   \renewcommand*{\CustomAbbreviationFields}{%
10868     name={\glsxtrlongnoshortname},
10869     sort={\the\glsshorttok},
10870     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10871     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10872     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10873     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10874     description={\the\glslongtok}%
10875   }%

```

Unset the regular attribute if it has been set.

```

10876 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10877   \glshasattribute{\the\glslabeltok}{regular}%
10878   {%
10879     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10880   }%
10881   {}%
10882 }%
10883 }%

```

```

10884 {%
10885   \GlsXtrUseAbbrStyleFmts{long-desc}%
10886 }

```

glsxtrlonghyphen `\glsxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10887 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10888 {%
10889   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10890   \glsfirstlonghyphenfont{#1}%
10891 }%
10892 }

```

rpostthyphenshort `\glsxtrpostthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glsxtrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```

10893 \newcommand*{\glsxtrpostthyphenshort}[2]{%
10894 {%
10895   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10896   \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10897   \glsxtrfullsep{#1}%
10898   \glsxtrparen
10899   {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10900   \ifglsxtrininsertinside\else{#2}\fi
10901 }%
10902 }%
10903 }

```

hyphensubsequent `\glsxtrpostthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10904 \newcommand*{\glsxtrpostthyphensubsequent}[2]{%
10905   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%
10906   \ifglsxtrininsertinside \else{#2}\fi
10907 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10908 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10909 {%
10910   \renewcommand*{\CustomAbbreviationFields}{%
10911     name={\glxtrlongshortname},
10912     sort={\the\glsshorttok},
10913     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10914     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10915     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10916     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10917   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10918     \csdef{glsxtrpostlink\glscategorylabel}{%
10919       \glxtrifwasfirstuse
10920       {%
10921         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10922       }%
10923     }%

```

Put the insertion into the post-link:

```

10924       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10925     }%
10926   }%
10927   \glshasattribute{\the\glslabeltok}{regular}%
10928   {%
10929     \glissetattribute{\the\glslabeltok}{regular}{false}%
10930   }%
10931   {}%
10932 }%
10933 }%
10934 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10935 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10936 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10937 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10938 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10939 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10940 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10941   \glsabbrvfont{\glsaccessshort{##1}}%
10942 }%
10943 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10944   \glsabbrvfont{\glsaccessshortpl{##1}}%
10945 }%
10946 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10947   \glsabbrvfont{\Glsaccessshort{##1}}%
10948 }%
10949 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

```

```

10950 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10951 }%
    First use full form:
10952 \renewcommand*{\glsxtrfullformat}[2]{%
10953 \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10954 }%
10955 \renewcommand*{\glsxtrfullplformat}[2]{%
10956 \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10957 }%
10958 \renewcommand*{\Glsxtrfullformat}[2]{%
10959 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10960 }%
10961 \renewcommand*{\Glsxtrfullplformat}[2]{%
10962 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10963 }%

```

In-line format.

```

10964 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10965 \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10966 \ifglsxtrininsertinside{##2}\fi}%
10967 \ifglsxtrininsertinside \else{##2}\fi
10968 }%
10969 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10970 \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10971 \ifglsxtrininsertinside{##2}\fi}%
10972 \ifglsxtrininsertinside \else{##2}\fi
10973 }%
10974 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10975 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10976 \ifglsxtrininsertinside{##2}\fi}%
10977 \ifglsxtrininsertinside \else{##2}\fi
10978 }%
10979 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10980 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10981 \ifglsxtrininsertinside{##2}\fi}%
10982 \ifglsxtrininsertinside \else{##2}\fi
10983 }%
10984 }

```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10985 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10986 {%
10987 \renewcommand*{\CustomAbbreviationFields}{%
10988 name={\glsxtrlongshortdescname},
10989 sort={\glsxtrlongshortdescsort},%
10990 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10991 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10992 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10993 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%

```

```

10994 }%
10995 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10996   \csdef{glstrpostlink\glscategorylabel}{%
10997     \glstrifwasfirstuse
10998     {%
10999       \glstrposthyphenshort{\glslabel}{\glsinsert}}%
11000   }%
11001   {%

```

Put the insertion into the post-link:

```

11002     \glstrposthyphensubsequent{\glslabel}{\glsinsert}}%
11003   }%
11004 }%
11005 \glshasattribute{\the\glslabeltok}{regular}%
11006 {%
11007   \glsssetAttribute{\the\glslabeltok}{regular}{false}%
11008 }%
11009 {}%
11010 }%
11011 }%
11012 {%
11013   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
11014 }

```

rshorthyphenlong

```
\glstrshorthyphenlong{<label>}{<short>}{<long>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
11015 \newcommand*{\glstrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
11016 {%
```

If *<insert>* starts with a hyphen, redefine `\glstrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.)

```

11017   \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{%
11018   \glsfirstabbrvhyphenfont{#2\ifglstrinsertinside{#4}\fi}%
11019   \ifglstrinsertinside\else{#4}\fi
11020   \glstrfullsep{#1}%
11021   \glstrparen{\glsfirstlonghyphenfont{#3\ifglstrinsertinside{#4}\fi}%
11022   \ifglstrinsertinside\else{#4}\fi}%
11023 }%
11024 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```
11025 \newabbreviationstyle{short-hyphen-long-hyphen}%
```

```

11026 {%
11027   \renewcommand*{\CustomAbbreviationFields}{%
11028     name={\glxtrshortlongname},
11029     sort={\the\glsshorttok},
11030     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
11031       \protect\glxtrfullsep{\the\glslabeltok}%
11032       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11033     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
11034       \protect\glxtrfullsep{\the\glslabeltok}%
11035       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11036     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
11037     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

11038   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11039     \glshasattribute{\the\glslabeltok}{regular}%
11040     {%
11041       \glissetattribute{\the\glslabeltok}{regular}{false}%
11042     }%
11043   }%
11044 }%
11045 }%
11046 {%
11047   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
11048   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11049   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11050   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11051   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11052   \renewcommand*{\glxtrfullformat}[2]{%
11053     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
11054   }%
11055   \renewcommand*{\glxtrfullplformat}[2]{%
11056     \glxtrshorthyphenlong{##1}%
11057     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
11058   }%
11059   \renewcommand*{\Glsxtrfullformat}[2]{%
11060     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
11061   }%
11062   \renewcommand*{\Glsxtrfullplformat}[2]{%
11063     \glxtrshorthyphenlong{##1}%
11064     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
11065   }%
11066 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

11067 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
11068 {%
11069   \renewcommand*{\CustomAbbreviationFields}{%

```

```

11070   name={\glxtrshortlongdescname},
11071   sort={\glxtrshortlongdescsort},
11072   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
11073     \protect\glxtrfullsep{\the\glslabeltok}%
11074     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11075   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
11076     \protect\glxtrfullsep{\the\glslabeltok}%
11077     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11078   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11079   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
11080 }%

```

Unset the regular attribute if it has been set.

```

11081 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11082   \glshasattribute{\the\glslabeltok}{regular}%
11083   {%
11084     \glissetattribute{\the\glslabeltok}{regular}{false}%
11085   }%
11086   {}%
11087 }%
11088 }%
11089 {%
11090   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11091 }

```

`\glxtrshorthyphen` `\glxtrshorthyphen{<short>}{<label>}{<insert>}`

Used by short-hyphen-postlong-hyphen. The `<insert>` is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

11092 \newcommand*{\glxtrshorthyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

11093 {%
11094   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
11095     \glsfirstabbrvhyphenfont{#1}%
11096   }%
11097 }

```

`\glxtrposthyphenlong` `\glxtrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the `<short>` part. This always uses the singular long form.

```

11098 \newcommand*{\glxtrposthyphenlong}[2]{%
11099   {%

```

```

11100 \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11101 \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11102 \glxtrfullsep{#1}%
11103 \glxtrparen
11104 {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
11105 \ifglxtrininsertinside\else{#2}\fi
11106 }%
11107 }%
11108 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

11109 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11110 {%
11111 \renewcommand*{\CustomAbbreviationFields}{%
11112 name={\glxtrshortlongname},
11113 sort={\the\glsshorttok},
11114 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11115 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11116 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
11117 description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11118 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11119 \csdef{glxtrpostlink\glscategorylabel}{%
11120 \glxtrifwasfirstuse
11121 {%
11122 \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11123 }%
11124 }%

```

Put the insertion into the post-link:

```

11125 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11126 }%
11127 }%
11128 \glshasattribute{\the\glslabeltok}{regular}%
11129 {%
11130 \glissetattribute{\the\glslabeltok}{regular}{false}%
11131 }%
11132 {}%
11133 }%
11134 }%
11135 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11136 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11137 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
11138 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11139 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11140 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

11141 \renewcommand*{\glxtrsubsequentfmt}[2]{%
11142   \glsabbrvfont{\glsaccessshort{##1}}%
11143 }%
11144 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11145   \glsabbrvfont{\glsaccessshortpl{##1}}%
11146 }%
11147 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11148   \glsabbrvfont{\Glsaccessshort{##1}}%
11149 }%
11150 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11151   \glsabbrvfont{\Glsaccessshortpl{##1}}%
11152 }%

```

First use full form:

```

11153 \renewcommand*{\glxtrfullformat}[2]{%
11154   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
11155 }%
11156 \renewcommand*{\glxtrfullplformat}[2]{%
11157   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
11158 }%
11159 \renewcommand*{\Glsxtrfullformat}[2]{%
11160   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
11161 }%
11162 \renewcommand*{\Glsxtrfullplformat}[2]{%
11163   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
11164 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

11165 \renewcommand*{\glxtrinlinefullformat}[2]{%
11166   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
11167   \ifglxtrininsertinside{##2}\fi%
11168   \ifglxtrininsertinside \else{##2}\fi
11169 }%
11170 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11171   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
11172   \ifglxtrininsertinside{##2}\fi%
11173   \ifglxtrininsertinside \else{##2}\fi
11174 }%
11175 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11176   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
11177   \ifglxtrininsertinside{##2}\fi%
11178   \ifglxtrininsertinside \else{##2}\fi
11179 }%
11180 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11181   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
11182   \ifglxtrininsertinside{##2}\fi%
11183   \ifglxtrininsertinside \else{##2}\fi
11184 }%
11185 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
11186 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
11187 {%
11188   \renewcommand*{\CustomAbbreviationFields}{%
11189     name={\glstrshortlongdescname},
11190     sort={\glstrshortlongdescsort},%
11191     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11192     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11193     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11194     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11195   }%
11196   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11197     \csdef{glstrpostlink\glscategorylabel}{%
11198       \glstrifwasfirstuse
11199       {%
11200         \glstrposthyphenlong{\glslabel}{\glsinsert}%
11201       }%
11202     }%
```

Put the insertion into the post-link:

```
11203       \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
11204     }%
11205   }%
11206   \glshasattribute{\the\glslabeltok}{regular}%
11207   {%
11208     \glissetattribute{\the\glslabeltok}{regular}{false}%
11209   }%
11210   {}%
11211 }%
11212 }%
11213 {%
11214   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
11215 }
```

1.7.8 Predefined Styles (No Short on First Use)

These styles show only the long form on first use and only the short form on subsequent use.

lsabbrvonlyfont

```
11216 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
11217 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
11218 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
11219 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
11220 \newcommand*{\glxtronlysuffix}{\glxtrabbrvpluralsuffix}
```

\glxtronlyname The default name format for this style.

```
11221 \newcommand*{\glxtronlyname}{%
11222   \protect\glxtronlyfont{\the\glsshorttok}%
11223 }
```

only-short-only

```
11224 \newabbreviationstyle{long-only-short-only}%
11225 {%
11226   \renewcommand*{\CustomAbbreviationFields}{%
11227     name={\glxtronlyname},
11228     sort={\the\glsshorttok},
11229     first={\protect\glxtronlyfont{\the\glslongtok}},%
11230     firstplural={\protect\glxtronlyfont{\the\glslongpltok}},%
11231     plural={\protect\glxtronlyfont{\the\glsshortpltok}},%
11232     description={\protect\glxtronlyfont{\the\glslongtok}}}%
11233 }
```

Unset the regular attribute if it has been set.

```
11233 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11234   \glshasattribute{\the\glslabeltok}{regular}%
11235   {%
11236     \glissetattribute{\the\glslabeltok}{regular}{false}%
11237   }%
11238   {}%
11239 }%
11240 }%
11241 {%
11242   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtronlysuffix}%
11243   \renewcommand*{\glxtronlyfont}[1]{\glxtronlyfont{##1}}%
11244   \renewcommand*{\glxtronlyfont}[1]{\glxtronlyfont{##1}}%
11245   \renewcommand*{\glxtronlyfont}[1]{\glxtronlyfont{##1}}%
11246   \renewcommand*{\glxtronlyfont}[1]{\glxtronlyfont{##1}}%
11247 }
```

The first use full form doesn't show the short form.

```
11247 \renewcommand*{\glxtrfullformat}[2]{%
11248   \glxtronlyfont{\glxtronlyfont{##1}}\ifglxtrinsertinside##2\fi}%
11249   \ifglxtrinsertinside\else##2\fi
11250 }%
11251 \renewcommand*{\glxtrfullplformat}[2]{%
11252   \glxtronlyfont{\glxtronlyfont{##1}}\ifglxtrinsertinside##2\fi}%
11253   \ifglxtrinsertinside\else##2\fi
11254 }%
11255 \renewcommand*{\GlsXtrfullformat}[2]{%
11256   \glxtronlyfont{\glxtronlyfont{##1}}\ifglxtrinsertinside##2\fi}%
11257   \ifglxtrinsertinside\else##2\fi
11258 }%
```

```

11259 \renewcommand*{\Glsxtrfullplformat}[2]{%
11260   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11261   \ifglxtrinsertinside\else##2\fi
11262 }%

```

The inline full form does show the short form.

```

11263 \renewcommand*{\glxtrinlinefullformat}[2]{%
11264   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11265   \ifglxtrinsertinside\else##2\fi
11266   \glxtrfullsep{##1}%
11267   \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
11268 }%
11269 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11270   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11271   \ifglxtrinsertinside\else##2\fi
11272   \glxtrfullsep{##1}%
11273   \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
11274 }%
11275 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11276   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11277   \ifglxtrinsertinside\else##2\fi
11278   \glxtrfullsep{##1}%
11279   \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
11280 }%
11281 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11282   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11283   \ifglxtrinsertinside\else##2\fi
11284   \glxtrfullsep{##1}%
11285   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11286 }%
11287 }

```

xtronlydescsort

```

11288 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

11289 \newcommand*{\glxtronlydescname}{%
11290   \protect\glslongfont{\the\glslongtok}}%
11291 }

```

short-only-desc

```

11292 \newabbreviationstyle{long-only-short-only-desc}%
11293 {%
11294   \renewcommand*{\CustomAbbreviationFields}{%
11295     name={\glxtronlydescname},
11296     sort={\glxtronlydescsort},%
11297     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11298     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11299     text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%

```

```

11300 plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
11301 }%

Unset the regular attribute if it has been set.

11302 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11303   \glshasattribute{\the\glslabeltok}{regular}%
11304   {%
11305     \glssetattribute{\the\glslabeltok}{regular}{false}%
11306   }%
11307   {}}%
11308 }%
11309}%
11310{%
11311 \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
11312}

```

1.8 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 `\glsxtrshort` with the `noindex` key set, which prevents the unwanted additions to the location list, and the `hyper` key set to `false`, which prevents the problem of nested links. This just leaves one thing left that needs to be dealt with, and that's what to do if the heading style uses `\MakeUppercase`.

Note that glossaries automatically loads `textcase`, so the label can be protected from case change with `textcase`'s `\NoCaseChange`. This means that we don't have a problem provided the page style uses `\MakeTextUppercase`, but the default heading page style uses `\MakeUppercase`.

To get around this, save the original definition of `\markboth` and `\markright` and adjust it so that `\MakeUppercase` is temporarily redefined to `\MakeTextUppercase`. Some packages or classes redefine these commands, so we can't just assume they still have the original kernel definition.

`\markright` Save original definition:

```
11313 \let\@glsxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```

11314 \renewcommand*{\markright}[1]{%
11315   \glsxtrmarkhook

```

```

11316 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
11317 \glsxtrrestoremarkhook
11318 }

```

\markboth Save original definition:

```

11319 \let\@glsxtr@org@markboth\markboth

```

Redefine (grouping not added in case it interferes with the original code):

```

11320 \renewcommand*{\markboth}[2]{%
11321 \glsxtrmarkhook
11322 \@glsxtr@org@markboth
11323 {\@glsxtrinmark#1\@glsxtrnotinmark}%
11324 {\@glsxtrinmark#2\@glsxtrnotinmark}%
11325 \glsxtrrestoremarkhook
11326 }

```

Also do this for \@starttoc

\@starttoc Save original definition:

```

11327 \let\@glsxtr@org@@starttoc\@starttoc

```

Redefine:

```

11328 \renewcommand*{\@starttoc}[1]{%
11329 \glsxtrmarkhook
11330 \@glsxtrinmark
11331 \@glsxtr@org@@starttoc{#1}%
11332 \@glsxtrnotinmark
11333 \glsxtrrestoremarkhook
11334 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

11335 \newcommand*{\glsxtrRevertMarks}{%
11336 \let\markright\@glsxtr@org@markright
11337 \let\markboth\@glsxtr@org@markboth
11338 \let\@starttoc\@glsxtr@org@@starttoc
11339 }

```

rRevertTocMarks Just restores \@starttoc.

```

11340 \newcommand*{\glsxtrRevertTocMarks}{%
11341 \let\@starttoc\@glsxtr@org@@starttoc
11342 }

```

\glsxtrifinmark

```

11343 \newcommand*{\glsxtrifinmark}[2]{#2}

```

\@glsxtrinmark

```

11344 \newrobustcmd*{\@glsxtrinmark}{%
11345 \let\glsxtrifinmark\@firstoftwo
11346 }

```

glsxtrnotinmark

```
11347 \newrobustcmd*{\@glsxtrnotinmark}{%
11348   \let\glsxtrifinmark\@secondoftwo
11349 }
```

eorpdforheading

```
11350 \ifdef\teorpdfstring
11351 {
11352   \newcommand*{\glsxtrtitleorpdforheading}[3]{\teorpdfstring{#1}{#2}}
11353 }
11354 {
11355   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
11356 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11357 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
11358 \let\@glsxtr@org@MakeUppercase\MakeUppercase
11359 \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11360 \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11361 \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11362 \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11363 \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11364 \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11365 \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11366 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11367 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11368 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11369 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11370 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11371 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11372 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11373 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11374 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11375 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11376 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11377 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11378 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11379 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11380 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11381 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
11382 \let\glsxtrifinmark\@firstoftwo
11383 \let\MakeUppercase\MakeTextUppercase
11384 \let\glsxtrtitleorpdforheading\@thirdofthree
11385 \let\glsxtrtitleshort\glsxtrheadshort
11386 \let\glsxtrtitleshortpl\glsxtrheadshortpl
```

```

11387 \let\Glsxtrtitleshort\Glsxtrheadshort
11388 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11389 \let\Glsxtrtitlename\Glsxtrheadname
11390 \let\Glsxtrtitlename\Glsxtrheadname
11391 \let\Glsxtrtitletext\Glsxtrheadtext
11392 \let\Glsxtrtitletext\Glsxtrheadtext
11393 \let\Glsxtrtitleplural\Glsxtrheadplural
11394 \let\Glsxtrtitleplural\Glsxtrheadplural
11395 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11396 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11397 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11398 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11399 \let\Glsxtrtitlelong\Glsxtrheadlong
11400 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11401 \let\Glsxtrtitlelong\Glsxtrheadlong
11402 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11403 \let\Glsxtrtitlefull\Glsxtrheadfull
11404 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11405 \let\Glsxtrtitlefull\Glsxtrheadfull
11406 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11407 }

```

`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.)

```

11408 \newcommand*{\Glsxtrrestoremarkhook}{%
11409   \let\Glsxtrifinmark\@secondoftwo
11410   \let\MakeUppercase\@Glsxtr@org@MakeUppercase
11411   \let\Glsxtrtitleorpdforheading\@Glsxtr@org@Glsxtrtitleorpdforheading
11412   \let\Glsxtrtitleshort\@Glsxtr@org@Glsxtrtitleshort
11413   \let\Glsxtrtitleshortpl\@Glsxtr@org@Glsxtrtitleshortpl
11414   \let\Glsxtrtitleshort\@Glsxtr@org@Glsxtrtitleshort
11415   \let\Glsxtrtitleshortpl\@Glsxtr@org@Glsxtrtitleshortpl
11416   \let\Glsxtrtitlename\@Glsxtr@org@Glsxtrtitlename
11417   \let\Glsxtrtitlename\@Glsxtr@org@Glsxtrtitlename
11418   \let\Glsxtrtitletext\@Glsxtr@org@Glsxtrtitletext
11419   \let\Glsxtrtitletext\@Glsxtr@org@Glsxtrtitletext
11420   \let\Glsxtrtitleplural\@Glsxtr@org@Glsxtrtitleplural
11421   \let\Glsxtrtitleplural\@Glsxtr@org@Glsxtrtitleplural
11422   \let\Glsxtrtitlefirst\@Glsxtr@org@Glsxtrtitlefirst
11423   \let\Glsxtrtitlefirst\@Glsxtr@org@Glsxtrtitlefirst
11424   \let\Glsxtrtitlefirstplural\@Glsxtr@org@Glsxtrtitlefirstplural
11425   \let\Glsxtrtitlefirstplural\@Glsxtr@org@Glsxtrtitlefirstplural
11426   \let\Glsxtrtitlelong\@Glsxtr@org@Glsxtrtitlelong
11427   \let\Glsxtrtitlelongpl\@Glsxtr@org@Glsxtrtitlelongpl
11428   \let\Glsxtrtitlelong\@Glsxtr@org@Glsxtrtitlelong
11429   \let\Glsxtrtitlelongpl\@Glsxtr@org@Glsxtrtitlelongpl
11430   \let\Glsxtrtitlefull\@Glsxtr@org@Glsxtrtitlefull

```

```

11431 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
11432 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
11433 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
11434 }

```

Instead of using one document-wide conditional, use `headuc` attribute to determine whether or not to use the all upper case form.

glsxtrheadshort Command used to display short form in the page header.

```

11435 \newcommand*{\glsxtrheadshort}[1]{%
11436   \protect\NoCaseChange
11437   {%
11438     \glsifattribute{#1}{headuc}{true}%
11439     {%
11440       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11441     }%
11442     {%
11443       \glsxtrshort[noindex,hyper=false]{#1}[]%
11444     }%
11445   }%
11446 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11447 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11448   \glsxtrshort[noindex,hyper=false]{#1}[]%
11449 }

```

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.

```

11450 \newcommand*{\glsxtrheadshortpl}[1]{%
11451   \protect\NoCaseChange
11452   {%
11453     \glsifattribute{#1}{headuc}{true}%
11454     {%
11455       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11456     }%
11457     {%
11458       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11459     }%
11460   }%
11461 }

```

lxtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

11462 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
11463   \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11464 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

11465 \newcommand*{\Glsxtrheadshort}[1]{%
11466   \protect\NoCaseChange
11467   {%
11468     \glsifattribute{#1}{headuc}{true}%
11469     {%
11470       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11471     }%
11472   }%
11473   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11474 }%
11475 }%
11476 }

```

lslxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11477 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11478   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11479 }

```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```

11480 \newcommand*{\Glsxtrheadshortpl}[1]{%
11481   \protect\NoCaseChange
11482   {%
11483     \glsifattribute{#1}{headuc}{true}%
11484     {%
11485       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11486     }%
11487   }%
11488   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11489 }%
11490 }%
11491 }

```

lxtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11492 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11493   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11494 }

```

\glsxtrheadname As above but for the name value.

```

11495 \newcommand*{\glsxtrheadname}[1]{%
11496   \protect\NoCaseChange
11497   {%
11498     \glsifattribute{#1}{headuc}{true}%
11499     {%

```

```

11500      \GLSname[noindex,hyper=false]{#1}[]%
11501    }%
11502    {%
11503      \glsname[noindex,hyper=false]{#1}[]%
11504    }%
11505  }%
11506 }

```

`\glsxtrtitlename` Command to display name value in section title and table of contents.

```

11507 \newrobustcmd*{\glsxtrtitlename}[1]{%
11508   \glsname[noindex,hyper=false]{#1}[]%
11509 }

```

`\Glsxtrheadname` First letter converted to upper case

```

11510 \newcommand*{\Glsxtrheadname}[1]{%
11511   \protect\NoCaseChange
11512   {%
11513     \glsifattribute{#1}{headuc}{true}%
11514     {%
11515       \GLSname[noindex,hyper=false]{#1}[]%
11516     }%
11517     {%
11518       \Glsname[noindex,hyper=false]{#1}[]%
11519     }%
11520   }%
11521 }

```

`\Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11522 %\changes{1.21}{2017-11-03}{new}
11523 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11524   \Glsname[noindex,hyper=false]{#1}[]%
11525 }

```

`\glsxtrheadtext` As above but for the text value.

```

11526 \newcommand*{\glsxtrheadtext}[1]{%
11527   \protect\NoCaseChange
11528   {%
11529     \glsifattribute{#1}{headuc}{true}%
11530     {%
11531       \GLStext[noindex,hyper=false]{#1}[]%
11532     }%
11533     {%
11534       \glstext[noindex,hyper=false]{#1}[]%
11535     }%
11536   }%
11537 }

```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
11538 \newrobustcmd*{\glsxtrtitletext}[1]{%
11539   \glstext[noindex,hyper=false]{#1}[]%
11540 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
11541 \newcommand*{\Glsxtrheadtext}[1]{%
11542   \protect\NoCaseChange
11543   {%
11544     \glsifattribute{#1}{headuc}{true}%
11545     {%
11546       \GLStext[noindex,hyper=false]{#1}[]%
11547     }%
11548     {%
11549       \Glstext[noindex,hyper=false]{#1}[]%
11550     }%
11551   }%
11552 }
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
11553 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11554   \Glstext[noindex,hyper=false]{#1}[]%
11555 }
```

`lsxtrheadplural` As above but for the plural value.

```
11556 \newcommand*{\glsxtrheadplural}[1]{%
11557   \protect\NoCaseChange
11558   {%
11559     \glsifattribute{#1}{headuc}{true}%
11560     {%
11561       \GLSplural[noindex,hyper=false]{#1}[]%
11562     }%
11563     {%
11564       \glsplural[noindex,hyper=false]{#1}[]%
11565     }%
11566   }%
11567 }
```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```
11568 \newrobustcmd*{\glsxtrtitleplural}[1]{%
11569   \glsplural[noindex,hyper=false]{#1}[]%
11570 }
```

`lsxtrheadplural` Convert first letter to upper case.

```
11571 \newcommand*{\Glsxtrheadplural}[1]{%
11572   \protect\NoCaseChange
11573   {%
```

```

11574 \glsifattribute{#1}{headuc}{true}%
11575 {%
11576 \GLSplural[noindex,hyper=false]{#1}[]%
11577 }%
11578 {%
11579 \Glsplural[noindex,hyper=false]{#1}[]%
11580 }%
11581 }%
11582 }

```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

11583 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11584 \GLsplural[noindex,hyper=false]{#1}[]%
11585 }

```

glsxtrheadfirst As above but for the first value.

```

11586 \newcommand*{\glsxtrheadfirst}[1]{%
11587 \protect\NoCaseChange
11588 {%
11589 \glsifattribute{#1}{headuc}{true}%
11590 {%
11591 \GLSfirst[noindex,hyper=false]{#1}[]%
11592 }%
11593 {%
11594 \glsfirst[noindex,hyper=false]{#1}[]%
11595 }%
11596 }%
11597 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```

11598 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
11599 \glsfirst[noindex,hyper=false]{#1}[]%
11600 }

```

Glsxtrheadfirst First letter converted to upper case

```

11601 \newcommand*{\Glsxtrheadfirst}[1]{%
11602 \protect\NoCaseChange
11603 {%
11604 \glsifattribute{#1}{headuc}{true}%
11605 {%
11606 \GLSfirst[noindex,hyper=false]{#1}[]%
11607 }%
11608 {%
11609 \Glsfirst[noindex,hyper=false]{#1}[]%
11610 }%
11611 }%
11612 }

```

`\lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11613 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11614   \Glsfirst[noindex,hyper=false]{#1}[]%
11615 }
```

`\headfirstplural` As above but for the firstplural value.

```
11616 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11617   \protect\NoCaseChange
11618   {%
11619     \glsifattribute{#1}{headuc}{true}%
11620     {%
11621       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11622     }%
11623     {%
11624       \glsfirstplural[noindex,hyper=false]{#1}[]%
11625     }%
11626   }%
11627 }
```

`\itlefirstplural` Command to display firstplural value in section title and table of contents.

```
11628 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
11629   \glsfirstplural[noindex,hyper=false]{#1}[]%
11630 }
```

`\headfirstplural` First letter converted to upper case

```
11631 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11632   \protect\NoCaseChange
11633   {%
11634     \glsifattribute{#1}{headuc}{true}%
11635     {%
11636       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11637     }%
11638     {%
11639       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11640     }%
11641   }%
11642 }
```

`\itlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11643 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
11644   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11645 }
```

`\glxtrheadlong` Command used to display long form in the page header.

```
11646 \newcommand*{\glxtrheadlong}[1]{%
11647   \protect\NoCaseChange
```

```

11648 {%
11649   \glsifattribute{#1}{headuc}{true}%
11650   {%
11651     \GLSxtrlong[noindex,hyper=false]{#1}[]%
11652   }%
11653   {%
11654     \glsxtrlong[noindex,hyper=false]{#1}[]%
11655   }%
11656 }%
11657 }

```

`\glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

11658 \newrobustcmd*{\glsxtrtitlelong}[1]{%
11659   \glsxtrlong[noindex,hyper=false]{#1}[]%
11660 }

```

`\glsxtrheadlongpl` 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.

```

11661 \newcommand*{\glsxtrheadlongpl}[1]{%
11662   \protect\NoCaseChange
11663   {%
11664     \glsifattribute{#1}{headuc}{true}%
11665     {%
11666       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11667     }%
11668     {%
11669       \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11670     }%
11671   }%
11672 }

```

`\sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

11673 \newrobustcmd*{\sxtrtitlelongpl}[1]{%
11674   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11675 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

11676 \newcommand*{\Glsxtrheadlong}[1]{%
11677   \protect\NoCaseChange
11678   {%
11679     \glsifattribute{#1}{headuc}{true}%
11680     {%
11681       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11682     }%
11683     {%
11684       \Glsxtrlong[noindex,hyper=false]{#1}[]%

```

```

11685 }%
11686 }%
11687 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11688 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11689   \Glsxtrlong[noindex,hyper=false]{#1}[]%
11690 }

```

lsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11691 \newcommand*{\Glsxtrheadlongpl}[1]{%
11692   \protect\NoCaseChange
11693   {%
11694     \glsifattribute{#1}{headuc}{true}%
11695     {%
11696       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11697     }%
11698     {%
11699       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11700     }%
11701   }%
11702 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11703 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11704   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11705 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11706 \newcommand*{\glsxtrheadfull}[1]{%
11707   \protect\NoCaseChange
11708   {%
11709     \glsifattribute{#1}{headuc}{true}%
11710     {%
11711       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11712     }%
11713     {%
11714       \glsxtrfull[noindex,hyper=false]{#1}[]%
11715     }%
11716   }%
11717 }

```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```

11718 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11719   \glsxtrfull[noindex,hyper=false]{#1}[]%
11720 }

```

`\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.

```

11721 \newcommand*{\glxtrheadfullpl}[1]{%
11722   \protect\NoCaseChange
11723   {%
11724     \gl@ifattribute{#1}{headuc}{true}%
11725     {%
11726       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11727     }%
11728     {%
11729       \glxtrfullpl[noindex,hyper=false]{#1}[]%
11730     }%
11731   }%
11732 }

```

`\sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```

11733 \newrobustcmd*{\glxtrtitlefullpl}[1]{%
11734   \glxtrfullpl[noindex,hyper=false]{#1}[]%
11735 }

```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```

11736 \newcommand*{\Glsxtrheadfull}[1]{%
11737   \protect\NoCaseChange
11738   {%
11739     \gl@ifattribute{#1}{headuc}{true}%
11740     {%
11741       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11742     }%
11743     {%
11744       \Glsxtrfull[noindex,hyper=false]{#1}[]%
11745     }%
11746   }%
11747 }

```

`\Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11748 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11749   \Glsxtrfull[noindex,hyper=false]{#1}[]%
11750 }

```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11751 \newcommand*{\Glsxtrheadfullpl}[1]{%
11752   \protect\NoCaseChange
11753   {%
11754     \gl@ifattribute{#1}{headuc}{true}%

```

```

11755   {%
11756     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11757   }%
11758   {%
11759     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11760   }%
11761 }%
11762 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11763 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11764   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11765 }

```

`\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.

```

11766 \ifdef\texorpdfstring
11767 {
11768   \newcommand*{\glsfmtshort}[1]{%
11769     \texorpdfstring
11770       {\glsxtrtitleshort{#1}}%
11771     {\glsentryshort{#1}}%
11772   }
11773 }
11774 {
11775   \newcommand*{\glsfmtshort}[1]{%
11776     \glsxtrtitleshort{#1}}
11777 }

```

Similarly for the plural version.

```

\glsfmtshortpl
11778 \ifdef\texorpdfstring
11779 {
11780   \newcommand*{\glsfmtshortpl}[1]{%
11781     \texorpdfstring
11782       {\glsxtrtitleshortpl{#1}}%
11783     {\glsentryshortpl{#1}}%
11784   }
11785 }
11786 {
11787   \newcommand*{\glsfmtshortpl}[1]{%
11788     \glsxtrtitleshortpl{#1}}
11789 }

```

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).

```
11790 \ifdef\teorpdfstring
11791 {
11792   \newcommand*\Glsfmtshort}[1]{%
11793     \teorpdfstring
11794       {\Glsxtrtitleshort{#1}}%
11795       {\glsentryshort{#1}}%
11796   }
11797 }
11798 {
11799   \newcommand*\Glsfmtshort}[1]{%
11800     \Glsxtrtitleshort{#1}}
11801 }
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
11802 \ifdef\teorpdfstring
11803 {
11804   \newcommand*\Glsfmtshortpl}[1]{%
11805     \teorpdfstring
11806       {\Glsxtrtitleshortpl{#1}}%
11807       {\glsentryshortpl{#1}}%
11808   }
11809 }
11810 {
11811   \newcommand*\Glsfmtshortpl}[1]{%
11812     \Glsxtrtitleshortpl{#1}}
11813 }
```

`\glsfmtname` As above but for the name value.

```
11814 \ifdef\teorpdfstring
11815 {
11816   \newcommand*\glsfmtname}[1]{%
11817     \teorpdfstring
11818       {\glsxtrtitlename{#1}}%
11819       {\glsentryname{#1}}%
11820   }
11821 }
11822 {
11823   \newcommand*\glsfmtname}[1]{%
11824     \glsxtrtitlename{#1}}
11825 }
```

`\Glsfmtname` First letter converted to upper case.

```
11826 \ifdef\teorpdfstring
11827 {
11828   \newcommand*\Glsfmtname}[1]{%
11829     \teorpdfstring
11830       {\Glsxtrtitlename{#1}}%
11831       {\glsentryname{#1}}%
11832   }
```

```

11832 }
11833 }
11834 {
11835   \newcommand*{\Glsfmtname}[1]{%
11836     \Glsxtrtitlename{#1}}
11837 }

```

`\glsfmttext` As above but for the text value.

```

11838 \ifdef\texorpdfstring
11839 {
11840   \newcommand*{\glsfmttext}[1]{%
11841     \texorpdfstring
11842       {\Glsxtrtitletext{#1}}%
11843       {\Glsentrytext{#1}}%
11844   }
11845 }
11846 {
11847   \newcommand*{\glsfmttext}[1]{%
11848     \Glsxtrtitletext{#1}}
11849 }

```

`\Glsfmttext` First letter converted to upper case.

```

11850 \ifdef\texorpdfstring
11851 {
11852   \newcommand*{\Glsfmttext}[1]{%
11853     \texorpdfstring
11854       {\Glsxtrtitletext{#1}}%
11855       {\Glsentrytext{#1}}%
11856   }
11857 }
11858 {
11859   \newcommand*{\Glsfmttext}[1]{%
11860     \Glsxtrtitletext{#1}}
11861 }

```

`\glsfmtplural` As above but for the plural value.

```

11862 \ifdef\texorpdfstring
11863 {
11864   \newcommand*{\glsfmtplural}[1]{%
11865     \texorpdfstring
11866       {\Glsxtrtitleplural{#1}}%
11867       {\Glsentryplural{#1}}%
11868   }
11869 }
11870 {
11871   \newcommand*{\glsfmtplural}[1]{%
11872     \Glsxtrtitleplural{#1}}
11873 }

```

`\Glsfmtplural` First letter converted to upper case.

```
11874 \ifdef\txorpdfstring
11875 {
11876   \newcommand*\Glsfmtplural}[1]{%
11877     \txorpdfstring
11878     {\Glsxtrtitleplural{#1}}%
11879     {\glsentryplural{#1}}%
11880   }
11881 }
11882 {
11883   \newcommand*\Glsfmtplural}[1]{%
11884     \Glsxtrtitleplural{#1}}
11885 }
```

`\glsfmtfirst` As above but for the first value.

```
11886 \ifdef\txorpdfstring
11887 {
11888   \newcommand*\glsfmtfirst}[1]{%
11889     \txorpdfstring
11890     {\glsxtrtitlefirst{#1}}%
11891     {\glsentryfirst{#1}}%
11892   }
11893 }
11894 {
11895   \newcommand*\glsfmtfirst}[1]{%
11896     \glsxtrtitlefirst{#1}}
11897 }
```

`\Glsfmtfirst` First letter converted to upper case.

```
11898 \ifdef\txorpdfstring
11899 {
11900   \newcommand*\Glsfmtfirst}[1]{%
11901     \txorpdfstring
11902     {\Glsxtrtitlefirst{#1}}%
11903     {\glsentryfirst{#1}}%
11904   }
11905 }
11906 {
11907   \newcommand*\Glsfmtfirst}[1]{%
11908     \Glsxtrtitlefirst{#1}}
11909 }
```

`\glsfmtfirstpl` As above but for the firstplural value.

```
11910 \ifdef\txorpdfstring
11911 {
11912   \newcommand*\glsfmtfirstpl}[1]{%
11913     \txorpdfstring
11914     {\glsxtrtitlefirstplural{#1}}%
11915     {\glsentryfirstplural{#1}}%
11916   }
11917 }
```

```

11916 }
11917 }
11918 {
11919   \newcommand*{\glsfmtfirstpl}[1]{%
11920     \glsxtrtitlefirstplural{#1}}
11921 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11922 \ifdef\texorpdfstring
11923 {
11924   \newcommand*{\Glsfmtfirstpl}[1]{%
11925     \texorpdfstring
11926       {\Glsxtrtitlefirstplural{#1}}%
11927       {\glsentryfirstplural{#1}}%
11928   }
11929 }
11930 {
11931   \newcommand*{\Glsfmtfirstpl}[1]{%
11932     \Glsxtrtitlefirstplural{#1}}
11933 }

```

`\glsfmtlong` As above but for the long value.

```

11934 \ifdef\texorpdfstring
11935 {
11936   \newcommand*{\glsfmtlong}[1]{%
11937     \texorpdfstring
11938       {\glsxtrtitlelong{#1}}%
11939       {\glsentrylong{#1}}%
11940   }
11941 }
11942 {
11943   \newcommand*{\glsfmtlong}[1]{%
11944     \glsxtrtitlelong{#1}}
11945 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11946 \ifdef\texorpdfstring
11947 {
11948   \newcommand*{\Glsfmtlong}[1]{%
11949     \texorpdfstring
11950       {\Glsxtrtitlelong{#1}}%
11951       {\glsentrylong{#1}}%
11952   }
11953 }
11954 {
11955   \newcommand*{\Glsfmtlong}[1]{%
11956     \Glsxtrtitlelong{#1}}
11957 }

```

`\glsfmtlongpl` As above but for the longplural value.

```
11958 \ifdef\txorpdfstring
11959 {
11960   \newcommand*{\glsfmtlongpl}[1]{%
11961     \txorpdfstring
11962     {\glsxtrtitlelongpl{#1}}%
11963     {\glsentrylongpl{#1}}%
11964   }
11965 }
11966 {
11967   \newcommand*{\glsfmtlongpl}[1]{%
11968     \glsxtrtitlelongpl{#1}}
11969 }
```

`\Glsfmtlongpl` First letter converted to upper case.

```
11970 \ifdef\txorpdfstring
11971 {
11972   \newcommand*{\Glsfmtlongpl}[1]{%
11973     \txorpdfstring
11974     {\Glsxtrtitlelongpl{#1}}%
11975     {\glsentrylongpl{#1}}%
11976   }
11977 }
11978 {
11979   \newcommand*{\Glsfmtlongpl}[1]{%
11980     \Glsxtrtitlelongpl{#1}}
11981 }
```

`\glsfmtfull` In-line full format.

```
11982 \ifdef\txorpdfstring
11983 {
11984   \newcommand*{\glsfmtfull}[1]{%
11985     \txorpdfstring
11986     {\glsxtrtitlefull{#1}}%
11987     {\glsxtrinlinefullformat{#1}{}}%
11988   }
11989 }
11990 {
11991   \newcommand*{\glsfmtfull}[1]{%
11992     \glsxtrtitlefull{#1}}
11993 }
```

`\Glsfmtfull` First letter converted to upper case.

```
11994 \ifdef\txorpdfstring
11995 {
11996   \newcommand*{\Glsfmtfull}[1]{%
11997     \txorpdfstring
11998     {\Glsxtrtitlefull{#1}}%
11999     {\Glsxtrinlinefullformat{#1}{}}%

```

```

12000 }
12001 }
12002 {
12003   \newcommand*{\Glsfmtfull}[1]{%
12004     \Glsxtrtitlefull{#1}}
12005 }

```

`\glsfmtfullpl` In-line full plural format.

```

12006 \ifdef\texorpdfstring
12007 {
12008   \newcommand*{\glsfmtfullpl}[1]{%
12009     \texorpdfstring
12010       {\Glsxtrtitlefullpl{#1}}%
12011       {\Glsxtrinlinefullplformat{#1}{}}%
12012   }
12013 }
12014 {
12015   \newcommand*{\glsfmtfullpl}[1]{%
12016     \Glsxtrtitlefullpl{#1}}
12017 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

12018 \ifdef\texorpdfstring
12019 {
12020   \newcommand*{\Glsfmtfullpl}[1]{%
12021     \texorpdfstring
12022       {\Glsxtrtitlefullpl{#1}}%
12023       {\Glsxtrinlinefullplformat{#1}{}}%
12024   }
12025 }
12026 {
12027   \newcommand*{\Glsfmtfullpl}[1]{%
12028     \Glsxtrtitlefullpl{#1}}
12029 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

12030 \newcommand*{\RequireGlossariesExtraLang}[1]{%
12031   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
12032 }

```

`sariesExtraLang`

```

12033 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
12034   \ProvidesFile{glossariesxtr-#1.ldf}%

```

12035 }

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. However, with `bib2gls` it might be useful to provide custom rules for a particular locale.)

`\glstr@loaddialect` The dialect label should be stored in `\this@dialect` before using this command.

```
12036 \newcommand{\glstr@loaddialect}{%
12037   \IfTrackedLanguageFileExists{\this@dialect}%
12038   {glossariesxtr-}% prefix
12039   {.ldf}%
12040   {%
12041     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
12042   }%
12043   {}% not found
```

If `glossaries-extra-bib2gls` has been loaded, `\@glstrdialecthook` will check for the associated script, otherwise it will do nothing.

```
12044 \@glstrdialecthook
12045 }
```

```
12046 \@ifpackageloaded{tracklang}
12047 {%
12048   \AnyTrackedLanguages
12049   {%
12050     \ForEachTrackedDialect{\this@dialect}{\glstr@loaddialect}%
12051   }%
12052   {}%
12053 }
12054 }
```

Load `glossaries-extra-stylemods` if required.

```
12055 \@glstr@redefstyles
```

and set the style:

```
12056 \@glstr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```
12057 \NeedsTeXFormat{LaTeX2e}
12058 \ProvidesPackage{glossaries-extra-bib2gls}[2019/03/22 1.39 (NLCT)]
```

These are some convenient macros for use with custom rules.

`\glshex`

```
12059 \newcommand*{\glshex}{\string\u}
```

`\glscapturedgroup`

```
12060 \newcommand*{\glscapturedgroup}{\string\$}
```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```
12061 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
12062   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
12063 }
```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```
12064 \newcommand*{\GlsXtrProvideCommand}{\providecommand}
```

`\GlsXtrRenewCommand` Like `\renewcommand` but only generates a warning rather than an error if the command isn't defined.

```
12065 \newcommand*{\GlsXtrRenewCommand}{\@star@or@long\GlsXtr@renewcommand}
```

`\GlsXtr@renewcommand`

```
12066 \newcommand*{\GlsXtr@renewcommand}[1]{%
12067   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
12068   \expandafter\@ifundefined\@gtempa
12069     {%
12070       \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
12071     }%
12072   \relax
12073   \relax
12074   \let\@ifdefinable\@rc@ifdefinable
12075   \newcommand#1%
12076 }
```

`\GlsXtrWrglossaryLocation` For use with indexcounter and bib2gls.

```
12077 \newcommand*{\GlsXtrWrglossaryLocation}[2]{#1}
```

`\GlsXtrIndexCounterLink`

`\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```
12078 \ifdef\hyperref
12079 {%
12080   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
12081     \GlsXtrIfHasField{indexcounter}{#2}%
12082     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
12083     {#1}%
12084   }
12085 }
12086 {
12087   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
12088 }
```

`\GlsXtrDualField` `\GlsXtrDualField`

The internal field used to store the dual label. The dual-field defaults to dual if no value is supplied so that's used as the default.

```
12089 \newcommand*{\GlsXtrDualField}{dual}
```

`\GlsXtrDualBackLink` `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```
12090 \newcommand*{\GlsXtrDualBackLink}[2]{%
12091   \glstrifhasfield{\GlsXtrDualField}{#2}%
12092   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
12093   {#2}%
12094 }
```

`\GlsXtrBibTeXEntryAliases` Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
12095 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
12096   article=bibtexentry,
12097   book=bibtexentry,
12098   booklet=bibtexentry,
12099   conference=bibtexentry,
12100   inbook=bibtexentry,
12101   incollection=bibtexentry,
12102   inproceedings=bibtexentry,
12103   manual=bibtexentry,
12104   mastersthesis=bibtexentry,
12105   misc=bibtexentry,
12106   phdthesis=bibtexentry,
12107   proceedings=bibtexentry,
12108   techreport=bibtexentry,
12109   unpublished=bibtexentry
12110 }
```

`\GlsXtrProvideBibTeXFields` Convenient shortcut to define the standard BibTeX fields.

```
12111 \newcommand*{\GlsXtrProvideBibTeXFields}{%
12112   \glsaddstoragekey{address}{}{\glstrbibaddress}%
12113   \glsaddstoragekey{author}{}{\glstrbibauthor}%
12114   \glsaddstoragekey{booktitle}{}{\glstrbibbooktitle}%
12115   \glsaddstoragekey{chapter}{}{\glstrbibchapter}%
12116   \glsaddstoragekey{edition}{}{\glstrbibedition}%
12117   \glsaddstoragekey{howpublished}{}{\glstrbibhowpublished}%
12118   \glsaddstoragekey{institution}{}{\glstrbibinstitution}%
12119   \glsaddstoragekey{journal}{}{\glstrbibjournal}%
}
```

```

12120 \glsaddstoragekey{month}{\glsxtrbibmonth}%
12121 \glsaddstoragekey{note}{\glsxtrbibnote}%
12122 \glsaddstoragekey{number}{\glsxtrbibnumber}%
12123 \glsaddstoragekey{organization}{\glsxtrbiborganization}%
12124 \glsaddstoragekey{pages}{\glsxtrbibpages}%
12125 \glsaddstoragekey{publisher}{\glsxtrbibpublisher}%
12126 \glsaddstoragekey{school}{\glsxtrbibschooll}%
12127 \glsaddstoragekey{series}{\glsxtrbibseries}%
12128 \glsaddstoragekey{title}{\glsxtrbibtitle}%
12129 \glsaddstoragekey{bibtextype}{\glsxtrbibtype}%
12130 \glsaddstoragekey{volume}{\glsxtrbibvolume}%
12131 }

```

Multiple supplementary references are only supported with bib2gls.

`\glstisuppllocation` This is like `\glstisupphypernumber` but the second argument is the external file name (which isn't obtained from the `externallocation` attribute). The third argument is the formatting (encap) control sequence *name*. This is ignored by default, but is set by `bib2gls` to the original encap in case it's required.

```

12132 \newcommand*\glstisuppllocation[3]{%
12133   {%
12134     \def\glstisuppllocationurl{#2}%
12135     \glshypernumber{#1}%
12136   }%
12137 }

```

`\glstrdisplaysupplloc` `\glstrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```

12138 \newcommand*\glstrdisplaysupplloc[5]{%
12139   \setentrycounter{#1}{#2}%
12140   \glstisuppllocation{#5}{#4}{#3}%
12141 }

```

`\glstrdisplaylocnameref` `\glstrdisplaylocnameref{<prefix>}{<counter>}{<format>}{<location>}{<name>}{<href>}{<hcounter>}{<external file>}` Used with the `[nameref]` record package option. The `<href>` argument was obtained from `\@currentHref` and the `<hcounter>` argument was obtained from `\theHentrycounter`, which is more reliable. If `hyperref` hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```

12142 \ifundef\hyperlink
12143 {
12144   \newcommand*\glstrdisplaylocnameref[8]{%
12145     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
12146   }

```

12147 }

12148 {

Default action uses $\langle hcounter \rangle$. Equations and pages typically don't have a title, so check the counter name.

12149 \newcommand*{\glxstrdisplaylocnameref}[8]{%

12150 \ifstrequal{#2}{equation}%

12151 {\glxstrnameref link{#3}{(#4)}{#2.#7}{#8}}%

12152 {%

12153 \ifstreempty{#5}%

12154 {%

No title, so just use the location as the link text.

12155 \glxstrnameref link{#3}{#4}{#2.#7}{#8}%

12156 }%

12157 {%

12158 \ifstrequal{#2}{page}%

12159 {\glxstrnameref link{#3}{#4}{#2.#7}{#8}}%

12160 {\glxstrnameref link{#3}{#5}{#2.#7}{#8}}%

12161 }%

12162 }%

12163 }

12164 }

lsxtrnameref link

`\glxstrfmt nameref link{<format>}{<title>}{<href>}{<external file>}`

12165 \newcommand*{\glxstrnameref link}[4]{%

Locally change $\backslash\text{glshyper number}$ to $\backslash\text{@first of one}$ to remove the normal location hyperlink.

12166 \begingroup

12167 \let\glshyper number\@first of one

If the $\langle\text{external file}\rangle$ argument is empty, an internal link is used, otherwise an external one is needed.

12168 \ifstreempty{#4}%

12169 {\glxstrfmt internal nameref{#3}{#1}{#2}}%

12170 {\glxstrfmt external nameref{#3}{#1}{#2}{#4}}%

12171 \endgroup

12172 }

lsxtrname loc link

`\glxstrname ref loc link{<prefix>}{<counter>}{<format>}{<location>}{<text>}{<external file>}`

Like `\@gls@numberlink`, this creates a hyperlink to the target obtained from the prefix, counter and location but uses `<text>` as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

```

12173 \newcommand{\glxtrnameloclink}[6]{%
12174   \begingroup
12175   \setentrycounter[#1]{#2}%
12176   \def\glxtr@locationhypertext{#5}%
12177   \let\glshypernumber\@firstofone
12178   \def\@glsnumberformat{#3}%
12179   \def\glxtrsupplocationurl{#6}%
12180   \toks@={}%
12181   \@glxtr@bibgls@removespaces#4 \@nil
12182 \endgroup
12183 }

```

`ls@removespaces`

```

12184 \def\@glxtr@bibgls@removespaces#1 #2\@nil{%
12185   \toks@=\expandafter{\the\toks@#1}%
12186   \ifx\\#2\\%
12187     \edef\x{\the\toks@}%
12188     \ifx\x\empty
12189       \else
12190         \protected@edef\x{\glentrycounter\@glo@counterprefix\the\toks@}%
12191         \ifvoid\glxtrsupplocationurl
12192           {%
12193             \expandafter\glxtrfmtinternalnameref\expandafter{\x}%
12194             {\@glsnumberformat}{\glxtr@locationhypertext}%
12195           }%
12196         {%
12197           \expandafter\glxtrfmtexternalnameref\expandafter{\x}%
12198           {\@glsnumberformat}{\glxtr@locationhypertext}{\glxtrsupplocationurl}%
12199         }%
12200       \fi
12201     \else
12202       \@gls@ReturnAfterFi{%
12203         \@glxtr@bibgls@removespaces#2\@nil
12204       }%
12205     \fi
12206 }

```

`tinternalnameref`

<code>\glxtrfmtinternalnameloc{<target>}{<format>}{<title>}</code>
--

```

12207 \newcommand*{\glxtrfmtinternalnameref}[3]{%
12208   \csuse{#2}{\glsdohyperlink{#1}{#3}}%
12209 }

```

externalnameref

`\glstrfmtexternalnameloc{<target>}{<format>}{<title>}{<file>}`

```
12210 \newcommand*{\glstrfmtexternalnameref}[4]{%
12211   \csuse{#2}{\hyperref{#4}{}{#1}{#3}}%
12212 }
```

\glxtrSetWidest

`\glxtrSetWidest{<type>}{<level>}{<text>}`

As from **bib2gls** v1.8, this is used by the set-widest resource option for the alttree and the styles provided by the glossary-longextra package.

```
12213 \newcommand*{\glxtrSetWidest}[3]{%
```

Check which style options have been provided. (The style packages may not have been loaded.)

```
12214   \ifdef\glupdatewidest
12215   {%
12216     \ifdef\glslongextraUpdateWidest
12217     {%
```

Relevant style packages all loaded. If the *<type>* has been given, append to glossary preamble.

```
12218     \ifstrepty{#1}
12219     {%
12220       \glupdatewidest[#2]{#3}%
12221       \ifnum#2=0\relax
12222         \glslongextraUpdateWidest{#3}%
12223       \else
12224         \glslongextraUpdateWidestChild{#2}{#3}%
12225       \fi
12226     }%
12227     {%
12228       \apptoglossarypreamble[#1]{\glupdatewidest[#2]{#3}}%
12229       \ifnum#2=0\relax
12230         \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12231       \else
12232         \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12233       \fi
12234     }%
12235   }%
12236   {%
```

Only alttree.

```
12237     \ifstrepty{#1}
12238     {%
12239       \glupdatewidest[#2]{#3}%
12240     }%
```

```

12241      {%
12242      \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
12243      }%
12244      }%
12245      }%
12246      {%

```

\glsupdatewidest hasn't been defined. This could just mean that the glossaries-extra-stylemods package hasn't been loaded.

```

12247      \ifdef\glssetwidest
12248      {%
12249      \ifdef\glslongextraUpdateWidest
12250      {%

```

Relevant glossary-tree and glossary-longextra have been loaded. If the *<type>* has been given, append to glossary preamble.

```

12251      \ifstrempy{#1}
12252      {%
12253      \glssetwidest[#2]{#3}%
12254      \ifnum#2=0\relax
12255      \glslongextraUpdateWidest{#3}%
12256      \else
12257      \glslongextraUpdateWidestChild{#2}{#3}%
12258      \fi
12259      }%
12260      {%
12261      \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
12262      \ifnum#2=0\relax
12263      \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12264      \else
12265      \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12266      \fi
12267      }%
12268      }%
12269      {%

```

Only altree.

```

12270      \ifstrempy{#1}
12271      {%
12272      \glssetwidest[#2]{#3}%
12273      }%
12274      {%
12275      \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
12276      }%
12277      }%
12278      }%
12279      {%
12280      \ifdef\glslongextraUpdateWidest
12281      {%

```

glossary-longextra has been loaded.

```

12282     \ifstrempy{#1}
12283     {%
12284         \ifnum#2=0\relax
12285             \glslongextraUpdateWidest{#3}%
12286         \else
12287             \glslongextraUpdateWidestChild{#2}{#3}%
12288         \fi
12289     }%
12290     {%
12291         \ifnum#2=0\relax
12292             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12293         \else
12294             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12295         \fi
12296     }%
12297 }%

```

Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.

```

12298     }%
12299 }%
12300 }%
12301 }

```

etWidestFallback `\glxtrSetWidestFallback{<max depth>}{<list>}`

Used when **bib2gls** can't determine the widest name. The *<list>* argument is a comma-separated list of glossary labels. The *<max depth>* refers to the maximum hierarchical depth. This will either be 0 (only top-level entries) or 2 (up to two child-levels).

```

12302 \newcommand*{\glxtrSetWidestFallback}[2]{%
12303     \ifnum#1=0\relax
12304         \ifdef\glsFindWidestTopLevelName
12305             {%
12306                 \glsFindWidestTopLevelName[#2]%
12307             }%
12308         {%
12309             \GlossariesExtraWarning{You need stylemods={tree} to
12310                 provide a fallback for set-widest}%
12311         }%
12312     \else
12313         \ifdef\glsFindWidestLevelTwo
12314             {%
12315                 \glsFindWidestLevelTwo[#2]%
12316                 \ifdef\glslongextraUpdateWidestChild
12317                     {%
12318                         \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
12319                         \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
12320                     }%

```

```

12321      {}%
12322    }%
12323    {%
12324      \GlossariesExtraWarning{You need stylemods={tree} to
12325        provide a fallback for set-widest}%
12326    }%
12327  \fi
12328 }

```

`r@labelprefixes` List of label prefixes.

```
12329 \newcommand*{\@glxstr@labelprefixes}{}
```

`arlabelprefixes` List of label prefixes.

```

12330 \newcommand*{\glxstrclearlabelprefixes}{%
12331   \renewcommand*{\@glxstr@labelprefixes}{}%
12332 }

```

`raddlabelprefix` Add prefix to the list. These should be added in the order of precedence with the last one as a fallback. This doesn't check against duplicates as it may be useful to replicate a prefix at the end as the fallback.

```

12333 \newcommand*{\glxstraddlabelprefix}[1]{%
12334   \ifstrempy{#1}%
12335   {\glxstraddlabelprefix{\empty}}%
12336   {%
12337     \ifdefempty\@glxstr@labelprefixes
12338     {\def\@glxstr@labelprefixes{#1}}%
12339     {\appto\@glxstr@labelprefixes{,#1}}%
12340   }%
12341 }

```

`pendlabelprefix` Inserts at the start of the list.

```

12342 \newcommand*{\glxstrprependlabelprefix}[1]{%
12343   \ifstrempy{#1}%
12344   {\glxstrprependlabelprefix{\empty}}%
12345   {%
12346     \ifdefempty\@glxstr@labelprefixes
12347     {\def\@glxstr@labelprefixes{#1}}%
12348     {\preto\@glxstr@labelprefixes{#1,}}%
12349   }%
12350 }

```

`nlabelprefixlist` `\glxstrifinlabelprefixlist{<prefix>}{<true>}{<false>}`

Test if the given prefix is in the list.

```

12351 \newcommand*{\glxstrifinlabelprefixlist}[3]{%
12352   \ifstrempy{#1}%

```

```

12353 {\glxstrifinlabelprefixlist{\empty}{#2}{#3}}%
12354 {%
12355 \DTLifinlist{#1}{\@glxstr@labelprefixes}{#2}{#3}%
12356 }%
12357 }

```

prefixlabellist This is provided for the benefit of **bib2gls**. It's possible that the user may add more prefixes after the start of the document, but that can lead to inconsistencies. The final element of the list (the fallback) is the only prefix of interest for **bib2gls**.

```

12358 \AtBeginDocument{%
12359 \protected@write\@auxout{}\string\providecommand{\string\@glxstr@prefixlabellist}[1]{}%
12360 \protected@write\@auxout{}\string\@glxstr@prefixlabellist{\@glxstr@labelprefixes}%
12361 }

```

t@prefixedlabel Iterate through all the prefixes and find the first prefix and label combination that exists. If none found, this could mean that it's the first \LaTeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```

12362 \newcommand*{\@glxstr@get@prefixedlabel}[1]{%
12363 \begingroup

```

Initialise to the unprefix label in the event that the list is empty.

```

12364 \edef\@gls@thislabel{#1}%
12365 \@for\@glxstr@prefix:=\@glxstr@labelprefixes\do
12366 {%
12367 \edef\@gls@thislabel{\@glxstr@prefix#1}%
12368 \ifglsentryexists{\@gls@thislabel}{\@endfortrue}{}%
12369 }%
12370 \edef\x{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\x
12371 }

```

\dglS Like \gls but tries the prefixes. (Can't use \pgls as that's provided by glossaries-prefix.) Since this command is designed for bib2gls's dual entry system, the "d" stands for "dual".

```

12372 \newrobustcmd*{\dglS}{\@gls@hyp@opt\dglS}

```

\@dglS

```

12373 \newcommand*{\@dglS}[2][]{%
12374 \@glxstr@get@prefixedlabel{#2}%
12375 \new@ifnextchar[{\@gls@{#1}{\@gls@thislabel}}{\@gls@{#1}{\@gls@thislabel}}[]}%
12376 }

```

\dglSpl

```

12377 \newrobustcmd*{\dglSpl}{\@gls@hyp@opt\dglSpl}

```

\@dglSpl

```

12378 \newcommand*{\@dglSpl}[2][]{%
12379 \@glxstr@get@prefixedlabel{#2}%
12380 \new@ifnextchar[{\@glspl@{#1}{\@gls@thislabel}}{\@glspl@{#1}{\@gls@thislabel}}[]}%
12381 }

```

```
\dGls
12382 \newrobustcmd*{\dGls}{\@gls@hyp@opt\dGls}
```

```
\@dGls
12383 \newcommand*{\@dGls}[2][]{%
12384   \@glsxtr@get@prefixedlabel{#2}%
12385   \new@ifnextchar[{\@Gls@{#1}{\@gls@thislabel}}{\@Gls@{#1}{\@gls@thislabel}}[]}%
12386 }
```

```
\dGlspl
12387 \newrobustcmd*{\dGlspl}{\@gls@hyp@opt\dGlspl}
```

```
\@dGlspl
12388 \newcommand*{\@dGlspl}[2][]{%
12389   \@glsxtr@get@prefixedlabel{#2}%
12390   \new@ifnextchar[{\@Glspl@{#1}{\@gls@thislabel}}{\@Glspl@{#1}{\@gls@thislabel}}[]}%
12391 }
```

```
\dGLS
12392 \newrobustcmd*{\dGLS}{\@gls@hyp@opt\dGLS}
```

```
\@dGLS
12393 \newcommand*{\@dGLS}[2][]{%
12394   \@glsxtr@get@prefixedlabel{#2}%
12395   \new@ifnextchar[{\@GLS@{#1}{\@gls@thislabel}}{\@GLS@{#1}{\@gls@thislabel}}[]}%
12396 }
```

```
\dGLSpl
12397 \newrobustcmd*{\dGLSpl}{\@gls@hyp@opt\dGLSpl}
```

```
\@dGLSpl
12398 \newcommand*{\@dGLSpl}[2][]{%
12399   \@glsxtr@get@prefixedlabel{#2}%
12400   \new@ifnextchar[{\@GLSpl@{#1}{\@gls@thislabel}}{\@GLSpl@{#1}{\@gls@thislabel}}[]}%
12401 }
```

`\dglslink` Like `\glslink` but tries the prefixes.

```
12402 \newrobustcmd*{\dglslink}[3][]{%
12403   \@glsxtr@get@prefixedlabel{#2}%
12404   \glslink[#1]{\@gls@thislabel}{#3}%
12405 }
```

`\dglldisp` Like `\glsdisp` but tries the prefixes.

```
12406 \newrobustcmd*{\dglldisp}[3][]{%
12407   \@glsxtr@get@prefixedlabel{#2}%
12408   \glsdisp[#1]{\@gls@thislabel}{#3}%
12409 }
```

Provide missing Greek letters for use in maths mode. These commands are recognised by bib2gls and will be mapped to the Mathematical Greek Italic letters. This ensures that the Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The \LaTeX version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

`\Alpha`

12410 `\providecommand*{\Alpha}{\mathrm{A}}`

`\Beta`

12411 `\providecommand*{\Beta}{\mathrm{B}}`

`\Epsilon`

12412 `\providecommand*{\Epsilon}{\mathrm{E}}`

`\Zeta`

12413 `\providecommand*{\Zeta}{\mathrm{Z}}`

`\Eta`

12414 `\providecommand*{\Eta}{\mathrm{H}}`

`\Iota`

12415 `\providecommand*{\Iota}{\mathrm{I}}`

`\Kappa`

12416 `\providecommand*{\Kappa}{\mathrm{K}}`

`\Mu`

12417 `\providecommand*{\Mu}{\mathrm{M}}`

`\Nu`

12418 `\providecommand*{\Nu}{\mathrm{N}}`

`\Omicron`

12419 `\providecommand*{\Omicron}{\mathrm{O}}`

`\Rho`

12420 `\providecommand*{\Rho}{\mathrm{P}}`

`\Tau`

12421 `\providecommand*{\Tau}{\mathrm{T}}`

`\Chi`

12422 `\providecommand*{\Chi}{\mathrm{X}}`

```

\Digamma
12423 \providecommand*\Digamma{\mathrm{F}}

\omicron
12424 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
12425 \@ifpackageloaded{upgreek}%
12426 {

\Upalpha
12427 \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
12428 \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
12429 \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
12430 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
12431 \providecommand*\Upeta{\mathrm{H}}

\Upiota
12432 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
12433 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
12434 \providecommand*\Upmu{\mathrm{M}}

\Upnu
12435 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
12436 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
12437 \providecommand*\Uprho{\mathrm{P}}

\Uptau
12438 \providecommand*\Uptau{\mathrm{T}}

```

`\Upchi`

```
12439 \providecommand*\Upchi{\mathrm{X}}
```

`\upomicron`

```
12440 \providecommand*\upomicron{\mathrm{o}}
```

```
12441 }%
```

```
12442 {}% upgreek.sty not loaded
```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-⟨tag⟩.ldf` (where `⟨tag⟩` identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of `⟨tag⟩`. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxtrcontrolrules
; \glxtrspacerules
; \glxtrnonprintablerules
; \glxtrcombiningdiacriticrules
, \glxtrhyphenrules
< \glxtrgeneralpuncrules
< \glxtrdigitrules
< \glxtrfractionrules
< \glxtrGeneralLatinIVrules
< \glxtrMathItalicGreekIrules
}
```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they’ve been made active.

```
12443 \newcommand*\glxtrcontrolrules{%
12444 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
12445 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
12446 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
12447 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
12448 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
12449 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
12450 0010\string'\string=\glshex 0011
12451 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
12452 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
```

```

12453 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
12454 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
12455 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
12456 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
12457 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
12458 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
12459 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
12460 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
12461 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
12462 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
12463 }

```

lsxtrspacerules These are space characters.

```

12464 \newcommand*{\glsxtrspacerules}{%
12465 \string' \string'\string;
12466 \string'\glshex 00A0\string'\string;
12467 \string'\glshex 2000\string'\string;
12468 \string'\glshex 2001\string'\string;
12469 \string'\glshex 2002\string'\string;
12470 \string'\glshex 2003\string'\string;
12471 \string'\glshex 2004\string'\string;
12472 \string'\glshex 2005\string'\string;
12473 \string'\glshex 2006\string'\string;
12474 \string'\glshex 2007\string'\string;
12475 \string'\glshex 2008\string'\string;
12476 \string'\glshex 2009\string'\string;
12477 \string'\glshex 200A\string'\string;
12478 \string'\glshex 3000\string'
12479 }

```

nonprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

12480 \newcommand*{\glsxtrnonprintablerules}{%
12481 \string'\glshex FEFF\string'\string;
12482 \string'\glshex 000A\string'\string;
12483 \string'\glshex 0009\string'\string;
12484 \string'\glshex 000C\string'\string;
12485 \string'\glshex 000B\string'
12486 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

12487 \newcommand*{\glsxtrcombiningdiacriticrules}{%
12488 \glsxtrcombiningdiacriticIrules\string;
12489 \glsxtrcombiningdiacriticIIrules\string;
12490 \glsxtrcombiningdiacriticIIIrules\string;
12491 \glsxtrcombiningdiacriticIVrules
12492 }

```

diacriticIrules First set of combining diacritic marks.

```

12493 \newcommand*{\glsxtrcombiningdiacriticIrules}{%

```

```

12494 \glshex 0301\string;% combining acute
12495 \glshex 0300\string;% combining grave
12496 \glshex 0306\string;% combining breve
12497 \glshex 0302\string;% combining circumflex
12498 \glshex 030C\string;% combining caron
12499 \glshex 030A\string;% combining ring
12500 \glshex 030D\string;% combining vertical line above
12501 \glshex 0308\string;% combining diaeresis
12502 \glshex 030B\string;% combining double acute
12503 \glshex 0303\string;% combining tilde
12504 \glshex 0307\string;% combining dot above
12505 \glshex 0304% combining macron
12506 }

```

iacriticIIrules Second set of combining diacritic marks.

```

12507 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
12508 \glshex 0337\string;% combining short solidus overlay
12509 \glshex 0327\string;% combining cedilla
12510 \glshex 0328\string;% combining ogonek
12511 \glshex 0323\string;% combining dot below
12512 \glshex 0332\string;% combining low line
12513 \glshex 0305\string;% combining overline
12514 \glshex 0309\string;% combining hook above
12515 \glshex 030E\string;% combining double vertical line above
12516 \glshex 030F\string;% combining double grave accent
12517 \glshex 0310\string;% combining candrabindu
12518 \glshex 0311\string;% combining inverted breve
12519 \glshex 0312\string;% combining turned comma above
12520 \glshex 0313\string;% combining comma above
12521 \glshex 0314\string;% combining reversed comma above
12522 \glshex 0315\string;% combining comma above right
12523 \glshex 0316\string;% combining grave accent below
12524 \glshex 0317% combining acute accent below
12525 }

```

acriticIIIrules Third set of combining diacritic marks.

```

12526 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
12527 \glshex 0318\string;% combining left tack below
12528 \glshex 0319\string;% combining right tack below
12529 \glshex 031A\string;% combining left angle above
12530 \glshex 031B\string;% combining horn
12531 \glshex 031C\string;% combining left half ring below
12532 \glshex 031D\string;% combining up tack below
12533 \glshex 031E\string;% combining down tack below
12534 \glshex 031F\string;% combining plus sign below
12535 \glshex 0320\string;% combining minus sign below
12536 \glshex 0321\string;% combining palatalized hook below
12537 \glshex 0322\string;% combining retroflex hook below
12538 \glshex 0324\string;% combining diaeresis below

```

```

12539 \glshex 0325\string;% combining ring below
12540 \glshex 0326\string;% combining comma below
12541 \glshex 0329\string;% combining vertical line below
12542 \glshex 032A\string;% combining bridge below
12543 \glshex 032B\string;% combining inverted double arch below
12544 \glshex 032C\string;% combining caron below
12545 \glshex 032D\string;% combining circumflex accent below
12546 \glshex 032E\string;% combining breve below
12547 \glshex 032F\string;% combining inverted breve below
12548 \glshex 0330\string;% combining tilde below
12549 \glshex 0331\string;% combining macron below
12550 \glshex 0333\string;% combining double low line
12551 \glshex 0334\string;% combining tilde overlay
12552 \glshex 0335\string;% combining short stroke overlay
12553 \glshex 0336\string;% combining long stroke overlay
12554 \glshex 0338\string;% combining long solidus overlay
12555 \glshex 0339\string;% combining combining right half ring below
12556 \glshex 033A\string;% combining inverted bridge below
12557 \glshex 033B\string;% combining square below
12558 \glshex 033C\string;% combining seagull below
12559 \glshex 033D\string;% combining x above
12560 \glshex 033E\string;% combining vertical tilde
12561 \glshex 033F\string;% combining double overline
12562 \glshex 0342\string;% combining Greek perispomeni
12563 \glshex 0344\string;% combining Greek dialytika tonos
12564 \glshex 0345\string;% combining Greek ypogegrammeni
12565 \glshex 0360\string;% combining double tilde
12566 \glshex 0361\string;% combining double inverted breve
12567 \glshex 0483\string;% combining Cyrillic titlo
12568 \glshex 0484\string;% combining Cyrillic palatalization
12569 \glshex 0485\string;% combining Cyrillic dasia pneumata
12570 \glshex 0486% combining Cyrillic psili pneumata
12571 }

```

iacriticIVrules Fourth set of combining diacritic marks.

```

12572 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
12573 \glshex 20D0\string;% combining left harpoon above
12574 \glshex 20D1\string;% combining right harpoon above
12575 \glshex 20D2\string;% combining long vertical line overlay
12576 \glshex 20D3\string;% combining short vertical line overlay
12577 \glshex 20D4\string;% combining anticlockwise arrow above
12578 \glshex 20D5\string;% combining clockwise arrow above
12579 \glshex 20D6\string;% combining left arrow above
12580 \glshex 20D7\string;% combining right arrow above
12581 \glshex 20D8\string;% combining ring overlay
12582 \glshex 20D9\string;% combining clockwise ring overlay
12583 \glshex 20DA\string;% combining anticlockwise ring overlay
12584 \glshex 20DB\string;% combining three dots above
12585 \glshex 20DC\string;% combining four dots above

```

```

12586 \glshex 20DD\string;% combining enclosing circle
12587 \glshex 20DE\string;% combining enclosing square
12588 \glshex 20DF\string;% combining enclosing diamond
12589 \glshex 20E0\string;% combining enclosing circle backslash
12590 \glshex 20E1% combining left right arrow above
12591 }

```

sxtrhyphenrules Hyphens.

```

12592 \newcommand*{\glsxtrhyphenrules}{%
12593 \string'\string-\string'\string;% ASCII hyphen
12594 \glshex 00AD\string;% soft hyphen
12595 \glshex 2010\string;% hyphen
12596 \glshex 2011\string;% non-breaking hyphen
12597 \glshex 2012\string;% figure dash
12598 \glshex 2013\string;% en dash
12599 \glshex 2014\string;% em dash
12600 \glshex 2015\string;% horizontal bar
12601 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
12602 }

```

eneralpuncrules General punctuation.

```

12603 \newcommand*{\glsxtrgeneralpuncrules}{%
12604 \glsxtrgeneralpuncIrules
12605 \string<\glsxtrcurrencyrules
12606 \string<\glsxtrgeneralpuncIIrules
12607 }

```

eneralpuncIrules First set of general punctuation.

```

12608 \newcommand*{\glsxtrgeneralpuncIrules}{%
12609 \string'\glshex 005F\string'% underscore
12610 \string<\glshex 00AF% macron
12611 \string<\string'\glshex 002C\string'% comma
12612 \string<\string'\glshex 003B\string'% semi-colon
12613 \string<\string'\glshex 003A\string'% colon
12614 \string<\string'\glshex 0021\string'% exclamation mark
12615 \string<\glshex 00A1% inverted exclamation mark
12616 \string<\string'\glshex 003F\string'% question mark
12617 \string<\glshex 00BF% inverted question mark
12618 \string<\string'\glshex 002F\string'% solidus
12619 \string<\string'\glshex 002E\string'% full stop
12620 \string<\glshex 00B4% acute accent
12621 \string<\string'\glshex 0060\string'% grave accent
12622 \string<\string'\glshex 005E\string'% circumflex accent
12623 \string<\glshex 00A8% diaersis
12624 \string<\string'\glshex 007E\string'% tilde
12625 \string<\glshex 00B7% middle dot
12626 \string<\glshex 00B8% cedilla
12627 \string<\string'\glshex 0027\string'% straight apostrophe
12628 \string<\string'\glshex 0022\string'% straight double quote

```

```

12629 \string<\glshex 00AB% left guillemet
12630 \string<\glshex 00BB% right guillemet
12631 \string<\string'\glshex 0028\string'% left parenthesis
12632 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
12633 \string<\string'\glshex 0029\string'% right parenthesis
12634 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
12635 \string<\string'\glshex 005B\string'% left square bracket
12636 \string<\string'\glshex 005D\string'% right square bracket
12637 \string<\string'\glshex 007B\string'% left curly bracket
12638 \string<\string'\glshex 007D\string'% right curly bracket
12639 \string<\glshex 00A7% section sign
12640 \string<\glshex 00B6% pilcrow sign
12641 \string<\glshex 00A9% copyright sign
12642 \string<\glshex 00AE% registered sign
12643 \string<\string'\glshex 0040\string'% at sign
12644 }

```

trcurrencyrules General punctuation.

```

12645 \newcommand*{\glxtrcurrencyrules}{%
12646 \glshex 00A4% currency sign
12647 \string<\glshex 0E3F% Thai currency symbol baht
12648 \string<\glshex 00A2% cent sign
12649 \string<\glshex 20A1% colon sign
12650 \string<\glshex 20A2% cruzeiro sign
12651 \string<\string'\glshex 0024\string'% dollar sign
12652 \string<\glshex 20AB% dong sign
12653 \string<\glshex 20AC% euro sign
12654 \string<\glshex 20A3% French franc sign
12655 \string<\glshex 20A4% lira sign
12656 \string<\glshex 20A5% mill sign
12657 \string<\glshex 20A6% naira sign
12658 \string<\glshex 20A7% peseta sign
12659 \string<\glshex 00A3% pound sign
12660 \string<\glshex 20A8% rupee sign
12661 \string<\glshex 20AA% new sheqel sign
12662 \string<\glshex 20A9% won sign
12663 \string<\glshex 00A5% yen sign
12664 }

```

eralpuncIIrules Second set of general punctuation.

```

12665 \newcommand*{\glxtrgeneralpuncIIrules}{%
12666 \string'\glshex 002A\string'% asterisk
12667 \string<\string'\glshex 005C\string'% backslash
12668 \string<\string'\glshex 0026\string'% ampersand
12669 \string<\string'\glshex 0023\string'% hash sign
12670 \string<\string'\glshex 0025\string'% percent sign
12671 \string<\string'\glshex 002B\string'% plus sign
12672 \string=<\glshex 207A\string=<\glshex 208A% super/subscript plus sign
12673 \string<\glshex 00B1% plus-minus sign

```

```

12674 \string<\glshex 00F7% division sign
12675 \string<\glshex 00D7% multiplication sign
12676 \string<\string'\glshex 003C\string'% less-than sign
12677 \string<\string'\glshex 003D\string'% equals sign
12678 \string<\string'\glshex 003E\string'% greater-than sign
12679 \string<\glshex 00AC% not sign
12680 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12681 \string<\glshex 00A6% broken bar
12682 \string<\glshex 00B0% degree sign
12683 \string<\glshex 00B5% micron sign
12684 }

```

eralLatinIrules Basic Latin alphabet.

```

12685 \newcommand*{\glxtrGeneralLatinIrules}{%
12686 \glxtrLatinA
12687 \string<b,B%
12688 \string<c,C%
12689 \string<d,D%
12690 \string<\glxtrLatinE
12691 \string<f,F%
12692 \string<g,G%
12693 \string<\glxtrLatinH
12694 \string<\glxtrLatinI
12695 \string<j,J%
12696 \string<\glxtrLatinK
12697 \string<\glxtrLatinL
12698 \string<\glxtrLatinM
12699 \string<\glxtrLatinN
12700 \string<\glxtrLatinO
12701 \string<\glxtrLatinP
12702 \string<q,Q%
12703 \string<r,R%
12704 \string<\glxtrLatinS
12705 \string<\glxtrLatinT
12706 \string<u,U%
12707 \string<v,V%
12708 \string<w,W%
12709 \string<\glxtrLatinX
12710 \string<y,Y%
12711 \string<z,Z
12712 }

```

ralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12713 \newcommand*{\glxtrGeneralLatinIIrules}{%
12714 \glxtrLatinA
12715 \string<b,B%
12716 \string<c,C%
12717 \string<d,D%
12718 \string<\glxtrLatinEth

```

```

12719 \string<\glxtrLatinE
12720 \string<f,F%
12721 \string<g,G%
12722 \string<\glxtrLatinH
12723 \string<\glxtrLatinI
12724 \string<j,J%
12725 \string<\glxtrLatinK
12726 \string<\glxtrLatinL
12727 \string<\glxtrLatinM
12728 \string<\glxtrLatinN
12729 \string<\glxtrLatinO
12730 \string<\glxtrLatinP
12731 \string<q,Q%
12732 \string<r,R%
12733 \string<\glxtrLatinS
12734 \string& SS \string, \glxtrLatinEszettSs
12735 \string<\glxtrLatinT
12736 \string<u,U%
12737 \string<v,V%
12738 \string<w,W%
12739 \string<\glxtrLatinX
12740 \string<y,Y%
12741 \string<z,Z%
12742 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12743 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12744 \glxtrLatinA
12745 \string<b,B%
12746 \string<c,C%
12747 \string<d,D%
12748 \string<\glxtrLatinEth
12749 \string<\glxtrLatinE
12750 \string<f,F%
12751 \string<g,G%
12752 \string<\glxtrLatinH
12753 \string<\glxtrLatinI
12754 \string<j,J%
12755 \string<\glxtrLatinK
12756 \string<\glxtrLatinL
12757 \string<\glxtrLatinM
12758 \string<\glxtrLatinN
12759 \string<\glxtrLatinO
12760 \string<\glxtrLatinP
12761 \string<q,Q%
12762 \string<r,R%
12763 \string<\glxtrLatinS
12764 \string& SZ, \glxtrLatinEszettSz
12765 \string<\glxtrLatinT

```

```

12766 \string<u,U%
12767 \string<v,V%
12768 \string<w,W%
12769 \string<\glxtrLatinX
12770 \string<y,Y%
12771 \string<z,Z%
12772 }

```

GeneralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12773 \newcommand*{\glxtrGeneralLatinIVrules}{%
12774 \glxtrLatinA
12775 \string& AE , \glxtrLatinAELigature
12776 \string<b,B%
12777 \string<c,C%
12778 \string<d,D%
12779 \string<\glxtrLatinEth
12780 \string<\glxtrLatinE
12781 \string<f,F%
12782 \string<g,G%
12783 \string<\glxtrLatinH
12784 \string<\glxtrLatinI
12785 \string<j,J%
12786 \string<\glxtrLatinK
12787 \string<\glxtrLatinL
12788 \string<\glxtrLatinM
12789 \string<\glxtrLatinN
12790 \string<\glxtrLatinO
12791 \string& OE , \glxtrLatinOELigature
12792 \string<\glxtrLatinP
12793 \string<q,Q%
12794 \string<r,R%
12795 \string<\glxtrLatinS
12796 \string& SS , \glxtrLatinEszettSs
12797 \string<\glxtrLatinT
12798 \string& th =\glshex 00DE
12799 \string& TH =\glshex 00FE
12800 \string<u,U%
12801 \string<v,V%
12802 \string<w,W%
12803 \string<\glxtrLatinX
12804 \string<y,Y%
12805 \string<z,Z%
12806 }

```

GeneralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12807 \newcommand*{\glxtrGeneralLatinVrules}{%
12808 \glxtrLatinA
12809 \string<b,B%

```

```

12810 \string<c,C%
12811 \string<d,D%
12812 \string<\glxtrLatinEth
12813 \string<\glxtrLatinE
12814 \string<f,F%
12815 \string<g,G%
12816 \string<\glxtrLatinH
12817 \string<\glxtrLatinI
12818 \string<j,J%
12819 \string<\glxtrLatinK
12820 \string<\glxtrLatinL
12821 \string<\glxtrLatinM
12822 \string<\glxtrLatinN
12823 \string<\glxtrLatinO
12824 \string<\glxtrLatinP
12825 \string<q,Q%
12826 \string<r,R%
12827 \string<\glxtrLatinS
12828 \string& SS , \glxtrLatinEszettSs
12829 \string<\glxtrLatinT
12830 \string& th =\glshex 00DE
12831 \string& TH =\glshex 00FE
12832 \string<u,U%
12833 \string<v,V%
12834 \string<w,W%
12835 \string<\glxtrLatinX
12836 \string<y,Y%
12837 \string<z,Z%
12838 }

```

`\glxtrGeneralLatinVrules` General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12839 \newcommand*{\glxtrGeneralLatinVrules}{%
12840 \glxtrLatinA
12841 \string<b,B%
12842 \string<c,C%
12843 \string<d,D%
12844 \string<\glxtrLatinEth
12845 \string<\glxtrLatinE
12846 \string<f,F%
12847 \string<g,G%
12848 \string<\glxtrLatinH
12849 \string<\glxtrLatinI
12850 \string<j,J%
12851 \string<\glxtrLatinK
12852 \string<\glxtrLatinL
12853 \string<\glxtrLatinM
12854 \string<\glxtrLatinN
12855 \string<\glxtrLatinO
12856 \string<\glxtrLatinP

```

```

12857 \string<q,Q%
12858 \string<r,R%
12859 \string<\glxtrLatinS
12860 \string& SZ , \glxtrLatinEszettSz
12861 \string<\glxtrLatinT
12862 \string& th =\glshex 00DE
12863 \string& TH =\glshex 00FE
12864 \string<u,U%
12865 \string<v,V%
12866 \string<w,W%
12867 \string<\glxtrLatinX
12868 \string<y,Y%
12869 \string<z,Z%
12870 }

```

allLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, CE between O and P, long S equivalent to S, þ between T and U and wynn as W).

```

12871 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12872 \glxtrLatinA
12873 \string<\glxtrLatinAELigature
12874 \string<b,B%
12875 \string<c,C%
12876 \string<d,D%
12877 \string<\glxtrLatinEth
12878 \string<\glxtrLatinE
12879 \string<f,F%
12880 \string<\glxtrLatinInsularG
12881 \string<\glxtrLatinH
12882 \string<\glxtrLatinI
12883 \string<j,J%
12884 \string<\glxtrLatinK
12885 \string<\glxtrLatinL
12886 \string<\glxtrLatinM
12887 \string<\glxtrLatinN
12888 \string<\glxtrLatinO
12889 \string<\glxtrLatinOELigature
12890 \string<\glxtrLatinP
12891 \string<q,Q%
12892 \string<r,R%
12893 \string<\glshex 017F=\glxtrLatinS % s and long s
12894 \string<\glxtrLatinT
12895 \string<\glxtrLatinThorn
12896 \string<u,U%
12897 \string<v,V%
12898 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12899 \string<\glxtrLatinX
12900 \string<y,Y%
12901 \string<z,Z%
12902 }

```

1LatinVIIIrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

12903 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
12904 \glxtrLatinA
12905 \string& AE , \glxtrLatinAELigature
12906 \string<b,B%
12907 \string<c,C%
12908 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12909 \string<\glxtrLatinE
12910 \string<f,F%
12911 \string<g,G%
12912 \string<\glxtrLatinH
12913 \string<\glxtrLatinI
12914 \string<j,J%
12915 \string<\glxtrLatinK
12916 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12917 \string<\glxtrLatinM
12918 \string<\glxtrLatinN
12919 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12920 \string& OE , \glxtrLatinOELigature
12921 \string<\glxtrLatinP
12922 \string<q,Q%
12923 \string<r,R%
12924 \string<\glxtrLatinS
12925 \string& SS , \glxtrLatinEszettSs
12926 \string<\glxtrLatinT
12927 \string& th =\glshex 00DE
12928 \string& TH =\glshex 00FE
12929 \string<u,U%
12930 \string<v,V%
12931 \string<w,W%
12932 \string<\glxtrLatinX
12933 \string<y,Y%
12934 \string<z,Z%
12935 }
```

\glxtrLatinA

```

12936 \newcommand*{\glxtrLatinA}{%
12937 a\string=\glshex 00AA\string=\glshex 2090,A
12938 }
```

\glxtrLatinE

```

12939 \newcommand*{\glxtrLatinE}{%
12940 e\string=\glshex 2091,E
12941 }
```

\glxtrLatinH

```

12942 \newcommand*{\glxtrLatinH}{%
12943 h\string=\glshex 2095,H
```

12944 }

\glsxtrLatinI

12945 \newcommand*{\glsxtrLatinI}{%

12946 i\string=\glshex 2071,I

12947 }

\glsxtrLatinK

12948 \newcommand*{\glsxtrLatinK}{%

12949 k\string=\glshex 2096,K

12950 }

\glsxtrLatinL

12951 \newcommand*{\glsxtrLatinL}{%

12952 l\string=\glshex 2097,L

12953 }

\glsxtrLatinM

12954 \newcommand*{\glsxtrLatinM}{%

12955 m\string=\glshex 2098,M

12956 }

\glsxtrLatinN

12957 \newcommand*{\glsxtrLatinN}{%

12958 n\string=\glshex 207F\string=\glshex 2099,N

12959 }

\glsxtrLatinO

12960 \newcommand*{\glsxtrLatinO}{%

12961 o\string=\glshex 00BA\string=\glshex 2092,O

12962 }

\glsxtrLatinP

12963 \newcommand*{\glsxtrLatinP}{%

12964 p\string=\glshex 209A,P

12965 }

\glsxtrLatinS

12966 \newcommand*{\glsxtrLatinS}{%

12967 s\string=\glshex 209B,S

12968 }

\glsxtrLatinT

12969 \newcommand*{\glsxtrLatinT}{%

12970 t\string=\glshex 209C,T

12971 }

```

\glxtrLatinX
12972 \newcommand*{\glxtrLatinX}{%
12973   x\string=\glshex 2093,X
12974 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
12975 \newcommand*{\glxtrLatinSchwa}{%
12976   \glshex 0259\string=\glshex 2094,\glshex 018F
12977 }

trLatinEszettSs
12978 \newcommand*{\glxtrLatinEszettSs}{%
12979   \glshex 00DF% eszett
12980   \string=\glshex 017Fs % long S s
12981 }

trLatinEszettSz
12982 \newcommand*{\glxtrLatinEszettSz}{%
12983   \glshex 00DF% eszett
12984   \string= \glshex 017Fz % long S z
12985 }

\glxtrLatinEth
12986 \newcommand*{\glxtrLatinEth}{%
12987   \glshex 00F0,\glshex 00D0% eth
12988 }

lsxtrLatinThorn
12989 \newcommand*{\glxtrLatinThorn}{%
12990   \glshex 00FE,\glshex 00DE% thorn
12991 }

LatinAELigature
12992 \newcommand*{\glxtrLatinAELigature}{%
12993   \glshex 00E6,\glshex 00C6% AE-ligature
12994 }

LatinOELigature
12995 \newcommand*{\glxtrLatinOELigature}{%
12996   \glshex 0153,\glshex 0152% OE-ligature
12997 }

\glxtrLatinAA
12998 \newcommand*{\glxtrLatinAA}{%
12999   \glshex 00E5=a\glshex 030A,% \aa
13000   \glshex 00C5=A\glshex 030A% \AA
13001 }

```

glsxtrLatinWynn

```
13002 \newcommand*{\glsxtrLatinWynn}{%  
13003   \glsheX 01BF,\glsheX 01F7% wynn  
13004 }
```

trLatinInsularG

```
13005 \newcommand*{\glsxtrLatinInsularG}{%  
13006   \glsheX 1D79,\glsheX A77D% insular G  
13007   \string; g, G  
13008 }
```

sxtrLatinOslash

```
13009 \newcommand*{\glsxtrLatinOslash}{%  
13010   \glsheX 00F8,\glsheX 00D8% \o, \O  
13011 }
```

sxtrLatinLslash

```
13012 \newcommand*{\glsxtrLatinLslash}{%  
13013   \glsheX 0142,\glsheX 0141% \l, \L  
13014 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
13015 \newcommand*{\glsxtrMathUpGreekIrules}{%  
13016   \glsxtrUpAlpha  
13017   \string<\glsxtrUpBeta  
13018   \string<\glsxtrUpGamma  
13019   \string<\glsxtrUpDelta  
13020   \string<\glsxtrUpEpsilon  
13021   \string<\glsxtrUpDigamma  
13022   \string<\glsxtrUpZeta  
13023   \string<\glsxtrUpEta  
13024   \string<\glsxtrUpTheta  
13025   \string<\glsxtrUpIota  
13026   \string<\glsxtrUpKappa  
13027   \string<\glsxtrUpLambda  
13028   \string<\glsxtrUpMu  
13029   \string<\glsxtrUpNu  
13030   \string<\glsxtrUpXi  
13031   \string<\glsxtrUpOmicron  
13032   \string<\glsxtrUpPi  
13033   \string<\glsxtrUpRho  
13034   \string<\glsxtrUpSigma  
13035   \string<\glsxtrUpTau  
13036   \string<\glsxtrUpUpsilon  
13037   \string<\glsxtrUpPhi  
13038   \string<\glsxtrUpChi  
13039   \string<\glsxtrUpPsi  
13040   \string<\glsxtrUpOmega  
13041 }
```

`\mathUpGreekIIrules` Doesn't include digamma.

```

13042 \newcommand*{\glxtrMathUpGreekIIrules}{%
13043   \glxtrUpAlpha
13044   \string<\glxtrUpBeta
13045   \string<\glxtrUpGamma
13046   \string<\glxtrUpDelta
13047   \string<\glxtrUpEpsilon
13048   \string<\glxtrUpZeta
13049   \string<\glxtrUpEta
13050   \string<\glxtrUpTheta
13051   \string<\glxtrUpIota
13052   \string<\glxtrUpKappa
13053   \string<\glxtrUpLambda
13054   \string<\glxtrUpMu
13055   \string<\glxtrUpNu
13056   \string<\glxtrUpXi
13057   \string<\glxtrUpOmicron
13058   \string<\glxtrUpPi
13059   \string<\glxtrUpRho
13060   \string<\glxtrUpSigma
13061   \string<\glxtrUpTau
13062   \string<\glxtrUpUpsilon
13063   \string<\glxtrUpPhi
13064   \string<\glxtrUpChi
13065   \string<\glxtrUpPsi
13066   \string<\glxtrUpOmega
13067 }
```

`\mathItalicGreekIrules` Includes (upright) digamma between epsilon and zeta (there isn't an italic digamma), so don't mix with `\glxtrMathUpGreekIrules` or there may be unexpected results.

```

13068 \newcommand*{\glxtrMathItalicGreekIrules}{%
13069   \glxtrMathItalicAlpha
13070   \string<\glxtrMathItalicBeta
13071   \string<\glxtrMathItalicGamma
13072   \string<\glxtrMathItalicDelta
13073   \string<\glxtrMathItalicEpsilon
13074   \string<\glxtrUpDigamma
13075   \string<\glxtrMathItalicZeta
13076   \string<\glxtrMathItalicEta
13077   \string<\glxtrMathItalicTheta
13078   \string<\glxtrMathItalicIota
13079   \string<\glxtrMathItalicKappa
13080   \string<\glxtrMathItalicLambda
13081   \string<\glxtrMathItalicMu
13082   \string<\glxtrMathItalicNu
13083   \string<\glxtrMathItalicXi
13084   \string<\glxtrMathItalicOmicron
13085   \string<\glxtrMathItalicPi
13086   \string<\glxtrMathItalicRho
```

```

13087 \string<\glxtrMathItalicSigma
13088 \string<\glxtrMathItalicTau
13089 \string<\glxtrMathItalicUpsilon
13090 \string<\glxtrMathItalicPhi
13091 \string<\glxtrMathItalicChi
13092 \string<\glxtrMathItalicPsi
13093 \string<\glxtrMathItalicOmega
13094 }

```

`\icGreekIIrules` Doesn't include digamma.

```

13095 \newcommand*{\glxtrMathItalicGreekIIrules}{%
13096 \glxtrMathItalicAlpha
13097 \string<\glxtrMathItalicBeta
13098 \string<\glxtrMathItalicGamma
13099 \string<\glxtrMathItalicDelta
13100 \string<\glxtrMathItalicEpsilon
13101 \string<\glxtrMathItalicZeta
13102 \string<\glxtrMathItalicEta
13103 \string<\glxtrMathItalicTheta
13104 \string<\glxtrMathItalicIota
13105 \string<\glxtrMathItalicKappa
13106 \string<\glxtrMathItalicLambda
13107 \string<\glxtrMathItalicMu
13108 \string<\glxtrMathItalicNu
13109 \string<\glxtrMathItalicXi
13110 \string<\glxtrMathItalicOmicron
13111 \string<\glxtrMathItalicPi
13112 \string<\glxtrMathItalicRho
13113 \string<\glxtrMathItalicSigma
13114 \string<\glxtrMathItalicTau
13115 \string<\glxtrMathItalicUpsilon
13116 \string<\glxtrMathItalicPhi
13117 \string<\glxtrMathItalicChi
13118 \string<\glxtrMathItalicPsi
13119 \string<\glxtrMathItalicOmega
13120 }

```

`\upperGreekIrules` Upper case only (includes upright digamma).

```

13121 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
13122 \glshex 1D6E2% upper case alpha (maths italic)
13123 \string<\glshex 1D6E3% upper case beta (maths italic)
13124 \string<\glshex 1D6E4% upper case gamma (maths italic)
13125 \string<\glshex 1D6E5% upper case delta (maths italic)
13126 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13127 \string<\glshex 03DC% upper case digamma
13128 \string<\glshex 1D6E7% upper case zeta (maths italic)
13129 \string<\glshex 1D6E8% upper case eta (maths italic)
13130 \string<\glshex 1D6E9% upper case theta (maths italic)
13131 \string=<\glshex 1D6F3% upper case theta variant (maths italic)

```

```

13132 \string<\glshex 1D6EA% upper case iota (maths italic)
13133 \string<\glshex 1D6EB% upper case kappa (maths italic)
13134 \string<\glshex 1D6EC% upper case lambda (maths italic)
13135 \string<\glshex 1D6ED% upper case mu (maths italic)
13136 \string<\glshex 1D6EE% upper case nu (maths italic)
13137 \string<\glshex 1D6EF% upper case xi (maths italic)
13138 \string<\glshex 1D6F0% upper case omicron (maths italic)
13139 \string<\glshex 1D6F1% upper case pi (maths italic)
13140 \string<\glshex 1D6F2% upper case rho (maths italic)
13141 \string<\glshex 1D6F4% upper case sigma (maths italic)
13142 \string<\glshex 1D6F5% upper case tau (maths italic)
13143 \string<\glshex 1D6F6% upper case upsilon (maths italic)
13144 \string<\glshex 1D6F7% upper case phi (maths italic)
13145 \string<\glshex 1D6F8% upper case chi (maths italic)
13146 \string<\glshex 1D6F9% upper case psi (maths italic)
13147 \string<\glshex 1D6FA% upper case omega (maths italic)
13148 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

13149 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
13150 \glshex 1D6E2% upper case alpha (maths italic)
13151 \string<\glshex 1D6E3% upper case beta (maths italic)
13152 \string<\glshex 1D6E4% upper case gamma (maths italic)
13153 \string<\glshex 1D6E5% upper case delta (maths italic)
13154 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13155 \string<\glshex 1D6E7% upper case zeta (maths italic)
13156 \string<\glshex 1D6E8% upper case eta (maths italic)
13157 \string<\glshex 1D6E9% upper case theta (maths italic)
13158 \string<\glshex 1D6F3% upper case theta variant (maths italic)
13159 \string<\glshex 1D6EA% upper case iota (maths italic)
13160 \string<\glshex 1D6EB% upper case kappa (maths italic)
13161 \string<\glshex 1D6EC% upper case lambda (maths italic)
13162 \string<\glshex 1D6ED% upper case mu (maths italic)
13163 \string<\glshex 1D6EE% upper case nu (maths italic)
13164 \string<\glshex 1D6EF% upper case xi (maths italic)
13165 \string<\glshex 1D6F0% upper case omicron (maths italic)
13166 \string<\glshex 1D6F1% upper case pi (maths italic)
13167 \string<\glshex 1D6F2% upper case rho (maths italic)
13168 \string<\glshex 1D6F4% upper case sigma (maths italic)
13169 \string<\glshex 1D6F5% upper case tau (maths italic)
13170 \string<\glshex 1D6F6% upper case upsilon (maths italic)
13171 \string<\glshex 1D6F7% upper case phi (maths italic)
13172 \string<\glshex 1D6F8% upper case chi (maths italic)
13173 \string<\glshex 1D6F9% upper case psi (maths italic)
13174 \string<\glshex 1D6FA% upper case omega (maths italic)
13175 }

```

owerGreekIrules Lower case only (includes upright digamma).

```

13176 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%

```

```

13177 \glshex 1D6FC% lower case alpha (maths italic)
13178 \string<\glshex 1D6FD% lower case beta (maths italic)
13179 \string<\glshex 1D6FE% lower case gamma (maths italic)
13180 \string<\glshex 1D6FF% lower case delta (maths italic)
13181 \string<\glshex 1D700% lower case epsilon (maths italic)
13182 \string=\glshex 1D716% lower case epsilon variant (maths italic)
13183 \string<\glshex 03DD% lower case digamma
13184 \string<\glshex 1D701% lower case zeta (maths italic)
13185 \string<\glshex 1D702% lower case eta (maths italic)
13186 \string<\glshex 1D703% lower case theta (maths italic)
13187 \string=\glshex 1D717% lower case theta variant (maths italic)
13188 \string<\glshex 1D704% lower case iota (maths italic)
13189 \string<\glshex 1D705% lower case kappa (maths italic)
13190 \string=\glshex 1D718% lower case kappa variant (maths italic)
13191 \string<\glshex 1D706% lower case lambda (maths italic)
13192 \string<\glshex 1D707% lower case mu (maths italic)
13193 \string<\glshex 1D708% lower case nu (maths italic)
13194 \string<\glshex 1D709% lower case xi (maths italic)
13195 \string<\glshex 1D70A% lower case omicron (maths italic)
13196 \string<\glshex 1D70B% lower case pi (maths italic)
13197 \string=\glshex 1D71B% lower case pi variant (maths italic)
13198 \string<\glshex 1D70C% lower case rho (maths italic)
13199 \string=\glshex 1D71A% lower case rho variant (maths italic)
13200 \string<\glshex 1D70D% lower case final sigma (maths italic)
13201 \string=\glshex 1D70E% lower case sigma (maths italic)
13202 \string<\glshex 1D70F% lower case tau (maths italic)
13203 \string<\glshex 1D710% lower case upsilon (maths italic)
13204 \string<\glshex 1D711% lower case phi (maths italic)
13205 \string=\glshex 1D719% lower case phi variant (maths italic)
13206 \string<\glshex 1D712% lower case chi (maths italic)
13207 \string<\glshex 1D713% lower case psi (maths italic)
13208 \string<\glshex 1D714% lower case omega (maths italic)
13209 }

```

LowerGreeklIrules Lower case only (doesn't includes upright digamma).

```

13210 \newcommand*{\glxtrMathItalicLowerGreeklIrules}{%
13211 \glshex 1D6FC% lower case alpha (maths italic)
13212 \string<\glshex 1D6FD% lower case beta (maths italic)
13213 \string<\glshex 1D6FE% lower case gamma (maths italic)
13214 \string<\glshex 1D6FF% lower case delta (maths italic)
13215 \string<\glshex 1D700% lower case epsilon (maths italic)
13216 \string=\glshex 1D716% lower case epsilon variant (maths italic)
13217 \string<\glshex 1D701% lower case zeta (maths italic)
13218 \string<\glshex 1D702% lower case eta (maths italic)
13219 \string<\glshex 1D703% lower case theta (maths italic)
13220 \string=\glshex 1D717% lower case theta variant (maths italic)
13221 \string<\glshex 1D704% lower case iota (maths italic)
13222 \string<\glshex 1D705% lower case kappa (maths italic)
13223 \string=\glshex 1D718% lower case kappa variant (maths italic)

```

```

13224 \string<\glshex 1D706% lower case lambda (maths italic)
13225 \string<\glshex 1D707% lower case mu (maths italic)
13226 \string<\glshex 1D708% lower case nu (maths italic)
13227 \string<\glshex 1D709% lower case xi (maths italic)
13228 \string<\glshex 1D70A% lower case omicron (maths italic)
13229 \string<\glshex 1D70B% lower case pi (maths italic)
13230 \string=<\glshex 1D71B% lower case pi variant (maths italic)
13231 \string<\glshex 1D70C% lower case rho (maths italic)
13232 \string=<\glshex 1D71A% lower case rho variant (maths italic)
13233 \string<\glshex 1D70D% lower case final sigma (maths italic)
13234 \string=<\glshex 1D70E% lower case sigma (maths italic)
13235 \string<\glshex 1D70F% lower case tau (maths italic)
13236 \string<\glshex 1D710% lower case upsilon (maths italic)
13237 \string<\glshex 1D711% lower case phi (maths italic)
13238 \string=<\glshex 1D719% lower case phi variant (maths italic)
13239 \string<\glshex 1D712% lower case chi (maths italic)
13240 \string<\glshex 1D713% lower case psi (maths italic)
13241 \string<\glshex 1D714% lower case omega (maths italic)
13242 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

13243 \newcommand*{\glxtrMathGreekIrules}{%
13244 \glxtrMathItalicAlpha
13245 \string;\glxtrUpAlpha
13246 \string<\glxtrMathItalicBeta
13247 \string;\glxtrUpBeta
13248 \string<\glxtrMathItalicGamma
13249 \string;\glxtrUpGamma
13250 \string<\glxtrMathItalicDelta
13251 \string;\glxtrUpDelta
13252 \string<\glxtrMathItalicEpsilon
13253 \string;\glxtrUpEpsilon
13254 \string<\glxtrUpDigamma
13255 \string<\glxtrMathItalicZeta
13256 \string;\glxtrUpZeta
13257 \string<\glxtrMathItalicEta
13258 \string;\glxtrUpEta
13259 \string<\glxtrMathItalicTheta
13260 \string;\glxtrUpTheta
13261 \string<\glxtrMathItalicIota
13262 \string;\glxtrUpIota
13263 \string<\glxtrMathItalicKappa
13264 \string;\glxtrUpKappa
13265 \string<\glxtrMathItalicLambda
13266 \string;\glxtrUpLambda
13267 \string<\glxtrMathItalicMu
13268 \string;\glxtrUpMu
13269 \string<\glxtrMathItalicNu
13270 \string;\glxtrUpNu

```

```

13271 \string<\glxtrMathItalicXi
13272 \string;\glxtrUpXi
13273 \string<\glxtrMathItalicOmicron
13274 \string;\glxtrUpOmicron
13275 \string<\glxtrMathItalicPi
13276 \string;\glxtrUpPi
13277 \string<\glxtrMathItalicRho
13278 \string;\glxtrUpRho
13279 \string<\glxtrMathItalicSigma
13280 \string;\glxtrUpSigma
13281 \string<\glxtrMathItalicTau
13282 \string;\glxtrUpTau
13283 \string<\glxtrMathItalicUpsilon
13284 \string;\glxtrUpUpsilon
13285 \string<\glxtrMathItalicPhi
13286 \string;\glxtrUpPhi
13287 \string<\glxtrMathItalicChi
13288 \string;\glxtrUpChi
13289 \string<\glxtrMathItalicPsi
13290 \string;\glxtrUpPsi
13291 \string<\glxtrMathItalicOmega
13292 \string;\glxtrUpOmega
13293 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

13294 \newcommand*{\glxtrMathGreekIIrules}{%
13295 \glxtrMathItalicAlpha
13296 \string;\glxtrUpAlpha
13297 \string<\glxtrMathItalicBeta
13298 \string;\glxtrUpBeta
13299 \string<\glxtrMathItalicGamma
13300 \string;\glxtrUpGamma
13301 \string<\glxtrMathItalicDelta
13302 \string;\glxtrUpDelta
13303 \string<\glxtrMathItalicEpsilon
13304 \string;\glxtrUpEpsilon
13305 \string<\glxtrMathItalicZeta
13306 \string;\glxtrUpZeta
13307 \string<\glxtrMathItalicEta
13308 \string;\glxtrUpEta
13309 \string<\glxtrMathItalicTheta
13310 \string;\glxtrUpTheta
13311 \string<\glxtrMathItalicIota
13312 \string;\glxtrUpIota
13313 \string<\glxtrMathItalicKappa
13314 \string;\glxtrUpKappa
13315 \string<\glxtrMathItalicLambda
13316 \string;\glxtrUpLambda
13317 \string<\glxtrMathItalicMu

```

```

13318 \string;\glxtrUpMu
13319 \string<\glxtrMathItalicNu
13320 \string;\glxtrUpNu
13321 \string<\glxtrMathItalicXi
13322 \string;\glxtrUpXi
13323 \string<\glxtrMathItalicOmicron
13324 \string;\glxtrUpOmicron
13325 \string<\glxtrMathItalicPi
13326 \string;\glxtrUpPi
13327 \string<\glxtrMathItalicRho
13328 \string;\glxtrUpRho
13329 \string<\glxtrMathItalicSigma
13330 \string;\glxtrUpSigma
13331 \string<\glxtrMathItalicTau
13332 \string;\glxtrUpTau
13333 \string<\glxtrMathItalicUpsilon
13334 \string;\glxtrUpUpsilon
13335 \string<\glxtrMathItalicPhi
13336 \string;\glxtrUpPhi
13337 \string<\glxtrMathItalicChi
13338 \string;\glxtrUpChi
13339 \string<\glxtrMathItalicPsi
13340 \string;\glxtrUpPsi
13341 \string<\glxtrMathItalicOmega
13342 \string;\glxtrUpOmega
13343 }

```

`\glxtrUpAlpha`

```

13344 \newcommand*{\glxtrUpAlpha}{%
13345 \glshex 03B1,% lower case alpha
13346 \glshex 0391% upper case alpha
13347 }

```

`\glxtrUpBeta`

```

13348 \newcommand*{\glxtrUpBeta}{%
13349 \glshex 03B2,% lower case beta
13350 \glshex 0392% upper case beta
13351 }

```

`\glxtrUpGamma`

```

13352 \newcommand*{\glxtrUpGamma}{%
13353 \glshex 03B3,% lower case gamma
13354 \glshex 0393% upper case gamma
13355 }

```

`\glxtrUpDelta`

```

13356 \newcommand*{\glxtrUpDelta}{%
13357 \glshex 03B4,% lower case delta
13358 \glshex 0394% upper case delta

```

13359 }

glsxtrUpEpsilon

```
13360 \newcommand*{\glsxtrUpEpsilon}{%
13361   \glsheX 03B5% lower case epsilon
13362   \string=\glsheX 03F5,% lower case epsilon variant
13363   \glsheX 0395% upper case epsilon
13364 }
```

glsxtrUpDigamma

```
13365 \newcommand*{\glsxtrUpDigamma}{%
13366   \glsheX 03DD,% lower case digamma
13367   \glsheX 03DC% upper case digamma
13368 }
```

\glsxtrUpZeta

```
13369 \newcommand*{\glsxtrUpZeta}{%
13370   \glsheX 03B6,% lower case zeta
13371   \glsheX 0396% upper case zeta
13372 }
```

\glsxtrUpEta

```
13373 \newcommand*{\glsxtrUpEta}{%
13374   \glsheX 03B7,% lower case eta
13375   \glsheX 0397% upper case eta
13376 }
```

\glsxtrUpTheta

```
13377 \newcommand*{\glsxtrUpTheta}{%
13378   \glsheX 03B8% lower case theta
13379   \string=\glsheX 03D1,% lower case theta variant
13380   \glsheX 0398% upper case theta
13381 }
```

\glsxtrUpIota

```
13382 \newcommand*{\glsxtrUpIota}{%
13383   \glsheX 03B9,% lower case iota
13384   \glsheX 0399% upper case iota
13385 }
```

\glsxtrUpKappa

```
13386 \newcommand*{\glsxtrUpKappa}{%
13387   \glsheX 03BA% lower case kappa
13388   \string=\glsheX 03F0,% lower case kappa variant
13389   \glsheX 039A% upper case kappa
13390 }
```

\glxtrUpLambda

```
13391 \newcommand*{\glxtrUpLambda}{%
13392   \glshex 03BB,% lower lambda
13393   \glshex 039B% upper case lambda
13394 }
```

\glxtrUpMu

```
13395 \newcommand*{\glxtrUpMu}{%
13396   \glshex 03BC,% lower case mu
13397   \glshex 039C% upper case mu
13398 }
```

\glxtrUpNu

```
13399 \newcommand*{\glxtrUpNu}{%
13400   \glshex 03BD,% lower case nu
13401   \glshex 039D% upper case nu
13402 }
```

\glxtrUpXi

```
13403 \newcommand*{\glxtrUpXi}{%
13404   \glshex 03BE,% lower case xi
13405   \glshex 039E% upper case xi
13406 }
```

\glxtrUpOmicron

```
13407 \newcommand*{\glxtrUpOmicron}{%
13408   \glshex 03BF,% lower case omicron
13409   \glshex 039F% upper case omicron
13410 }
```

\glxtrUpPi

```
13411 \newcommand*{\glxtrUpPi}{%
13412   \glshex 03C0% lower case pi
13413   \string=\glshex 03D6,% lower case pi variant
13414   \glshex 03A0% upper case pi
13415 }
```

\glxtrUpRho

```
13416 \newcommand*{\glxtrUpRho}{%
13417   \glshex 03C1% lower case rho
13418   \string=\glshex 03F1,% lower case rho variant
13419   \glshex 03A1% upper case rho
13420 }
```

\glxtrUpSigma

```
13421 \newcommand*{\glxtrUpSigma}{%
13422   \glshex 03C2% lower case sigma
13423   \string=\glshex 03C3,% lower case sigma
```

```

13424 \glshex 03A3% upper case sigma
13425 }

```

`\glxtrUpTau`

```

13426 \newcommand*{\glxtrUpTau}{%
13427 \glshex 03C4,% lower case tau
13428 \glshex 03A4% upper case tau
13429 }

```

`glxtrUpUpsilon`

```

13430 \newcommand*{\glxtrUpUpsilon}{%
13431 \glshex 03C5,% lower case upsilon
13432 \glshex 03A5% upper case upsilon
13433 }

```

`\glxtrUpPhi`

```

13434 \newcommand*{\glxtrUpPhi}{%
13435 \glshex 03C6% lower case phi
13436 \string=\glshex 03D5,% lower case phi variant
13437 \glshex 03A6% upper case phi
13438 }

```

`\glxtrUpChi`

```

13439 \newcommand*{\glxtrUpChi}{%
13440 \glshex 03C7,% lower case chi
13441 \glshex 03A7% upper case chi
13442 }

```

`\glxtrUpPsi`

```

13443 \newcommand*{\glxtrUpPsi}{%
13444 \glshex 03C8,% lower case psi
13445 \glshex 03A8% upper case psi
13446 }

```

`\glxtrUpOmega`

```

13447 \newcommand*{\glxtrUpOmega}{%
13448 \glshex 03C9,% lower case omega
13449 \glshex 03A9% upper case omega
13450 }

```

`MathItalicAlpha`

```

13451 \newcommand*{\glxtrMathItalicAlpha}{%
13452 \glshex 1D6FC,% lower case alpha (maths italic)
13453 \glshex 1D6E2% upper case alpha (maths italic)
13454 }

```

`rMathItalicBeta`

```

13455 \newcommand*{\glxtrMathItalicBeta}{%

```

```

13456 \glshex 1D6FD,% lower case beta (maths italic)
13457 \glshex 1D6E3% upper case beta (maths italic)
13458 }

```

MathItalicGamma

```

13459 \newcommand*{\glxtrMathItalicGamma}{%
13460 \glshex 1D6FE,% lower case gamma (maths italic)
13461 \glshex 1D6E4% upper case gamma (maths italic)
13462 }

```

MathItalicDelta

```

13463 \newcommand*{\glxtrMathItalicDelta}{%
13464 \glshex 1D6FF,% lower case delta (maths italic)
13465 \glshex 1D6E5% upper case delta (maths italic)
13466 }

```

MathItalicEpsilon

```

13467 \newcommand*{\glxtrMathItalicEpsilon}{%
13468 \glshex 1D700% lower case epsilon (maths italic)
13469 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
13470 \glshex 1D6E6% upper case epsilon (maths italic)
13471 }

```

MathItalicZeta

```

13472 \newcommand*{\glxtrMathItalicZeta}{%
13473 \glshex 1D701,% lower case zeta (maths italic)
13474 \glshex 1D6E7% upper case zeta (maths italic)
13475 }

```

MathItalicEta

```

13476 \newcommand*{\glxtrMathItalicEta}{%
13477 \glshex 1D702,% lower case eta (maths italic)
13478 \glshex 1D6E8% upper case eta (maths italic)
13479 }

```

MathItalicTheta

```

13480 \newcommand*{\glxtrMathItalicTheta}{%
13481 \glshex 1D703% lower case theta (maths italic)
13482 \string=\glshex 1D717,% lower case theta variant (maths italic)
13483 \glshex 1D6E9% upper case theta (maths italic)
13484 \string=\glshex 1D6F3% upper case theta variant (maths italic)
13485 }

```

MathItalicIota

```

13486 \newcommand*{\glxtrMathItalicIota}{%
13487 \glshex 1D704,% lower case iota (maths italic)
13488 \glshex 1D6EA% upper case iota (maths italic)
13489 }

```

MathItalicKappa

```
13490 \newcommand*{\glxtrMathItalicKappa}{%
13491   \glshex 1D705% lower case kappa (maths italic)
13492   \string=\glshex 1D718,% lower case kappa variant (maths italic)
13493   \glshex 1D6EB% upper case kappa (maths italic)
13494 }
```

MathItalicLambda

```
13495 \newcommand*{\glxtrMathItalicLambda}{%
13496   \glshex 1D706,% lower case lambda (maths italic)
13497   \glshex 1D6EC% upper case lambda (maths italic)
13498 }
```

MathItalicMu

```
13499 \newcommand*{\glxtrMathItalicMu}{%
13500   \glshex 1D707,% lower case mu (maths italic)
13501   \glshex 1D6ED% upper case mu (maths italic)
13502 }
```

MathItalicNu

```
13503 \newcommand*{\glxtrMathItalicNu}{%
13504   \glshex 1D708,% lower case nu (maths italic)
13505   \glshex 1D6EE% upper case nu (maths italic)
13506 }
```

MathItalicXi

```
13507 \newcommand*{\glxtrMathItalicXi}{%
13508   \glshex 1D709,% lower case xi (maths italic)
13509   \glshex 1D6EF% upper case xi (maths italic)
13510 }
```

MathItalicOmicron

```
13511 \newcommand*{\glxtrMathItalicOmicron}{%
13512   \glshex 1D70A,% lower case omicron (maths italic)
13513   \glshex 1D6F0% upper case omicron (maths italic)
13514 }
```

MathItalicPi

```
13515 \newcommand*{\glxtrMathItalicPi}{%
13516   \glshex 1D70B% lower case pi (maths italic)
13517   \string=\glshex 1D71B,% lower case pi variant (maths italic)
13518   \glshex 1D6F1% upper case pi (maths italic)
13519 }
```

MathItalicRho

```
13520 \newcommand*{\glxtrMathItalicRho}{%
13521   \glshex 1D70C% lower case rho (maths italic)
13522   \string=\glshex 1D71A,% lower case rho variant (maths italic)
```

```

13523 \glshex 1D6F2% upper case rho (maths italic)
13524 }

```

MathItalicSigma

```

13525 \newcommand*{\glsxtrMathItalicSigma}{%
13526 \glshex 1D70D% lower case final sigma (maths italic)
13527 \string=\glshex 1D70E,% lower case sigma (maths italic)
13528 \glshex 1D6F4% upper case sigma (maths italic)
13529 }

```

trMathItalicTau

```

13530 \newcommand*{\glsxtrMathItalicTau}{%
13531 \glshex 1D70F,% lower case tau (maths italic)
13532 \glshex 1D6F5% upper case tau (maths italic)
13533 }

```

thItalicUpsilon

```

13534 \newcommand*{\glsxtrMathItalicUpsilon}{%
13535 \glshex 1D710,% lower case upsilon (maths italic)
13536 \glshex 1D6F6% upper case upsilon (maths italic)
13537 }

```

trMathItalicPhi

```

13538 \newcommand*{\glsxtrMathItalicPhi}{%
13539 \glshex 1D711% lower case phi (maths italic)
13540 \string=\glshex 1D719,% lower case phi variant (maths italic)
13541 \glshex 1D6F7% upper case phi (maths italic)
13542 }

```

trMathItalicChi

```

13543 \newcommand*{\glsxtrMathItalicChi}{%
13544 \glshex 1D712,% lower case chi (maths italic)
13545 \glshex 1D6F8% upper case chi (maths italic)
13546 }

```

trMathItalicPsi

```

13547 \newcommand*{\glsxtrMathItalicPsi}{%
13548 \glshex 1D713,% lower case psi (maths italic)
13549 \glshex 1D6F9% upper case psi (maths italic)
13550 }

```

MathItalicOmega

```

13551 \newcommand*{\glsxtrMathItalicOmega}{%
13552 \glshex 1D714,% lower case omega (maths italic)
13553 \glshex 1D6FA% upper case omega (maths italic)
13554 }

```

thItalicPartial

```
13555 \newcommand*{\glxtrMathItalicPartial}{%
13556   \glshex 1D715% partial differential (maths italic)
13557 }
```

MathItalicNabla

```
13558 \newcommand*{\glxtrMathItalicNabla}{%
13559   \glshex 1D6FB% nabla (maths italic)
13560 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
13561 \newcommand*{\glxtrdigitrules}{%
13562   0\string=\glshex 2080\string=\glshex 2070
13563   \string<1\string=\glshex 2081\string=\glshex 00B9
13564   \string<2\string=\glshex 2082\string=\glshex 00B2
13565   \string<3\string=\glshex 2083\string=\glshex 00B3
13566   \string<4\string=\glshex 2084\string=\glshex 2074
13567   \string<5\string=\glshex 2085\string=\glshex 2075
13568   \string<6\string=\glshex 2086\string=\glshex 2076
13569   \string<7\string=\glshex 2087\string=\glshex 2077
13570   \string<8\string=\glshex 2088\string=\glshex 2078
13571   \string<9\string=\glshex 2089\string=\glshex 2079
13572 }
```

BasicDigitrules Digits from the Basic Latin set.

```
13573 \newcommand*{\glxtrBasicDigitrules}{%
13574   0\string<1\string<2\string<3\string<4%
13575   \string<5\string<6\string<7\string<8\string<9%
13576 }
```

criptDigitrules Subscript digits.

```
13577 \newcommand*{\glxtrSubScriptDigitrules}{%
13578   \glshex 2080% subscript 0
13579   \string<\glshex 2081% subscript 1
13580   \string<\glshex 2082% subscript 2
13581   \string<\glshex 2083% subscript 3
13582   \string<\glshex 2084% subscript 4
13583   \string<\glshex 2085% subscript 5
13584   \string<\glshex 2086% subscript 6
13585   \string<\glshex 2087% subscript 7
13586   \string<\glshex 2088% subscript 8
13587   \string<\glshex 2089% subscript 9
13588 }
```

criptDigitrules Superscript digits.

```
13589 \newcommand*{\glxtrSuperScriptDigitrules}{%
13590   \glshex 2070% superscript 0
13591   \string<\glshex 00B9% superscript 1
```

```

13592 \string<\glshex 00B2% superscript 2
13593 \string<\glshex 00B3% superscript 3
13594 \string<\glshex 2074% superscript 4
13595 \string<\glshex 2075% superscript 5
13596 \string<\glshex 2076% superscript 6
13597 \string<\glshex 2077% superscript 7
13598 \string<\glshex 2078% superscript 8
13599 \string<\glshex 2079% superscript 9
13600 }

```

trfractionrules Vulgar fractions.

```

13601 \newcommand*{\glxtrfractionrules}{%
13602   \glshex 215F% fraction numerator one (1/)
13603   \string<\glshex 2189% zero thirds (0/3 = 0)
13604   \string<\glshex 2152% one tenth (1/10 = 0.1)
13605   \string<\glshex 2151% one ninth (1/9 ~ 0.111)
13606   \string<\glshex 215B% one eighth (1/8 = 0.125)
13607   \string<\glshex 2150% one seventh (1/7 ~ 0.143)
13608   \string<\glshex 2159% one sixth (1/6 ~ 0.167)
13609   \string<\glshex 2155% one fifth (1/5 = 0.2)
13610   \string<\glshex 00BC% one quarter (1/4 = 0.25)
13611   \string<\glshex 2153% one third (1/3 ~ 0.333)
13612   \string<\glshex 215C% three eighths (3/8 = 0.375)
13613   \string<\glshex 2156% two fifths (2/5 = 0.4)
13614   \string<\glshex 00BD% one half (1/2 = 0.5)
13615   \string<\glshex 2157% three fifths (3/5 = 0.6)
13616   \string<\glshex 215D% five eighths (5/8 = 0.625)
13617   \string<\glshex 2154% two thirds (2/3 ~ 0.667)
13618   \string<\glshex 00BE% three quarters (3/4 = 0.75)
13619   \string<\glshex 2158% four fifths (4/5 = 0.8)
13620   \string<\glshex 215A% five sixths (5/6 ~ 0.833)
13621   \string<\glshex 215E% seven eighths (7/8 = 0.875)
13622 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

13623 \renewcommand{\@glxtrdialecthook}{%
13624   \ifundef\CurrentTrackedScript
13625   {%
13626     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13627     {%
13628       \edef\CurrentTrackedScript{%
13629         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13630     }%
13631   }%
13632 }%
13633 {}%
13634 \ifdef\CurrentTrackedScript
13635 {%
13636   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix

```

```

13637 \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
13638 \let\CurrentTrackedTag\CurrentTrackedScript
13639 \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.1df}
13640 {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13641 {}%
13642 \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13643 }%
13644 {}%
13645 }

```

If `\glsxtr@loaddialect` has been defined, then `glossaries-extra-bib2gls` has been loaded after `glossaries-extra`. (For example, through `\glossariesextrasetup`.) Not recommended, but if this has been done try to find the associated language resources.

```

13646 \ifdef\glsxtr@loaddialect
13647 {%
13648 \@ifpackageloaded{tracklang}
13649 {%
13650 \AnyTrackedLanguages
13651 {%
13652 \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13653 }%
13654 {}%
13655 }
13656 {}
13657 }
13658 {}

```

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:

```
13659 \NeedsTeXFormat{LaTeX2e}
13660 \ProvidesPackage{glossaries-extra-stylemods}[2019/03/22 1.39 (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`

```
13661 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13662 \DeclareOption{all}{%
13663   \appto\@glxtr@loadstyles{%
13664     \RequirePackage{glossary-inline}%
13665     \RequirePackage{glossary-list}%
13666     \RequirePackage{glossary-tree}%
13667     \RequirePackage{glossary-mcols}%
13668     \RequirePackage{glossary-long}%
13669     \RequirePackage{glossary-longragged}%
13670     \RequirePackage{glossary-longbooktabs}%
13671     \RequirePackage{glossary-super}%
13672     \RequirePackage{glossary-superragged}%
13673     \RequirePackage{glossary-bookindex}%
13674     \RequirePackage{glossary-longextra}%
13675   }
13676 }

13677 \DeclareOption*{%
13678   \IfFileExists{glossary-\CurrentOption.sty}
13679   {\eappto\@glxtr@loadstyles{%
13680     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13681   }%
13682   {%
```

```

13683     \PackageError{glossaries-extra-styles}%
13684     {Unknown option ‘\CurrentOption’}{}%
13685   }%
13686 }

```

Process the package options:

```
13687 \ProcessOptions
```

Load the required packages:

```
13688 \@glsxtr@loadstyles
```

Adjust the styles so that they all have the post description hook. Also, instead of having a hard-coded \space before the location, use:

sxtrprelocation This uses \providecommand as the same command is also provided by glossary-bookindex.

```
13689 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

ewglossarystyle

```

13690 \providecommand{\renewglossarystyle}[2]{%
13691   \ifcsundef{@glsstyle@#1}%
13692   {%
13693     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13694   }%
13695   {%
13696     \csdef{@glsstyle@#1}{#2}%
13697   }%
13698 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying to add this.

```

13699 \ifdef{\@glsstyle@listdotted}
13700 {%
13701   \renewglossarystyle{listdotted}{%
13702     \setglossarystyle{list}%
13703     \renewcommand*{\glossentry}[2]{%
13704       \item[]\makebox[\glslistdottedwidth][l]{%
13705         \glsentryitem{##1}%
13706         \glstarget{##1}{\glossentryname{##1}}%
13707         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13708         \glossentrydesc{##1}\glspostdescription}%
13709     \renewcommand*{\subglossentry}[3]{%
13710       \item[]\makebox[\glslistdottedwidth][l]{%
13711         \glssubentryitem{##2}%
13712         \glstarget{##2}{\glossentryname{##2}}%
13713         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13714         \glossentrydesc{##2}\glspostdescription}%

```

```

13715 }
13716 }
13717 {%

```

Assume the style isn't required if it hasn't already been defined.

```

13718 }

```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

The other list styles would be easier to adapt if the space before the number list wasn't hard coded.

```

13719 \ifdef{\@glsstyle@list}
13720 {%

```

listprelocation Space before number list for top-level entries.

```

13721 \newcommand{\glslistprelocation}{\glstrprelocation}

```

childprelocation Space before number list for child entries.

```

13722 \newcommand{\glslistchildprelocation}{\glslistprelocation}

```

childpostlocation Full stop after number list.

```

13723 \newcommand{\glslistchildpostlocation}{.}

```

\glslistdesc

```

13724 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}

```

Redefine list to use these commands.

```

13725 \renewglossarystyle{list}{%
13726 \renewenvironment{theglossary}%
13727 {\begin{description}}{\end{description}}%
13728 \renewcommand*\{glossaryheader}{}%
13729 \renewcommand*\{glsgroupheading}[1]{}%
13730 \renewcommand*\{glossentry}[2]{%
13731 \item[\glssentryitem{##1}]%
13732 \glstarget{##1}{\glossentryname{##1}}]
13733 \glslistdesc{##1}\glslistprelocation ##2}%
13734 \renewcommand*\{subglossentry}[3]{%
13735 \glssubentryitem{##2}%
13736 \glstarget{##2}{\strut}\space
13737 \glslistdesc{##2}%
13738 \glslistchildprelocation ##3\glslistchildpostlocation}%
13739 \renewcommand*\{glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13740 }
13741 }
13742 {}

```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```

13743 \ifdef{\@glsstyle@altlist}
13744 {%

```

```

13745 \renewglossarystyle{altlist}{%
13746   \setglossarystyle{list}%
13747   \renewcommand*{\glossentry}[2]{%
13748     \item[\glsentryitem{##1}%
13749       \glstarget{##1}{\glossentryname{##1}}]%
13750     \mbox{}\par\nobreak\@afterheading
13751     \glslistdesc{##1}\glslistprelocation ##2}%
13752   \renewcommand{\subglossentry}[3]{%
13753     \par
13754     \glssubentryitem{##2}%
13755     \glstarget{##2}{\strut}\glslistdesc{##2}%
13756     \glslistchildprelocation ##3}%
13757 }
13758 }
13759 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13760 \ifdef{\@glsstyle@listgroup}
13761 {%
13762   \renewglossarystyle{listgroup}{%
13763     \setglossarystyle{list}%
13764     \renewcommand*{\glsgroupheading}[1]{%
13765       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13766       \mbox{}\par\nobreak\@afterheading
13767     }%
13768   }
13769 }
13770 {}

```

Similarly for listhypergroup.

```

13771 \ifdef{\@glsstyle@listhypergroup}
13772 {%
13773   \renewglossarystyle{listhypergroup}{%
13774     \setglossarystyle{list}%
13775     \renewcommand*{\glossaryheader}{%
13776       \glslistnavigationitem{\glsnavigation}}%
13777     \renewcommand*{\glsgroupheading}[1]{%
13778       \item[\glslistgroupheaderfmt
13779         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13780       \mbox{}\par\nobreak\@afterheading
13781     }%
13782   }
13783 }
13784 {}

```

Similarly for altlistgroup.

```

13785 \ifdef{\@glsstyle@altlistgroup}
13786 {%
13787   \renewglossarystyle{altlistgroup}{%
13788     \setglossarystyle{altlist}%
13789     \renewcommand*{\glsgroupheading}[1]{%

```

```

13790     \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13791     \mbox{}\par\nobreak\@afterheading
13792 }%
13793 }
13794 }
13795 {}

```

Similarly for `altlisthypergroup`.

```

13796 \ifdef{\@glsstyle@altlisthypergroup}
13797 {%
13798   \renewglossarystyle{altlisthypergroup}{%
13799     \setglossarystyle{altlist}%
13800     \renewcommand*\glossaryheader{%
13801       \glslistnavigationitem{\glsnavigation}}%
13802     \renewcommand*\glsgroupheading[1]{%
13803       \item[\glslistgroupheaderfmt
13804         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13805       \mbox{}\par\nobreak\@afterheading
13806     }%
13807   }
13808 }
13809 {}

```

2.3 Longtable Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glstrprelocation`.

```

13810 \ifcsdef{@glsstyle@long}
13811 {%
13812   \renewglossarystyle{long}{%
13813     \renewenvironment{theglossary}%
13814       {\begin{longtable}{lp{\glsdescwidth}}}%
13815       {\end{longtable}}%
13816     \renewcommand*\glossaryheader{}%
13817     \renewcommand*\glsgroupheading[1]{}%
13818     \renewcommand{\glossentry}[2]{%
13819       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13820       \glossentrydesc{##1}\glspostdescription
13821       \glstrprelocation ##2\tabularnewline
13822     }%
13823     \renewcommand{\subglossentry}[3]{%
13824       &
13825       \glssubentryitem{##2}%
13826       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13827       \glstrprelocation ##3\tabularnewline
13828     }%
13829     \ifglsgroupskip
13830       \renewcommand*\glsgroupskip{}%

```

```

13831 \else
13832 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13833 \fi
13834 }
13835 }
13836 {}

```

Three column style:

```

13837 \ifcsdef{@glsstyle@long3col}
13838 {%
13839 \renewglossarystyle{long3col}{%
13840 \renewenvironment{theglossary}%
13841 {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13842 {\end{longtable}}}%
13843 \renewcommand*{\glossaryheader}{}%
13844 \renewcommand*{\glsgroupheading}[1]{}%
13845 \renewcommand{\glossentry}[2]{%
13846 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13847 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13848 }%
13849 \renewcommand{\subglossentry}[3]{%
13850 &
13851 \glssubentryitem{##2}%
13852 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13853 ##3\tabularnewline
13854 }%

```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```

13855 \ifglsnogroupskip
13856 \renewcommand*{\glsgroupskip}{}%
13857 \else
13858 \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13859 \fi
13860 }
13861 }
13862 {}

```

Four column style:

```

13863 \ifcsdef{@glsstyle@long4col}
13864 {%
13865 \renewglossarystyle{long4col}{%
13866 \renewenvironment{theglossary}%
13867 {\begin{longtable}{llll}}}%
13868 {\end{longtable}}}%
13869 \renewcommand*{\glossaryheader}{}%
13870 \renewcommand*{\glsgroupheading}[1]{}%
13871 \renewcommand{\glossentry}[2]{%
13872 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13873 \glossentrydesc{##1}\glspostdescription &
13874 \glossentrysymbol{##1} &

```

```

13875     ##2\tabularnewline
13876 }%
13877 \renewcommand{\subglossentry}[3]{%
13878     &
13879     \glssubentryitem{##2}%
13880     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13881     \glossentrysymbol{##2} & ##3\tabularnewline
13882 }%

13883 \ifglsnogroupskip
13884     \renewcommand*\{glsgroupskip}{}%
13885 \else
13886     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13887 \fi
13888 }
13889 }
13890 {}

```

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 for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glxtrprelocation`.

```

13891 \ifcsdef{@glstyle@longragged}
13892 {%
13893     \renewglossarystyle{longragged}{%
13894         \renewenvironment{theglossary}%
13895             {\begin{longtable}{l>\raggedright}p{\glstdescwidth}}}%
13896             {\end{longtable}}}%
13897     \renewcommand*\{glossaryheader}{}%
13898     \renewcommand*\{glsgroupheading}[1]{}%
13899     \renewcommand{\glossentry}[2]{%
13900         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13901         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13902         \tabularnewline
13903     }%
13904     \renewcommand{\subglossentry}[3]{%
13905         &
13906         \glssubentryitem{##2}%
13907         \glstarget{##2}{\strut}\glossentrydesc{##2}%
13908         \glspostdescription\glxtrprelocation ##3%
13909         \tabularnewline
13910     }%
13911     \ifglsnogroupskip
13912         \renewcommand*\{glsgroupskip}{}%
13913     \else

```

```

13914     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13915     \fi
13916   }
13917 }
13918 {}

```

Three and four column styles don't use `\glsxtrprelocation` since the number list is in its own column.

```

13919 \ifcsdef{@glsstyle@longragged3col}
13920 {%
13921   \renewglossarystyle{longragged3col}{%
13922     \renewenvironment{theglossary}%
13923       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13924         >{\raggedright}p{\glspagelistwidth}}}%
13925       {\end{longtable}}}%
13926     \renewcommand*{\glossaryheader}{}%
13927     \renewcommand*{\glsgroupheading}[1]{}%
13928     \renewcommand{\glossentry}[2]{%
13929       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13930       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13931     }%
13932     \renewcommand{\subglossentry}[3]{%
13933       &
13934       \glssubentryitem{##2}%
13935       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13936       ##3\tabularnewline
13937     }%
13938   \ifglsnogroupskip
13939     \renewcommand*{\glsgroupskip}{}%
13940   \else
13941     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13942   \fi
13943 }
13944 }
13945 {}

```

Four column style:

```

13946 \ifcsdef{@glsstyle@altlongragged4col}
13947 {%
13948   \renewglossarystyle{altlongragged4col}{%
13949     \renewenvironment{theglossary}%
13950       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13951         >{\raggedright}p{\glspagelistwidth}}}%
13952       {\end{longtable}}}%
13953     \renewcommand*{\glossaryheader}{}%
13954     \renewcommand*{\glsgroupheading}[1]{}%
13955     \renewcommand{\glossentry}[2]{%
13956       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13957       \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &

```

```

13958     ##2\tabularnewline
13959 }%
13960 \renewcommand{\subglossentry}[3]{%
13961     &
13962     \glssubentryitem{##2}%
13963     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13964     \glossentrysymbol{##2} & ##3\tabularnewline
13965 }%

13966 \ifglsnogroupskip
13967     \renewcommand*\{glsgroupskip\}{}%
13968 \else
13969     \renewcommand*\{glsgroupskip\}{& & \tabularnewline}%
13970 \fi
13971 }
13972 }
13973 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded \space changed to \glxtrprelocation.

```

13974 \ifcsdef{@glstyle@super}
13975 {%
13976     \renewglossarystyle{super}{%
13977         \renewenvironment{theglossary}%
13978             {\tablehead{}\tabletail{}}%
13979             \begin{supertabular}{lp{\glsdescwidth}}%
13980             {\end{supertabular}}%
13981         \renewcommand*\{glossaryheader\}{}%
13982         \renewcommand*\{glsgroupheading}[1]{}%
13983         \renewcommand{\glossentry}[2]{%
13984             \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13985             \glossentrydesc{##1}\glspostdescription
13986             \glxtrprelocation ##2\tabularnewline
13987         }%
13988         \renewcommand{\subglossentry}[3]{%
13989             &
13990             \glssubentryitem{##2}%
13991             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13992             \glxtrprelocation ##3\tabularnewline
13993         }%
13994         \ifglsnogroupskip
13995             \renewcommand*\{glsgroupskip\}{}%
13996         \else
13997             \renewcommand*\{glsgroupskip\}{& \tabularnewline}%
13998         \fi
13999     }

```

```

14000 }
14001 {}

    Three column style:
14002 \ifcsdef{@glsstyle@super3col}
14003 {%
14004     \renewglossarystyle{super3col}{%
14005         \renewenvironment{theglossary}%
14006             {\tablehead{}\tabletail{}}%
14007             \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
14008             {\end{supertabular}}%
14009         \renewcommand*{\glossaryheader}{}%
14010         \renewcommand*{\glsgroupheading}[1]{}%
14011         \renewcommand{\glossentry}[2]{%
14012             \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14013             \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14014         }%
14015         \renewcommand{\subglossentry}[3]{%
14016             &
14017             \glssubentryitem{##2}%
14018             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14019             ##3\tabularnewline
14020         }%

14021         \ifglsnogroupskip
14022             \renewcommand*{\glsgroupskip}{}%
14023         \else
14024             \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14025         \fi
14026     }
14027 }
14028 {}

```

Four column styles:

```

14029 \ifcsdef{@glsstyle@super4col}
14030 {%
14031     \renewglossarystyle{super4col}{%
14032         \renewenvironment{theglossary}%
14033             {\tablehead{}\tabletail{}}%
14034             \begin{supertabular}{llll}}}%
14035             \end{supertabular}}%
14036         \renewcommand*{\glossaryheader}{}%
14037         \renewcommand*{\glsgroupheading}[1]{}%
14038         \renewcommand{\glossentry}[2]{%
14039             \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14040             \glossentrydesc{##1}\glspostdescription &
14041             \glossentrysymbol{##1} & ##2\tabularnewline
14042         }%
14043         \renewcommand{\subglossentry}[3]{%
14044             &

```

```

14045     \glssubentryitem{##2}%
14046     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14047     \glossentrysymbol{##2} & ##3\tabularnewline
14048 }%

14049 \ifglsnogroupskip
14050     \renewcommand*{\glsgroupskip}{}%
14051 \else
14052     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
14053 \fi
14054 }
14055 }
14056 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glxtrprelocation`.

```

14057 \ifcsdef{@glstyle@superragged}
14058 {%
14059     \renewglossarystyle{superragged}{%
14060         \renewenvironment{theglossary}%
14061             {\tablehead{}}\tabletail{}}%
14062         \begin{supertabular}{l>{\raggedright}p{\glsglwidth}}%
14063             {\end{supertabular}}%
14064         \renewcommand*{\glossaryheader}{}%
14065         \renewcommand*{\glsgroupheading}[1]{}%
14066         \renewcommand{\glossentry}[2]{%
14067             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14068             \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
14069             \tabularnewline
14070         }%
14071         \renewcommand{\subglossentry}[3]{%
14072             &
14073             \glssubentryitem{##2}%
14074             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
14075             \glxtrprelocation ##3%
14076             \tabularnewline
14077         }%
14078         \ifglsnogroupskip
14079             \renewcommand*{\glsgroupskip}{}%
14080         \else
14081             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14082         \fi
14083     }
14084 }
14085 {}

```

Three column style:

```
14086 \ifcsdef{@glsstyle@superragged3col}
14087 {%
14088   \renewglossarystyle{superragged3col}{%
14089     \renewenvironment{theglossary}%
14090       {\tablehead{}\tabletail{}}%
14091       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
14092         >{\raggedright}p{\glspagelistwidth}}}%
14093       {\end{supertabular}}}%
14094   \renewcommand*{\glossaryheader}{}%
14095   \renewcommand*{\glsgroupheading}[1]{}%
14096   \renewcommand{\glossentry}[2]{%
14097     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
14098     \glsentrydesc{##1}\glspostdescription &
14099     ##2\tabularnewline
14100   }%
14101   \renewcommand{\subglossentry}[3]{%
14102     &
14103     \glssubentryitem{##2}%
14104     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
14105     ##3\tabularnewline
14106   }%
14107   \ifglsnogroupskip
14108     \renewcommand*{\glsgroupskip}{}%
14109   \else
14110     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14111   \fi
14112 }
14113 }
14114 {}
```

Four columns:

```
14115 \ifcsdef{@glsstyle@altsuperragged4col}
14116 {%
14117   \renewglossarystyle{altsuperragged4col}{%
14118     \renewenvironment{theglossary}%
14119       {\tablehead{}\tabletail{}}%
14120       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
14121         >{\raggedright}p{\glspagelistwidth}}}%
14122       {\end{supertabular}}}%
14123   \renewcommand*{\glossaryheader}{}%
14124   \renewcommand{\glossentry}[2]{%
14125     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
14126     \glsentrydesc{##1}\glspostdescription &
14127     \glsentrysymbol{##1} & ##2\tabularnewline
14128   }%
14129   \renewcommand{\subglossentry}[3]{%
14130     &
14131     \glssubentryitem{##2}%

```

```

14132      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14133      \glossentrysymbol{##2} & ##3\tabularnewline
14134    }%

14135    \ifglsnogroupskip
14136      \renewcommand*{\glsgroupskip}{}%
14137    \else
14138      \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
14139    \fi
14140  }
14141 }
14142 {}

```

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`.

```

14143 \ifdef{\@glsstyle@inline}
14144 {%
14145   \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}

   Just use \glxtrpostdescription instead of \glspostdescription.

14146   \renewcommand*{\glsinlinedescformat}[3]{%
14147     \space#1\glxtrpostdescription}
14148   \renewcommand*{\glsinlinesubdescformat}[3]{%
14149     #1\glxtrpostdescription}

```

The default settings don't show the location lists, so there's no adjustment for `\glxtrprelocation`.

```

14150 }
14151 {}

```

2.8 Tree Styles

Redefine both `\glstreenamefmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamefmt` to make it easier to change both at the same time or only change one without affecting the other.

```

14152 \ifdef\glstreenamefmt
14153 {
\glstreedefaultnamefmt
14154   \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}

\glstreenamefmt
14155   \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}

```

egroupheaderfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14156 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

eenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14157 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

```
14158 }
```

```
14159 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
14160 \ifdef{\@glsstyle@index}
```

```
14161 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
14162 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
14163 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
14164 \renewglossarystyle{index}{%
14165   \renewenvironment{theglossary}%
14166     {\setlength{\parindent}{0pt}%
14167      \setlength{\parskip}{0pt plus 0.3pt}%
14168      \let\item\glstreeitem
14169      \let\subitem\glstreesubitem
14170      \let\subsubitem\glstreesubsubitem
14171     }%
14172   {\par}%
14173   \renewcommand*{\glossaryheader}{}%
14174   \renewcommand*{\glsgroupheading}[1]{}%
14175   \renewcommand*{\glossentry}[2]{%
14176     \item\glssentryitem{##1}%
14177     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14178     \glstreesymbol{##1}%
14179     \glstreedesc{##1}%
14180     \glstreeprelocation ##2%
14181   }%
14182   \renewcommand{\subglossentry}[3]{%
14183     \ifcase##1\relax
14184       \item
14185     \or
14186       \subitem
14187       \glssubentryitem{##2}%
14188     \else
14189       \subsubitem
14190     \fi
14191     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
14192     \glstreechildsymbol{##2}%
```

```

14193     \glstreechilddesc{##2}%
14194     \glstreechildprelocation ##3%
14195 }%
14196 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14197 }
14198 }
14199 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

14200 \ifdef{\@glsstyle@indexgroup}
14201 {%
14202   \renewglossarystyle{indexgroup}{%
14203     \setglossarystyle{index}%
14204     \renewcommand*{\glsgroupheading}[1]{%
14205       \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
14206       \nopagebreak\indexspace
14207       \nobreak\@afterheading
14208     }%
14209   }
14210 }
14211 {}

```

Similarly for indexhypergroup.

```

14212 \ifdef{\@glsstyle@indexhypergroup}
14213 {%
14214   \renewglossarystyle{indexhypergroup}{%
14215     \setglossarystyle{index}%
14216     \renewcommand*{\glossaryheader}{%
14217       \item\glstreenavigationfmt{\glsnavigation}%
14218       \nobreak\@afterheading\indexspace}%
14219     \renewcommand*{\glsgroupheading}[1]{%
14220       \item\glstreegroupheaderfmt
14221         {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}%
14222       \nopagebreak\indexspace
14223       \nobreak\@afterheading}%
14224   }%
14225 }
14226 {}

```

Adjust tree style to remove hard coded space before number list.

```

14227 \ifdef{\@glsstyle@tree}
14228 {%

```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```

14229   \newcommand{\glstreedesc}[1]{%
14230     \glstreedeprepare\glossentrydesc{##1}\glspostdescription
14231   }

```

Similarly for the symbol.

`\glstreesymbol`

```
14232 \newcommand{\glstreesymbol}[1]{%
14233   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
14234 }%
```

And for the child entries:

`lstreechilddesc`

```
14235 \newcommand{\glstreechilddesc}[1]{%
14236   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
14237 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
14238 \newcommand{\glstreechildsymbol}[1]{%
14239   \glstreesymbol{#1}%
14240 }%

14241 \renewglossarystyle{tree}{%
14242   \renewenvironment{theglossary}%
14243     {\setlength{\parindent}{0pt}%
14244     \setlength{\parskip}{0pt plus 0.3pt}}%
14245   {%
14246     \renewcommand*{\glossaryheader}{}%
14247     \renewcommand*{\glsgroupheading}[1]{}%
14248     \renewcommand{\glossentry}[2]{%
14249       \hangindent0pt\relax
14250       \parindent0pt\relax
14251       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14252       \glstreesymbol{##1}%
14253       \glstreedesc{##1}%
14254       \glstreeprelocation##2\par
14255     }%
14256     \renewcommand{\subglossentry}[3]{%
14257       \hangindent##1\glstreeindent\relax
14258       \parindent##1\glstreeindent\relax
14259       \ifnum##1=1\relax
14260         \glssubentryitem{##2}%
14261       \fi
14262       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
14263       \glstreechildsymbol{##2}%
14264       \glstreechilddesc{##2}%
14265       \glstreechildprelocation ##3\par
14266     }%
14267     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14268   }%
14269 }
14270 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
14271 \ifdef{\@glstyle@treegroup}
```

```

14272 {%
14273   \renewglossarystyle{treegroup}{%
14274     \setglossarystyle{tree}%
14275     \renewcommand{\glsgroupheading}[1]{\par
14276       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
14277       \nopagebreak\indexspace\nobreak\@afterheading}%
14278   }
14279 }
14280 {}

```

Similarly for treehypergroup

```

14281 \ifdef{\@glsstyle@treehypergroup}
14282 {%
14283   \renewglossarystyle{treehypergroup}{%
14284     \setglossarystyle{tree}%
14285     \renewcommand*\{\glossaryheader}{%
14286       \par\noindent\glstreenavigationfmt{\glstravigation}\par
14287       \nobreak\@afterheading\indexspace}%
14288     \renewcommand*\{\glsgroupheading}[1]{%
14289       \par\noindent
14290       \glstreegroupheaderfmt
14291       {\glstravhypertarget{##1}{\glsgrouptitle{##1}}}\par
14292       \nopagebreak\indexspace\nobreak\@afterheading}%
14293   }
14294 }
14295 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

14296 \ifdef{\@glsstyle@treenoname}
14297 {%

```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```

14298   \newcommand{\glstreenonamedesc}[1]{%
14299     \glstreepredesc\glossentrydesc{##1}\glspostdescription
14300   }%

```

Similarly for the symbol.

treenonamesymbol

```

14301   \newcommand{\glstreenonamesymbol}[1]{%
14302     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
14303   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

14304   \newcommand{\glstreenonamechilddesc}[1]{%
14305     \glossentrydesc{##1}\glspostdescription
14306   }%

```

```

14307 \renewglossarystyle{treenoname}{%
14308   \renewenvironment{theglossary}%
14309     {\setlength{\parindent}{0pt}%
14310      \setlength{\parskip}{0pt plus 0.3pt}}%
14311   {}%
14312   \renewcommand*{\glossaryheader}{}%
14313   \renewcommand*{\glsgroupheading}[1]{}%
14314   \renewcommand{\glossentry}[2]{%
14315     \hangindent0pt\relax
14316     \parindent0pt\relax
14317     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14318     \glstreenonamesymbol{##1}%
14319     \glstreenonamedesc{##1}%
14320     \glstreeprelocation##2\par
14321   }%
14322   \renewcommand{\subglossentry}[3]{%
14323     \hangindent##1\glstreeindent\relax
14324     \parindent##1\glstreeindent\relax
14325     \ifnum##1=1\relax
14326       \glssubentryitem{##2}%
14327     \fi
14328     \glstarget{##2}{\strut}%
14329     \glstreenonamechilddesc{##2}%
14330     \glstreechildprelocation##3\par
14331   }%
14332   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14333 }
14334 }
14335 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

14336 \ifdef{\@glsstyle@treenonamegroup}
14337 {%
14338   \renewglossarystyle{treenonamegroup}{%
14339     \setglossarystyle{treenoname}%
14340     \renewcommand{\glsgroupheading}[1]{\par
14341       \noindent\glstreegroupheaderfmt
14342       {\glsgetgrouptitle{##1}}}%
14343     \nopagebreak\indexspace\nobreak\@afterheading
14344   }%
14345 }
14346 }
14347 {}

```

Similarly for `treenonamehypergroup`

```

14348 \ifdef{\@glsstyle@treenonamehypergroup}
14349 {%
14350   \renewglossarystyle{treenonamehypergroup}{%
14351     \setglossarystyle{treenoname}%
14352     \renewcommand*{\glossaryheader}{%

```

```

14353     \par\noindent\glstreenavigationfmt{\glsnavigation}\par
14354     \nobreak\@afterheading\indexspace}%
14355 \renewcommand*{\glsgroupheading}[1]{%
14356     \par\noindent
14357     \glstreegroupheaderfmt
14358     {\glsnavhypertarget{##1}{\glsgroupgetgrouptitle{##1}}}%
14359     \nopagebreak\indexspace\nobreak\@afterheading}%
14360 }
14361 }
14362 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

14363 \ifdef{\glstyle@almtree}
14364 {}%

```

Only redefine this style if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

14365 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
14366     {%
14367         \let\par\glxtrAltTreePar
14368         \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
14369         \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
14370     }%
14371 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

14372 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

14373 \newcommand{\glxtrAltTreePar}{%
14374     \@@par
14375     \glxtrAltTreeSetHangIndent
14376     \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
14377 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

14378 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
14379     \glxtralttreeSymbolDescLocation{#2}{#3}%
14380 }

```

`trtreetopindent` 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.

```
14381 \newlength\glxstrtreetopindent
```

`sxtralttreeInit` User-level initialisation for the `alttree` style.

```
14382 \newcommand*{\glxtralttreeInit}{%
14383   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
14384   \glxstrAltTreeIndent=\parindent
14385 }
```

`\gglsetwidest` The original `\glsetwidest` only uses `\def`. This uses `\gdef`.

```
14386 \newcommand*{\gglsetwidest}[2][0]{%
14387   \csgdef{@glswidestname\romannumeral#1}{#2}%
14388 }
```

`\eglssetwidest` The original `\glsetwidest` only uses `\def`. This uses `\protected@csedef`.

```
14389 \newcommand*{\eglssetwidest}[2][0]{%
14390   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14391 }
```

`\xglsetwidest` Like the above but uses `\protected@csxdef`.

```
14392 \newcommand*{\xglsetwidest}[2][0]{%
14393   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14394 }
```

`glupdatewidest` Only sets if new value is wider than old value.

```
14395 \newcommand*{\glupdatewidest}[2][0]{%
14396   \ifcsundef{@glswidestname\romannumeral#1}%
14397   {\csdef{@glswidestname\romannumeral#1}{#2}}%
14398   {%
14399     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14400     \settowidth{\dimen@ii}{#2}%
14401     \ifdim\dimen@ii>\dimen@
14402       \csdef{@glswidestname\romannumeral#1}{#2}%
14403     \fi
14404   }%
14405 }
```

`glupdatewidest` As above but global definition.

```
14406 \newcommand*{\gglupdatewidest}[2][0]{%
14407   \ifcsundef{@glswidestname\romannumeral#1}%
14408   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
14409   {%
14410     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14411     \settowidth{\dimen@ii}{#2}%
14412     \ifdim\dimen@ii>\dimen@
14413       \csgdef{@glswidestname\romannumeral#1}{#2}%
14414     \fi
14415 }
```

```

14415 }%
14416 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

14417 \newcommand*{\eglsupdatewidest}[2][0]{%
14418   \ifcsundef{@glswidestname\romannumeral#1}%
14419   {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
14420   {%
14421     \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
14422     \settowidth{\dimen@ii}{#2}%
14423     \ifdim\dimen@ii>\dimen0
14424     \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14425     \fi
14426   }%
14427 }

```

`glsupdatewidest` As above but global.

```

14428 \newcommand*{\xglsupdatewidest}[2][0]{%
14429   \ifcsundef{@glswidestname\romannumeral#1}%
14430   {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
14431   {%
14432     \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
14433     \settowidth{\dimen@ii}{#2}%
14434     \ifdim\dimen@ii>\dimen0
14435     \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14436     \fi
14437   }%
14438 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

14439 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

14440 \newcommand*{\glsgetwidestsubname}[1]{%
14441   \ifcsundef{@glswidestname\romannumeral#1}%
14442   {\@glswidestname}%
14443   {\csuse{@glswidestname\romannumeral#1}}%
14444 }

```

`glsFindWidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```

14445 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname

```

`glsFindWidestUsedTopLevelName` 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.

```

14446 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
14447   \dimen@=0pt\relax

```

```

14448 \gls@tmplen=Opt\relax
14449 \foralllglossaries[#1]{\@gls@type}%
14450 {%
14451 \forglsentries[\@gls@type]{\@glo@label}%
14452 {%
14453 \ifglsused{\@glo@label}%
14454 {%
14455 \ifglsashasparent{\@glo@label}%
14456 {}%
14457 {%
14458 \settowidth{\dimen@}%
14459 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14460 \ifdim\dimen@>\gls@tmplen
14461 \gls@tmplen=\dimen@
14462 \eglssetwidest{\glsentryname{\@glo@label}}%
14463 \fi
14464 }%
14465 }%
14466 {}%
14467 }%
14468 }%
14469 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

14470 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
14471 \dimen@=Opt\relax
14472 \gls@tmplen=Opt\relax
14473 \foralllglossaries[#1]{\@gls@type}%
14474 {%
14475 \forglsentries[\@gls@type]{\@glo@label}%
14476 {%
14477 \ifglsused{\@glo@label}%
14478 {%
14479 \settowidth{\dimen@}%
14480 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14481 \ifdim\dimen@>\gls@tmplen
14482 \gls@tmplen=\dimen@
14483 \eglssetwidest{\glsentryname{\@glo@label}}%
14484 \fi
14485 }%
14486 {}%
14487 }%
14488 }%
14489 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

14490 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
14491 \dimen@=Opt\relax

```

```

14492 \gls@tmplen=0pt\relax
14493 \foralllglossaries[#1]{\@gls@type}%
14494 {%
14495   \forallgsentries[\@gls@type]{\@glo@label}%
14496   {%
14497     \settowidth{\dimen@}%
14498     {\glstreenamfmt{\glstryname{\@glo@label}}}%
14499     \ifdim\dimen@>\gls@tmplen
14500       \gls@tmplen=\dimen@
14501       \eglssetwidest{\glstryname{\@glo@label}}%
14502     \fi
14503   }%
14504 }%
14505 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

14506 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
14507   \dimen@=0pt\relax
14508   \dimen@i=0pt\relax
14509   \dimen@ii=0pt\relax
14510   \foralllglossaries[#1]{\@gls@type}%
14511   {%
14512     \forallgsentries[\@gls@type]{\@glo@label}%
14513     {%
14514       \ifglused{\@glo@label}%
14515       {%
14516         \ifglshasparent{\@glo@label}%
14517         {%
14518           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
14519           \ifglshasparent{\@glo@parent}%
14520           {%
14521             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
14522             \ifglshasparent{\@glo@parent}%
14523             }%
14524           {%
14525             \settowidth{\gls@tmplen}%
14526             {\glstreenamfmt{\glstryname{\@glo@label}}}%
14527             \ifdim\gls@tmplen>\dimen@ii
14528               \dimen@ii=\gls@tmplen
14529               \eglssetwidest[2]{\glstryname{\@glo@label}}%
14530             \fi
14531           }%
14532         }%
14533       }%
14534       \settowidth{\gls@tmplen}%
14535       {\glstreenamfmt{\glstryname{\@glo@label}}}%
14536       \ifdim\gls@tmplen>\dimen@i
14537         \dimen@i=\gls@tmplen

```

```

14538         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14539     \fi
14540 }%
14541 }%
14542 {%
14543     \settowidth{\gls@tmplen}%
14544         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14545     \ifdim\gls@tmplen>\dimen@
14546         \dimen@=\gls@tmplen
14547         \eglssetwidest{\glsentryname{\@glo@label}}%
14548     \fi
14549 }%
14550 }%
14551 {}%
14552 }%
14553 }%
14554 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

14555 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
14556     \dimen@=0pt\relax
14557     \dimen@i=0pt\relax
14558     \dimen@ii=0pt\relax
14559     \foralllglossaries[#1]{\@gls@type}%
14560     {%
14561         \forglsentries[\@gls@type]{\@glo@label}%
14562         {%
14563             \ifglshasparent{\@glo@label}%
14564             {%
14565                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
14566                 \ifglshasparent{\@glo@parent}%
14567                 {%
14568                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
14569                     \ifglshasparent{\@glo@parent}%
14570                     {}%
14571                 }%
14572                 \settowidth{\gls@tmplen}%
14573                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14574                 \ifdim\gls@tmplen>\dimen@ii
14575                     \dimen@ii=\gls@tmplen
14576                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
14577                 \fi
14578             }%
14579         }%
14580     {%
14581         \settowidth{\gls@tmplen}%
14582             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14583         \ifdim\gls@tmplen>\dimen@i
14584             \dimen@i=\gls@tmplen

```

```

14585         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14586     \fi
14587 }%
14588 }%
14589 {%
14590     \settowidth{\gls@tmplen}%
14591         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14592     \ifdim\gls@tmplen>\dimen@
14593         \dimen@=\gls@tmplen
14594         \eglssetwidest{\glsentryname{\@glo@label}}%
14595     \fi
14596 }%
14597 }%
14598 }%
14599 }

```

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.

```

14600 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
14601     \dimen@=0pt\relax
14602     \gls@tmplen=0pt\relax
14603     #2=0pt\relax
14604     \foralllglossaries[#1]{\@gls@type}%
14605     {%
14606         \forglsentries[\@gls@type]{\@glo@label}%
14607         {%
14608             \ifglsused{\@glo@label}%
14609             {%
14610                 \settowidth{\dimen@}%
14611                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14612                 \ifdim\dimen@>\gls@tmplen
14613                     \gls@tmplen=\dimen@
14614                     \eglssetwidest{\glsentryname{\@glo@label}}%
14615                 \fi
14616                 \settowidth{\dimen@}%
14617                     {\glsentrysymbol{\@glo@label}}%
14618                 \ifdim\dimen@>#2\relax
14619                     #2=\dimen@
14620                 \fi
14621             }%
14622         }%
14623     }%
14624 }%
14625 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

14626 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14627     \dimen@=0pt\relax
14628     \gls@tmplen=0pt\relax

```

```

14629      #2=Opt\relax
14630      \foralllglossaries[#1]{\@gls@type}%
14631      {%
14632        \forglsentries[\@gls@type]{\@glo@label}%
14633        {%
14634          \settowidth{\dimen@}%
14635            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14636          \ifdim\dimen@>\gls@tmplen
14637            \gls@tmplen=\dimen@
14638            \eglssetwidest{\glsentryname{\@glo@label}}%
14639          \fi
14640          \settowidth{\dimen@}%
14641            {\glsentrysymbol{\@glo@label}}%
14642          \ifdim\dimen@>#2\relax
14643            #2=\dimen@
14644          \fi
14645        }%
14646      }%
14647    }

```

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.

```

14648    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14649      \dimen@=Opt\relax
14650      \gls@tmplen=Opt\relax
14651      #2=Opt\relax
14652      #3=Opt\relax
14653      \foralllglossaries[#1]{\@gls@type}%
14654      {%
14655        \forglsentries[\@gls@type]{\@glo@label}%
14656        {%
14657          \ifglsused{\@glo@label}%
14658          {%
14659            \settowidth{\dimen@}%
14660              {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14661            \ifdim\dimen@>\gls@tmplen
14662              \gls@tmplen=\dimen@
14663              \eglssetwidest{\glsentryname{\@glo@label}}%
14664            \fi
14665            \settowidth{\dimen@}%
14666              {\glsentrysymbol{\@glo@label}}%
14667            \ifdim\dimen@>#2\relax
14668              #2=\dimen@
14669            \fi
14670            \settowidth{\dimen@}%
14671              {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14672            \ifdim\dimen@>#3\relax

```

```

14673         #3=\dimen@
14674     \fi
14675 }%
14676 {}%
14677 }%
14678 }%
14679 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14680 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14681     \dimen@=0pt\relax
14682     \gls@tmplen=0pt\relax
14683     #2=0pt\relax
14684     #3=0pt\relax
14685     \forallglossaries[#1]{\@gls@type}%
14686     {%
14687         \forglsentries[\@gls@type]{\@glo@label}%
14688         {%
14689             \settowidth{\dimen@}%
14690                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14691             \ifdim\dimen@>\gls@tmplen
14692                 \gls@tmplen=\dimen@
14693                 \eglssetwidest{\glsentryname{\@glo@label}}%
14694             \fi
14695             \settowidth{\dimen@}%
14696                 {\glsentrysymbol{\@glo@label}}%
14697             \ifdim\dimen@>#2\relax
14698                 #2=\dimen@
14699             \fi
14700             \settowidth{\dimen@}%
14701                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14702             \ifdim\dimen@>#3\relax
14703                 #3=\dimen@
14704             \fi
14705         }%
14706     }%
14707 }

```

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.

```

14708 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14709     \dimen@=0pt\relax
14710     \gls@tmplen=0pt\relax
14711     #2=0pt\relax
14712     \forallglossaries[#1]{\@gls@type}%
14713     {%
14714         \forglsentries[\@gls@type]{\@glo@label}%
14715         {%

```

```

14716      \ifglsused{\@glo@label}%
14717      {%
14718          \settowidth{\dimen@}%
14719              {\glstreenamefmt{\glentryname{\@glo@label}}}%
14720          \ifdim\dimen@>\gls@tmplen
14721              \gls@tmplen=\dimen@
14722              \eglssetwidest{\glentryname{\@glo@label}}%
14723          \fi
14724          \settowidth{\dimen@}%
14725              {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14726          \ifdim\dimen@>#2\relax
14727              #2=\dimen@
14728          \fi
14729      }%
14730      {}%
14731  }%
14732  }%
14733  }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14734  \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14735      \dimen@=0pt\relax
14736      \gls@tmplen=0pt\relax
14737      #2=0pt\relax
14738      \forallglossaries[#1]{\@gls@type}%
14739      {%
14740          \forglsentries[\@gls@type]{\@glo@label}%
14741          {%
14742              \settowidth{\dimen@}%
14743                  {\glstreenamefmt{\glentryname{\@glo@label}}}%
14744              \ifdim\dimen@>\gls@tmplen
14745                  \gls@tmplen=\dimen@
14746                  \eglssetwidest{\glentryname{\@glo@label}}%
14747              \fi
14748              \settowidth{\dimen@}%
14749                  {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14750              \ifdim\dimen@>#2\relax
14751                  #2=\dimen@
14752              \fi
14753          }%
14754      }%
14755  }

```

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`.

```

14756  \newcommand*{\glstxtrComputeTreeIndent}[1]{%
14757      \glstreeindent=\glstxtrtreetopindent\relax
14758  }

```

uteTreeSubIndent

```
\glxstrComputeTreeSubIndent{<level>}{<label>}{<register>}
```

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
14759 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14760   \ifcsundef{@glswidestname\romannumeral#1}%
14761   {%
14762     \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
14763   }%
14764   {%
14765     \settowidth{#3}{\glstreenamfmt{%
14766       \csname @glswidestname\romannumeral#1\endcsname\space}}%
14767   }%
14768 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14769 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14770 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14771 \renewglossarystyle{alttree}{%
14772   \renewenvironment{theglossary}%
14773   {%
14774     \glxstralttreeInit
14775     \def\@gls@prevlevel{-1}%
14776     \mbox{}\par}%
14777   {\par}%
14778   \renewcommand*{\glossaryheader}{}%
14779   \renewcommand*{\glsgroupheading}[1]{}%
14780   \renewcommand{\glossentry}[2]{%
14781     \ifnum\@gls@prevlevel=0\relax
14782     \else
14783       \glxstrComputeTreeIndent{##1}%
14784     \fi
14785     \parindent\glstreeindent
14786     \glxstrAltTreeSetHangIndent
14787     \makebox[Opt][r]%
14788     {%
14789       \glstreenambox{\glstreeindent}%
14790     }%
14791     \glsentryitem{##1}%
14792     \glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
14793   }%
14794 }
```

```

14795     \glxtralttreeSymbolDescLocation{##1}{##2}%
14796     \def\@gls@prevlevel{0}%
14797 }
14798 \renewcommand{\subglossentry}[3]{%
14799     \ifnum##1=1\relax
14800         \glssubentryitem{##2}%
14801     \fi
14802     \ifnum\@gls@prevlevel=##1\relax
14803     \else
14804         \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14805         \ifnum\@gls@prevlevel<##1\relax
14806             \setlength\glstreeindent\gls@tmplen
14807             \addtolength\glstreeindent\parindent
14808             \parindent\glstreeindent
14809         \else
14810             \ifnum\@gls@prevlevel=0\relax
14811                 \glxtrComputeTreeIndent{##2}%
14812             \else
14813                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14814             \fi
14815             \addtolength\parindent{-\glstreeindent}%
14816             \setlength\glstreeindent\parindent
14817         \fi
14818     \fi
14819     \glxtrAltTreeSetSubHangIndent{##1}%
14820     \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14821         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14822     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14823     \def\@gls@prevlevel{##1}%
14824 }%
14825 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14826 }
14827 }%
14828 {%
14829 }

```

Redefine alttreegroup so that it discourages a break after group headings. Can't use \@afterheading here as it messes with the first item of the group.

```

14830 \ifdef{\@glsstyle@alttreegroup}
14831 {%
14832     \renewglossarystyle{alttreegroup}{%
14833         \setglossarystyle{alttree}%
14834         \renewcommand{\glsgroupheading}[1]{\par
14835             \def\@gls@prevlevel{-1}%
14836             \hangindent0pt\relax
14837             \parindent0pt\relax
14838             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14839         \nopagebreak\indexspace\nopagebreak
14840     }%
14841 }%

```

```

14842 }%
14843 {%
14844 }

```

Similarly for `alttreehypergroup`.

```

14845 \ifdef{\@glsstyle@alttreehypergroup}
14846 {%
14847   \renewglossarystyle{alttreehypergroup}{%
14848     \setglossarystyle{alttree}%
14849     \renewcommand*{\glossaryheader}{%
14850       \par
14851       \def\@gls@prevlevel{-1}%
14852       \hangindent0pt\relax
14853       \parindent0pt\relax
14854       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14855     }%
14856     \renewcommand*{\glsgroupheading}[1]{%
14857       \par
14858       \def\@gls@prevlevel{-1}%
14859       \hangindent0pt\relax
14860       \parindent0pt\relax
14861       \glstreegroupheaderfmt
14862       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14863       \nopagebreak\indexspace\nopagebreak
14864     }%
14865   }
14866 }%
14867 {%
14868 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14869 \ifdef{\@glsstyle@mcolindexgroup}
14870 {%
14871   \renewglossarystyle{mcolindexgroup}{%
14872     \setglossarystyle{mcolindex}%
14873     \renewcommand*{\glsgroupheading}[1]{%
14874       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14875       \nopagebreak\indexspace\nobreak\@afterheading
14876     }%
14877   }
14878 }%
14879 {%
14880 }

```

Similarly for `mcolindexhypergroup`.

```

14881 \ifdef{\@glsstyle@mcolindexhypergroup}
14882 {%

```

```

14883 \renewglossarystyle{mcolindexhypergroup}{%
14884   \setglossarystyle{mcolindex}%
14885   \renewcommand*{\glossaryheader}{%
14886     \item\glstreenavigationfmt{\glsnavigation}%
14887     \indexspace
14888   }%
14889   \renewcommand*{\glsgroupheading}[1]{%
14890     \item\glstreegroupheaderfmt
14891       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14892     \nopagebreak\indexspace\nobreak\@afterheading
14893   }%
14894 }
14895 }%
14896 {%
14897 }

```

Similarly for mcolindexspannav.

```

14898 \ifdef{\@glsstyle@mcolindexspannav}
14899 {%
14900   \renewglossarystyle{mcolindexspannav}{%
14901     \setglossarystyle{index}%
14902     \renewenvironment{theglossary}%
14903     {%
14904       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14905       \setlength{\parindent}{0pt}%
14906       \setlength{\parskip}{0pt plus 0.3pt}%
14907       \let\item\glstreeitem}%
14908     {\end{multicols}}}%
14909   \renewcommand*{\glsgroupheading}[1]{%
14910     \item\glstreegroupheaderfmt
14911       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14912     \nopagebreak\indexspace\nobreak\@afterheading
14913   }%
14914 }
14915 }%
14916 {%
14917 }

```

Similarly for mcoltreegroup.

```

14918 \ifdef{\@glsstyle@mcoltreegroup}
14919 {%
14920   \renewglossarystyle{mcoltreegroup}{%
14921     \setglossarystyle{mcoltree}%
14922     \renewcommand{\glsgroupheading}[1]{\par
14923       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14924     \nopagebreak\indexspace\nobreak\@afterheading
14925   }%
14926 }
14927 }%
14928 {%

```

14929 }

Similarly for mcoltreehypergroup.

```
14930 \ifdef{\@glsstyle@mcoltreehypergroup}
14931 {%
14932   \renewglossarystyle{mcoltreehypergroup}{%
14933     \setglossarystyle{mcoltree}%
14934     \renewcommand*\glossaryheader{%
14935       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14936     }%
14937     \renewcommand*\glsgroupheading}[1]{%
14938       \par\noindent
14939       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14940       \nopagebreak\indexspace\nobreak\@afterheading
14941     }%
14942   }
14943 }%
14944 {%
14945 }
```

Similarly for mcoltreespannav.

```
14946 \ifdef{\@glsstyle@mcoltreespannav}
14947 {%
14948   \renewglossarystyle{mcoltreespannav}{%
14949     \setglossarystyle{tree}%
14950     \renewenvironment{theglossary}%
14951     {%
14952       \begin{multicols}{\glsncols}%
14953       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14954       \setlength{\parindent}{0pt}%
14955       \setlength{\parskip}{0pt plus 0.3pt}%
14956     }%
14957     {\end{multicols}}%
14958     \renewcommand*\glsgroupheading}[1]{%
14959       \par\noindent
14960       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14961       \nopagebreak\indexspace\nobreak\@afterheading
14962     }%
14963   }
14964 }%
14965 {%
14966 }
```

Similarly for mcoltreenonamegroup.

```
14967 \ifdef{\@glsstyle@mcoltreenonamegroup}
14968 {%
14969   \renewglossarystyle{mcoltreenonamegroup}{%
14970     \setglossarystyle{mcoltreenoname}%
14971     \renewcommand*\glsgroupheading}[1]{\par
14972       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14973     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14974 }%
14975 }
14976 }%
14977 {%
14978 }

```

Similarly for mcoltreenonamehypergroup.

```

14979 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14980 {%
14981   \renewglossarystyle{mcoltreenonamehypergroup}{%
14982     \setglossarystyle{mcoltreenoname}%
14983     \renewcommand*{\glossaryheader}{%
14984       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14985     \renewcommand*{\glsgroupheading}[1]{%
14986       \par\noindent
14987       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14988       \nopagebreak\indexspace\nobreak\@afterheading}%
14989   }
14990 }%
14991 {%
14992 }

```

Similarly for mcoltreenonamespannav.

```

14993 \ifdef{\@glsstyle@mcoltreenonamespannav}
14994 {%
14995   \renewglossarystyle{mcoltreenonamespannav}{%
14996     \setglossarystyle{treenoname}%
14997     \renewenvironment{theglossary}%
14998     {%
14999       \begin{multicols}{\glsmcols}%
15000       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
15001       \setlength{\parindent}{0pt}%
15002       \setlength{\parskip}{0pt plus 0.3pt}%
15003     }%
15004     {\end{multicols}}}%
15005   \renewcommand*{\glsgroupheading}[1]{%
15006     \par\noindent
15007     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
15008     \nopagebreak\indexspace\nobreak\@afterheading}%
15009   }
15010 }%
15011 {%
15012 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

15013 \ifdef{\@glsstyle@mcolalttree}
15014 {%
15015   \renewglossarystyle{mcolalttree}{%
15016     \setglossarystyle{alttree}%
15017     \renewenvironment{theglossary}%

```

```

15018   {%
15019       \glstralttreeInit
15020       \def\@gls@prevlevel{-1}%
15021       \begin{multicols}{\glsmcols}%
15022   }%
15023   {\par\end{multicols}}%
15024 }
15025 }%
15026 {%
15027 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

15028 \ifdef{\@glsstyle@mcolalmtreegroup}
15029 {%
15030   \renewglossarystyle{mcolalmtreegroup}{%
15031     \setglossarystyle{mcolalmtree}%
15032     \renewcommand{\glsgroupheading}[1]{\par
15033       \def\@gls@prevlevel{-1}%
15034       \hangindent0pt\relax
15035       \parindent0pt\relax
15036       \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
15037       \nopagebreak\indexspace\nopagebreak
15038   }%
15039 }
15040 }%
15041 {%
15042 }

```

Similarly for mcolalmtreehypergroup.

```

15043 \ifdef{\@glsstyle@mcolalmtreehypergroup}
15044 {%
15045   \renewglossarystyle{mcolalmtreehypergroup}{%
15046     \setglossarystyle{mcolalmtree}%
15047     \renewcommand*\{\glossaryheader}{%
15048       \par
15049       \def\@gls@prevlevel{-1}%
15050       \hangindent0pt\relax
15051       \parindent0pt\relax
15052       \glstreenavigationfmt{\glsnavigation}%
15053       \par\indexspace
15054   }%
15055   \renewcommand*\{\glsgroupheading}[1]{%
15056     \par
15057     \def\@gls@prevlevel{-1}%
15058     \hangindent0pt\relax
15059     \parindent0pt\relax
15060     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
15061     \nopagebreak\indexspace\nopagebreak
15062   }%
15063 }

```

```

15064 }%
15065 {%
15066 }

```

Similarly for mcolalttreespannav.

```

15067 \ifdef{\@glsstyle@mcolalttreespannav}
15068 {%
15069   \renewglossarystyle{mcolalttreespannav}{%
15070     \setglossarystyle{alttree}%
15071     \renewenvironment{theglossary}%
15072     {%
15073       \glsextralttreeInit
15074       \def\@gls@prevlevel{-1}%
15075       \begin{multicols}{\@gls@col}%
15076         [\noindent\glstreenavigationfmt{\@gls@navigation}]%
15077     }%
15078     {\par\end{multicols}}%
15079     \renewcommand*{\@gls@groupheading}[1]{%
15080       \par
15081       \def\@gls@prevlevel{-1}%
15082       \hangindent0pt\relax
15083       \parindent0pt\relax
15084       \glstreegroupheaderfmt{\@gls@navhypertarget{##1}{\@gls@getgrouptitle{##1}}}%
15085       \nopagebreak\indexspace\nopagebreak
15086     }%
15087   }
15088 }%
15089 {%
15090 }

```

Reset the default style

```

15091 \ifx\@glossary@default@style\relax
15092 \else
15093   \setglossarystyle{\@gls@current@style}
15094 \fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
15095 \NeedsTeXFormat{LaTeX2e}
15096 \ProvidesPackage{glossary-bookindex}[2019/03/22 1.39 (NLCT)]
```

Load required packages.

```
15097 \RequirePackage{multicol}
15098 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
15099 \newcommand{\glxstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
15100 \newcommand*{\glxstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
15101 \newcommand*{\glxstrbookindexsubname}[1]{\glxstrbookindexname{#1}}
```

`glxstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
15102 \providecommand*{\glxstrprelocation}{\space}
```

`indexprelocation` Separator used before location list for top-level entries. Version 1.22 has removed the `\ifglsnopostdot` check since this style doesn't display the description.

```
15103 \newcommand*{\glxstrbookindexprelocation}[1]{%
15104   \glxstrifhasfield{location}{#1}%
15105   {,\glxstrprelocation}%
15106   {\glxstrprelocation}%
15107 }
```

`glxstrsubprelocation` Separator used before location list for sub-entries.

```
15108 \newcommand*{\glxstrbookindexsubprelocation}[1]{%
15109   \glxstrbookindexprelocation{#1}%
15110 }
```

`bookindexlocation` `\glxstrbookindexlocation{<label>}{<location>}`

Displays the location.

```
15111 \newcommand*{\glxstrbookindexlocation}[2]{#2}
```

`indexsublocation` `\glxtrbookindexlocation{<label>}{<location>}`

Displays the location for sub-entries.

```
15112 \newcommand*{\glxtrbookindexsublocation}{\glxtrbookindexlocation}
```

`xparentchildsep` Separator used between top-level parent and child entry.

```
15113 \newcommand{\glxtrbookindexparentchildsep}{\nopagebreak}
```

`rentsubchildsep` Separator used between sub-level parent and child entry.

```
15114 \newcommand{\glxtrbookindexparentsubchildsep}{\glxtrbookindexparentchildsep}
```

`ookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
15115 \newcommand{\glxtrbookindexbetween}[2]{}{}
```

`indexsubbetween` Between two level 1 entries identified by the labels in the arguments.

```
15116 \newcommand{\glxtrbookindexsubbetween}[2]{}{}
```

`exsubsubbetween` Between two level 2 entries identified by the labels in the arguments.

```
15117 \newcommand{\glxtrbookindexsubsubbetween}[2]{}{}
```

`indexatendgroup` At the end of a letter group. The argument is the index of the last top-level entry.

```
15118 \newcommand{\glxtrbookindexatendgroup}[1]{}{}
```

`exsubatendgroup` At the end of a letter group. The argument is the index of the last level 1 entry.

```
15119 \newcommand{\glxtrbookindexsubatendgroup}[1]{}{}
```

`ubsubatendgroup` At the end of a letter group. The argument is the index of the last level 2 entry.

```
15120 \newcommand{\glxtrbookindexsubsubatendgroup}[1]{}{}
```

`kindexgroupskip` Group separator.

```
15121 \newcommand{\glxtrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

`dexformatheader` Group separator.

```
15122 \newcommand*{\glxtrbookindexformatheader}[1]{%
15123   \par{\centering\glstreegroupheaderfmt{#1}\par}%
15124 }
```

`okindexbookmark` Book mark group heading if supported.

```
15125 \ifdef\pdfbookmark
15126 {%
15127   \newcommand*{\glxtrbookindexbookmark}[2]{%
15128     \ifdefstring{@@glossarysec}{chapter}%
15129     {\pdfbookmark[1]{#1}{#2}}%
15130     {\pdfbookmark[2]{#1}{#2}}%
15131   }
15132 }
```

```

15131 }
15132 }
15133 {%
15134 \newcommand*{\glxtrbookindexbookmark}[2]{%
15135 }

```

kindexcolspread

```

15136 \newcommand*{\glxtrbookindexcolspread}{%

```

dexmulticolseenv

```

15137 \newcommand*{\glxtrbookindexmulticolseenv}{multicols}

```

Define the style.

```

15138 \newglossarystyle{bookindex}{%
15139 \setglossarystyle{index}%
15140 \renewenvironment{theglossary}%
15141 {%
15142 \ifnum\glxtrbookindexcols>1\relax
15143 \ifdefempty\glxtrbookindexcolspread
15144 {%
15145 \edef\glxtr@beginbookindex{%
15146 \noexpand\begin{\glxtrbookindexmulticolseenv}
15147 {\glxtrbookindexcols}%
15148 }%
15149 }%
15150 {%
15151 \edef\glxtr@beginbookindex{%
15152 \noexpand\begin{\glxtrbookindexmulticolseenv}%
15153 {\glxtrbookindexcols}[\glxtrbookindexcolspread]%
15154 }%
15155 }%
15156 \else
15157 \def\glxtr@beginbookindex{%
15158 \fi
15159 \glxtr@beginbookindex
15160 \setlength{\parindent}{0pt}%
15161 \setlength{\parskip}{0pt plus 0.3pt}%
15162 \let\@glxtr@bookindex@sep\glxtrbookindexparentchildsep
15163 \let\@glxtr@bookindex@subsep\glxtrbookindexparentschildsep
15164 \let\@glxtr@bookindex@between\@gobble
15165 \let\@glxtr@bookindex@subbetween\@gobble
15166 \let\@glxtr@bookindex@subsubbetween\@gobble
15167 \let\@glxtr@bookindex@atendgroup\relax
15168 \let\@glxtr@bookindex@subatendgroup\relax
15169 \let\@glxtr@bookindex@subsubatendgroup\relax
15170 \let\@glxtr@bookindexgroupskip\relax
15171 }%
15172 {%

```

Do end group hooks.

```
15173 \@glxtr@bookindex@subsubatendgroup
15174 \@glxtr@bookindex@subatendgroup
15175 \@glxtr@bookindex@atendgroup
```

End multicols environment.

```
15176 \ifnum\glxtrbookindexcols>1\relax
15177 \edef\glxtr@endbookindex{%
15178 \noexpand\end{\glxtrbookindexmulticolscenv}%
15179 }%
15180 \else
15181 \def\glxtr@endbookindex{%
15182 \fi
15183 \glxtr@endbookindex
15184 }%
```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```
15185 \renewcommand*{\glossaryheader}{\raggedright}%
```

Top level entry format.

```
15186 \renewcommand*{\glossentry}[2]{%
```

Do separator.

```
15187 \@glxtr@bookindex@between{##1}%
```

Update separators.

```
15188 \let\@glxtr@bookindex@sep\glxtrbookindexparentchildsep
15189 \let\@glxtr@bookindex@subsep\glxtrbookindexparentsubchildsep
15190 \let\@glxtr@bookindex@subbetween\@gobble
15191 \let\@glxtr@bookindex@subsubbetween\@gobble
15192 \edef\@glxtr@bookindex@between{%
15193 \noexpand\glxtrbookindexbetween{##1}%
15194 }%
15195 \edef\@glxtr@bookindex@atendgroup{%
15196 \noexpand\glxtrbookindexatendgroup{##1}%
15197 }%
15198 \let\@glxtr@bookindex@subatendgroup\relax
15199 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Format entry.

```
15200 \glstreeitem
15201 \glstryitem{##1}%
15202 \glstarget{##1}{\glxtrbookindexname{##1}}%
15203 \glxtrbookindexprelocation{##1}%
15204 \glxtrbookindexlocation{##1}{##2}%
15205 }%
15206 \renewcommand{\subglossentry}[3]{%
15207 \ifcase##1\relax
```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```
15208 \glstreeitem
15209 \or
```

Level 1.

```
15210 \@glxtr@bookindex@sep
15211 \@glxtr@bookindex@subbetween{##2}%
15212 \let\@glxtr@bookindex@sep\relax
```

Update separators.

```
15213 \let\@glxtr@bookindex@subsubbetween\@gobble
15214 \let\@glxtr@bookindex@subsep\glxtrbookindexparentsubchildsep
15215 \edef\@glxtr@bookindex@subbetween{%
15216 \noexpand\glxtrbookindexsubbetween{##2}%
15217 }%
15218 \edef\@glxtr@bookindex@atsubendgroup{%
15219 \noexpand\glxtrbookindexatsubendgroup{##1}%
15220 }%
```

Start sub-item.

```
15221 \glstreesubitem
15222 \glssubentryitem{##2}%
15223 \else
```

All other levels.

```
15224 \@glxtr@bookindex@subsep
15225 \@glxtr@bookindex@subsubbetween{##2}%
```

Update separators.

```
15226 \let\@glxtr@bookindex@subsep\relax
15227 \edef\@glxtr@bookindex@subsubbetween{%
15228 \noexpand\glxtrbookindexsubsubbetween{##2}%
15229 }%
15230 \edef\@glxtr@bookindex@atsubsubendgroup{%
15231 \noexpand\glxtrbookindexatsubsubendgroup{##1}%
15232 }%
```

Start sub-sub-item.

```
15233 \glstreesubsubitem
15234 \fi
```

Format entry.

```
15235 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
15236 \glxtrbookindexsubprelocation{##2}%
15237 \glxtrbookindexsublocation{##2}{##3}%
15238 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
15239 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
15240 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
15241 \@glxtr@bookindex@subsubatendgroup
15242 \@glxtr@bookindex@subatendgroup
```

```

15243 \glstr@bookindex@atendgroup
15244 \glstr@bookindexgroupskip

```

Update separators.

```

15245 \let\glstr@bookindexgroupskip\glstrbookindexgroupskip
15246 \let\glstr@bookindex@between\@gobble
15247 \let\glstr@bookindex@atendgroup\relax
15248 \let\glstr@bookindex@subatendgroup\relax
15249 \let\glstr@bookindex@subsubatendgroup\relax

```

Fetch the group title from the label supplied in #1.

```

15250 \glstrgetgrouptitle{##1}{\thisgrptitle}%

```

Do the PDF bookmark if supported.

```

15251 \glstrbookindexbookmark{\thisgrptitle}{index.##1}%

```

Format the group title.

```

15252 \glstrbookindexformatheader{\thisgrptitle}%
15253 \nopagebreak\indexspace\nopagebreak\@afterheading
15254 }%
15255 }

```

Some supplementary commands that may be useful. These store the entry label for the current page. Since the page number is needed in the control sequence, this uses `\glstrbookindexthepage` instead of `\thepage` in case the page numbering has been set to something that contains formatting commands.

`bookindexthepage` The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```

15256 \newcommand{\glstrbookindexthepage}{%
15257 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
15258 }

```

`bookindexmarkentry` Writes entry information to the .aux file. The argument is the entry label.

```

15259 \newcommand*{\glstrbookindexmarkentry}[1]{%
15260 \protected@write\@auxout
15261 {\let\glstrbookindexthepage\relax}%
15262 {\string\glstr@setbookindexmark{\glstrbookindexthepage}{#1}}%
15263 }

```

`setbookindexmark`

```

15264 \newcommand*{\glstr@setbookindexmark}[2]{%
15265 \ifcsundef\glstr@idxfirstmark@#1%
15266 {\csgdef\glstr@idxfirstmark@#1}{#2}}%
15267 {}%
15268 \csgdef\glstr@idxlastmark@#1}{#2}%
15269 }

```

`indexfirstmarkfmt`

```

15270 \newcommand*{\glstrbookindexfirstmarkfmt}[1]{%
15271 \glstryname{#1}%
15272 }

```

kindexfirstmark

```
15273 \newcommand*{\glxtrbookindexfirstmark}{%  
15274   \letcs{\glxtr@label}{\glxtr@idxfirstmark@\glxtrbookindexthepage}%  
15275   \ifdef\glxtr@label  
15276     {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%  
15277     {}%  
15278 }
```

indexlastmarkfmt

```
15279 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%  
15280   \glstryname{#1}%  
15281 }
```

okindexlastmark

```
15282 \newcommand*{\glxtrbookindexlastmark}{%  
15283   \letcs{\glxtr@label}{\glxtr@idxlastmark@\glxtrbookindexthepage}%  
15284   \ifdef\glxtr@label  
15285     {\glxtrbookindexlastmarkfmt{\glxtr@label}}%  
15286     {}%  
15287 }
```

4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
15288 \NeedsTeXFormat{LaTeX2e}
15289 \ProvidesPackage{glossary-longextra}[2019/03/22 1.39 (NLCT)]

Load required packages.
15290 \RequirePackage{glossary-longbooktabs}
```

longextraNameFmt `\glslongextraNameFmt{<label>}`

Governs the way the name is displayed.

```
15291 \newcommand{\glslongextraNameFmt}[1]{%
15292   \glsentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%
15293 }
```

longextraDescFmt `\glslongextraDescFmt{<label>}`

Governs the way the description is displayed.

```
15294 \newcommand{\glslongextraDescFmt}[1]{%
15295   \glossentrydesc{#1}\glspostdescription
15296 }
```

longextraSymbolFmt `\glslongextraSymbolFmt{<label>}`

Governs the way the symbol is displayed.

```
15297 \newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

longextraLocationFmt `\glslongextraLocationFmt{<label>}{<location list>}`

Governs the way the location is displayed.

```
15298 \newcommand{\glslongextraLocationFmt}[2]{#2}
```

gextraSubNameFmt

```
\glslongextraSubNameFmt{<level>}{<label>}
```

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
15299 \newcommand{\glslongextraSubNameFmt}[2]{%
15300 \glssubentryitem{#2}\glstarget{#2}{\strut}%
15301 }
```

gextraSubDescFmt

```
\glslongextraSubDescFmt{<level>}{<label>}
```

Governs the way the child description is displayed.

```
15302 \newcommand{\glslongextraSubDescFmt}[2]{%
15303 \glslongextraDescFmt{#2}%
15304 }
```

extraSubSymbolFmt

```
\glslongextraSubSymbolFmt{<level>}{<label>}
```

Governs the way the child symbol is displayed.

```
15305 \newcommand{\glslongextraSubSymbolFmt}[2]{%
15306 \glslongextraSymbolFmt{#2}%
15307 }
```

raSubLocationFmt

```
\glslongextraSubLocationFmt{<level>}{<label>}{<location list>}
```

Governs the way the child location list is displayed.

```
15308 \newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

gextraNameAlign Alignment for the name column.

```
15309 \newcommand{\glslongextraNameAlign}{l}
```

gextraDescAlign Alignment for the description column.

```
15310 \newcommand{\glslongextraDescAlign}{>{\raggedright}p{\glsdescwidth}}
```

extraSymbolAlign Alignment for the symbol column.

```
15311 \newcommand{\glslongextraSymbolAlign}{c}
```

raLocationAlign Alignment for the location column.

```
15312 \newcommand{\glslongextraLocationAlign}{>\raggedright}p{\glspagelistwidth}}
```

traGroupHeading Used to format the letter group headings. The first argument is the number of columns in the table. The second is the group *label* (not the title).

```
15313 \newcommand{\glslongextraGroupHeading}[2]{}
```

traHeaderFormat Format for the column headers.

```
15314 \newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}
```

aNameDescHeader

```
15315 \newcommand{\glslongextraNameDescHeader}{%
15316 \glslongextraNameDescTabularHeader\endhead
15317 \glslongextraNameDescTabularFooter\endfoot
15318 }
```

scTabularHeader

```
15319 \newcommand{\glslongextraNameDescTabularHeader}{%
15320 \toprule
15321 \glslongextraHeaderFmt\entryname &
15322 \glslongextraHeaderFmt\descriptionname\tabularnewline
15323 \midrule
15324 }
```

scTabularFooter

```
15325 \newcommand{\glslongextraNameDescTabularFooter}{%
15326 \bottomrule
15327 }
```

Unlike the alttree style, there aren't different widths for the hierarchical levels.

gextraSetWidest Provide in case the tree styles haven't been loaded.

```
15328 \newcommand*{\glslongextraSetWidest}[1]{%
15329 \def\@glslongextrawidestname{#1}%
15330 }
```

extrawidestname Pick up the widest name from the alttree style if it has been set. (Will expand to nothing otherwise.)

```
15331 \newcommand*{\@glslongextrawidestname}{\csuse{\glswidestname}}
```

traUpdateWidest

```
15332 \newcommand*{\glslongextraUpdateWidest}[1]{%
15333 \ifundef\@glslongextrawidestname
15334 {\def\@glslongextrawidestname{#1}}%
15335 {%
15336 \settowidth{\dimen@}{\@glslongextrawidestname}%
15337 \settowidth{\dimen@ii}{#1}%
15338 \ifdim\dimen@ii>\dimen@
```

```

15339      \def\@glslongextrawidestname{#1}%
15340      \fi
15341    }%
15342 }

```

updateWidestChild

```
\glslongextraUpdateWidestChild{<level>}{<text>}
```

Used by `\glsxtrSetWidest` in `glossaries-extra-bib2gls`. Does nothing by default, since the default action in these styles is to omit the child name. If the child name should be displayed, then this needs to be redefined to use `\glslongextraUpdateWidest`.

```
15343 \newcommand*{\glslongextraUpdateWidestChild}[2]{}
```

traSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```

15344 \newcommand{\glslongextraSetDescWidth}{%
15345   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%

   Has the widest name been set.
15346   \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
15347   \ifdim\dimen@>\gls@tmplen
15348     \gls@tmplen=\dimen@
15349   \fi

```

Description width is `\linewidth` less `4\tabcolsep` less the width of the name column.

```

15350   \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
15351 }

```

SymSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```
15352 \newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15353   \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```
15354   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```

Subtract `2\tabcolsep` and the symbol header width.

```

15355   \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
15356 }

```

LocSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name, location and description columns.

```
15357 \newcommand{\glslongextraLocSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15358   \glslongextraSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15359 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15360 }
```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that have name, symbol, location and description columns.

```
15361 \newcommand{\glslongextraSymLocSetDescWidth}{%
```

Work out the size for just the name, symbol and description style.

```
15362 \glslongextraSymSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15363 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15364 }
```

ExtraUseTabular If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```
15365 \newif\ifGlsLongExtraUseTabular
15366 \GlsLongExtraUseTabularfalse
```

raTabularVAlign Only used with the tabular setting.

```
15367 \newcommand*{\glslongextraTabularVAlign}{c}
```

long-name-desc Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```
15368 \newglossarystyle{long-name-desc}%
15369 {%
15370   \ifGlsLongExtraUseTabular
15371     \renewenvironment{theglossary}%
15372     {%
15373       \glslongextraSetDescWidth
15374       \edef\@glslongextra@begintab{%
15375         \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
15376         \expandonce\glslongextraNameAlign
15377         \expandonce\glslongextraDescAlign}}%
15378       \@glslongextra@begintab
15379     }%
15380     {%
15381       \glslongextraNameDescTabularFooter
15382       \end{tabular}%
15383     }%
15384     \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
15385   \else
15386     \renewenvironment{theglossary}%
15387     {%
15388       \glspatchLToutput
15389       \glslongextraSetDescWidth
15390       \edef\@glslongextra@begintab{%
15391         \noexpand\begin{longtable}{%
```

```

15392         \expandonce\glslongextraNameAlign
15393         \expandonce\glslongextraDescAlign}}}%
15394     \@glslongextra@begintab
15395 }%
15396 {\end{longtable}}}%
15397 \renewcommand*{\glossaryheader}{\glslongextraNameDescHeader}%
15398 \fi
15399 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
15400 \renewcommand{\glossentry}[2]{%
15401     \glslongextraNameFmt{##1} &
15402     \glslongextraDescFmt{##1}\tabularnewline
15403 }%
15404 \renewcommand{\subglossentry}[3]{%
15405     \glslongextraSubNameFmt{##1}{##2}
15406     &
15407     \glslongextraSubDescFmt{##1}{##2}%
15408     \tabularnewline
15409 }%
15410 \ifglsnogroupskip
15411     \renewcommand*{\glsgroupskip}{}%
15412 \else
15413     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15414 \fi
15415 }

```

cLocationHeader

```

15416 \newcommand{\glslongextraNameDescLocationHeader}{%
15417   \glslongextraNameDescLocationTabularHeader\endhead
15418   \glslongextraNameDescLocationTabularFooter\endfoot
15419 }

```

onTabularHeader

```

15420 \newcommand{\glslongextraNameDescLocationTabularHeader}{%
15421   \toprule
15422   \glslongextraHeaderFmt\entryname &
15423   \glslongextraHeaderFmt\descriptionname &
15424   \glslongextraHeaderFmt\pagelistname\tabularnewline
15425   \midrule
15426 }

```

onTabularFooter

```

15427 \newcommand{\glslongextraNameDescLocationTabularFooter}{%
15428   \bottomrule
15429 }

```

g-name-desc-loc Three columns: name, description and location list.

```

15430 \newglossarystyle{long-name-desc-loc}%
15431 {%
15432   \ifGlsLongExtraUseTabular

```

```

15433 \renewenvironment{theglossary}%
15434 {%
15435     \glslongextraLocSetDescWidth
15436     \edef\@glslongextra@begintab{%
15437         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15438             \expandonce\glslongextraNameAlign
15439             \expandonce\glslongextraDescAlign
15440             \expandonce\glslongextraLocationAlign
15441         }}%
15442     \@glslongextra@begintab
15443 }%
15444 {%
15445     \glslongextraNameDescLocationTabularFooter
15446     \end{tabular}%
15447 }%
15448 \renewcommand*\glossaryheader{\glslongextraNameDescLocationTabularHeader}%
15449 \else
15450 \renewenvironment{theglossary}%
15451 {%
15452     \glspatchLToutput
15453     \glslongextraLocSetDescWidth
15454     \edef\@glslongextra@begintab{%
15455         \noexpand\begin{longtable}{%
15456             \expandonce\glslongextraNameAlign
15457             \expandonce\glslongextraDescAlign
15458             \expandonce\glslongextraLocationAlign
15459         }}%
15460     \@glslongextra@begintab
15461 }%
15462 {\end{longtable}}%
15463 \renewcommand*\glossaryheader{\glslongextraNameDescLocationHeader}%
15464 \fi
15465 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15466 \renewcommand{\glossentry}[2]{%
15467     \glslongextraNameFmt{##1} &
15468     \glslongextraDescFmt{##1} &
15469     \glslongextraLocationFmt{##1}{##2}\tabularnewline
15470 }%
15471 \renewcommand{\subglossentry}[3]{%
15472     \glslongextraSubNameFmt{##1}{##2}&
15473     \glslongextraSubDescFmt{##1}{##2}&
15474     \glslongextraSubLocationFmt{##1}{##2}{##3}%
15475     \tabularnewline
15476 }%
15477 \ifglsgroupskip
15478     \renewcommand*\glsgroupskip{}%
15479 \else
15480     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15481 \fi

```

15482 }

aDescNameHeader

```
15483 \newcommand{\glslongextraDescNameHeader}{%
15484   \glslongextraDescNameTabularHeader\endhead
15485   \glslongextraDescNameTabularFooter\endfoot
15486 }
```

meTabularHeader

```
15487 \newcommand{\glslongextraDescNameTabularHeader}{%
15488   \toprule
15489   \glslongextraHeaderFmt\descriptionname&
15490   \glslongextraHeaderFmt\entryname \tabularnewline
15491   \midrule
15492 }
```

meTabularFooter

```
15493 \newcommand{\glslongextraDescNameTabularFooter}{%
15494   \bottomrule
15495 }
```

long-desc-name Like name-desc but swaps the columns.

```
15496 \newglossarystyle{long-desc-name}%
15497 {%
15498   \ifGlsLongExtraUseTabular
15499     \renewenvironment{theglossary}%
15500     {%
15501       \glslongextraSetDescWidth
15502       \edef\@glslongextra@begintab{%
15503         \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
15504         \expandonce\glslongextraDescAlign
15505         \expandonce\glslongextraNameAlign}}%
15506       \@glslongextra@begintab
15507     }%
15508     {%
15509       \glslongextraDescNameTabularFooter
15510       \end{tabular}%
15511     }%
15512     \renewcommand*\glossaryheader{\glslongextraDescNameTabularHeader}%
15513   \else
15514     \renewenvironment{theglossary}%
15515     {%
15516       \glspatchLToutput
15517       \glslongextraSetDescWidth
15518       \edef\@glslongextra@begintab{%
15519         \noexpand\begin{longtable}{%
15520         \expandonce\glslongextraDescAlign
15521         \expandonce\glslongextraNameAlign}}%
15522       \@glslongextra@begintab
```

```

15523 }%
15524 {\end{longtable}}%
15525 \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
15526 \fi
15527 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
15528 \renewcommand{\glossentry}[2]{%
15529   \glslongextraDescFmt{##1} &
15530   \glslongextraNameFmt{##1}\tabularnewline
15531 }%
15532 \renewcommand{\subglossentry}[3]{%
15533   \glslongextraSubDescFmt{##1}{##2} &
15534   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15535 }%
15536 \ifglsgroupskip
15537   \renewcommand*{\glsgroupskip}{}%
15538 \else
15539   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15540 \fi
15541 }

```

DescNameHeader

```

15542 \newcommand{\glslongextraLocationDescNameHeader}{%
15543   \glslongextraLocationDescNameTabularHeader\endhead
15544   \glslongextraLocationDescNameTabularFooter\endfoot
15545 }

```

meTabularHeader

```

15546 \newcommand{\glslongextraLocationDescNameTabularHeader}{%
15547   \toprule
15548   \glslongextraHeaderFmt\pagelistname&
15549   \glslongextraHeaderFmt\descriptionname&
15550   \glslongextraHeaderFmt\entryname \tabularnewline
15551   \midrule
15552 }

```

meTabularFooter

```

15553 \newcommand{\glslongextraLocationDescNameTabularFooter}{%
15554   \bottomrule
15555 }

```

g-loc-desc-name Three columns: location, description and name.

```

15556 \newglossarystyle{long-loc-desc-name}%
15557 {%
15558   \ifGlsLongExtraUseTabular
15559   {%
15560     \glslongextraLocSetDescWidth
15561     \edef\@glslongextra@begintab{%
15562       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15563         \expandonce\glslongextraLocationAlign

```

```

15564         \expandonce\glslongextraDescAlign
15565         \expandonce\glslongextraNameAlign}}}%
15566     \@glslongextra@begintab
15567 }%
15568 {%
15569     \glslongextraLocationDescNameTabularFooter
15570     \end{tabular}}%
15571 }%
15572 \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameTabularHeader}}%
15573 \else
15574 \renewenvironment{theglossary}%
15575 {%
15576     \glspatchLToutput
15577     \glslongextraLocSetDescWidth
15578     \edef\@glslongextra@begintab{%
15579         \noexpand\begin{longtable}{%
15580             \expandonce\glslongextraLocationAlign
15581             \expandonce\glslongextraDescAlign
15582             \expandonce\glslongextraNameAlign}}}%
15583     \@glslongextra@begintab
15584 }%
15585 {\end{longtable}}}%
15586 \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameHeader}}%
15587 \fi
15588 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15589 \renewcommand{\glossentry}[2]{%
15590     \glslongextraLocationFmt{##1}{##2} &
15591     \glslongextraDescFmt{##1} &
15592     \glslongextraNameFmt{##1}\tabularnewline
15593 }%
15594 \renewcommand{\subglossentry}[3]{%
15595     \glslongextraSubLocationFmt{##1}{##2}{##3} &
15596     \glslongextraSubDescFmt{##1}{##2} &
15597     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15598 }%
15599 \ifglsgroupskip
15600     \renewcommand*{\glsgroupskip}{}%
15601 \else
15602     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15603 \fi
15604 }

```

meDescSymHeader

```

15605 \newcommand{\glslongextraNameDescSymHeader}{%
15606     \glslongextraNameDescSymTabularHeader\endhead
15607     \glslongextraNameDescSymTabularFooter\endfoot
15608 }

```

ymTabularHeader

```

15609 \newcommand{\glslongextraNameDescSymTabularHeader}{%
15610 \toprule
15611 \glslongextraHeaderFmt\entryname &
15612 \glslongextraHeaderFmt\descriptionname &
15613 \glslongextraHeaderFmt\symbolname\tabularnewline
15614 \midrule
15615 }

```

ymTabularFooter

```

15616 \newcommand{\glslongextraNameDescSymTabularFooter}{%
15617 \bottomrule
15618 }

```

g-name-desc-sym Three column style with symbol in the third column.

```

15619 \newglossarystyle{long-name-desc-sym}%
15620 {%
15621 \ifGlsLongExtraUseTabular
15622 \renewenvironment{theglossary}%
15623 {%
15624 \glslongextraSymSetDescWidth
15625 \edef\@glslongextra@begintab{%
15626 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15627 \expandonce\glslongextraNameAlign
15628 \expandonce\glslongextraDescAlign
15629 \expandonce\glslongextraSymbolAlign
15630 }}%
15631 \@glslongextra@begintab
15632 }%
15633 {%
15634 \glslongextraNameDescSymTabularFooter
15635 \end{tabular}%
15636 }%
15637 \renewcommand*\glossaryheader{\glslongextraNameDescSymTabularHeader}%
15638 \else
15639 \renewenvironment{theglossary}%
15640 {%
15641 \glspatchLToutput
15642 \glslongextraSymSetDescWidth
15643 \edef\@glslongextra@begintab{%
15644 \noexpand\begin{longtable}{%
15645 \expandonce\glslongextraNameAlign
15646 \expandonce\glslongextraDescAlign
15647 \expandonce\glslongextraSymbolAlign
15648 }}%
15649 \@glslongextra@begintab
15650 }%
15651 {\end{longtable}}%
15652 \renewcommand*\glossaryheader{\glslongextraNameDescSymHeader}%
15653 \fi

```

```

15654 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15655 \renewcommand{\glossentry}[2]{%
15656   \glslongextraNameFmt{##1} &
15657   \glslongextraDescFmt{##1} &
15658   \glslongextraSymbolFmt{##1}\tabularnewline
15659 }%
15660 \renewcommand{\subglossentry}[3]{%
15661   \glslongextraSubNameFmt{##1}{##2} &
15662   \glslongextraSubDescFmt{##1}{##2} &
15663   \glslongextraSubSymbolFmt{##1}{##2}%
15664   \tabularnewline
15665 }%
15666 \ifglsgnogroupskip
15667   \renewcommand*{\glsgroupskip}{}%
15668 \else
15669   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15670 \fi
15671 }

```

onLocationHeader

```

15672 \newcommand{\glslongextraNameDescSymLocationHeader}{%
15673   \glslongextraNameDescSymLocationTabularHeader\endhead
15674   \glslongextraNameDescSymLocationTabularFooter\endfoot
15675 }

```

onTabularHeader

```

15676 \newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
15677   \toprule
15678   \glslongextraHeaderFmt\entryname &
15679   \glslongextraHeaderFmt\descriptionname &
15680   \glslongextraHeaderFmt\symbolname &
15681   \glslongextraHeaderFmt\pagelistname\tabularnewline
15682   \midrule
15683 }

```

onTabularFooter

```

15684 \newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
15685   \bottomrule
15686 }

```

me-desc-sym-loc Four columns: name, description and location

```

15687 \newglossarystyle{long-name-desc-sym-loc}%
15688 {%
15689   \ifGlsLongExtraUseTabular
15690     \renewenvironment{theglossary}%
15691     {%
15692       \glslongextraSymLocSetDescWidth
15693       \edef\@glslongextra@begintab{%
15694         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%

```

```

15695         \expandonce\glslongextraNameAlign
15696         \expandonce\glslongextraDescAlign
15697         \expandonce\glslongextraSymbolAlign
15698         \expandonce\glslongextraLocationAlign
15699     } }%
15700     \@glslongextra@begintab
15701 }%
15702 {%
15703     \glslongextraNameDescSymLocationTabularFooter
15704     \end{tabular}%
15705 }%
15706 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationTabularHeader}%
15707 \else
15708 \renewenvironment{theglossary}%
15709 {%
15710     \glspatchLToutput
15711     \glslongextraSymLocSetDescWidth
15712     \edef\@glslongextra@begintab{%
15713         \noexpand\begin{longtable}{%
15714             \expandonce\glslongextraNameAlign
15715             \expandonce\glslongextraDescAlign
15716             \expandonce\glslongextraSymbolAlign
15717             \expandonce\glslongextraLocationAlign
15718         } }%
15719     \@glslongextra@begintab
15720 }%
15721 {\end{longtable}}%
15722 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationHeader}%
15723 \fi
15724 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
15725 \renewcommand{\glossentry}[2]{%
15726     \glslongextraNameFmt{##1} &
15727     \glslongextraDescFmt{##1} &
15728     \glslongextraSymbolFmt{##1}&
15729     \glslongextraLocationFmt{##1}{##2}\tabularnewline
15730 }%
15731 \renewcommand{\subglossentry}[3]{%
15732     \glslongextraSubNameFmt{##1}{##2} &
15733     \glslongextraSubDescFmt{##1}{##2} &
15734     \glslongextraSubSymbolFmt{##1}{##2}&
15735     \glslongextraSubLocationFmt{##1}{##2}{##3}%
15736     \tabularnewline
15737 }%
15738 \ifglsgroupskip
15739     \renewcommand*{\glsgroupskip}{}%
15740 \else
15741     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15742 \fi
15743 }

```

meSymDescHeader

```
15744 \newcommand{\glslongextraNameSymDescHeader}{%
15745   \glslongextraNameSymDescTabularHeader\endhead
15746   \glslongextraNameSymDescTabularFooter\endfoot
15747 }
```

scTabularHeader

```
15748 \newcommand{\glslongextraNameSymDescTabularHeader}{%
15749   \toprule
15750   \glslongextraHeaderFmt\entryname &
15751   \glslongextraHeaderFmt\symbolname &
15752   \glslongextraHeaderFmt\descriptionname\tabularnewline
15753   \midrule
15754 }
```

scTabularFooter

```
15755 \newcommand{\glslongextraNameSymDescTabularFooter}{%
15756   \bottomrule
15757 }
```

g-name-sym-desc Three column style with symbol in the second column.

```
15758 \newglossarystyle{long-name-sym-desc}%
15759 {%
15760   \ifGlsLongExtraUseTabular
15761     \renewenvironment{theglossary}%
15762     {%
15763       \glslongextraSymSetDescWidth
15764       \edef\@glslongextra@begintab{%
15765         \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
15766         \expandonce\glslongextraNameAlign
15767         \expandonce\glslongextraSymbolAlign
15768         \expandonce\glslongextraDescAlign
15769       }}%
15770       \@glslongextra@begintab
15771     }%
15772     {%
15773       \glslongextraNameSymDescTabularFooter
15774       \end{tabular}%
15775     }%
15776     \renewcommand*\glossaryheader{\glslongextraNameSymDescTabularHeader}%
15777   \else
15778     \renewenvironment{theglossary}%
15779     {%
15780       \glspatchLToutput
15781       \glslongextraSymSetDescWidth
15782       \edef\@glslongextra@begintab{%
15783         \noexpand\begin{longtable}{%
15784         \expandonce\glslongextraNameAlign
15785         \expandonce\glslongextraSymbolAlign
```

```

15786         \expandonce\glslongextraDescAlign
15787     } }%
15788     \@glslongextra@begintab
15789     }%
15790     {\end{longtable}}%
15791     \renewcommand*\glossaryheader{\glslongextraNameSymDescHeader}%
15792     \fi
15793     \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15794     \renewcommand\glossentry[2]{%
15795         \glslongextraNameFmt{##1} &
15796         \glslongextraSymbolFmt{##1} &
15797         \glslongextraDescFmt{##1}\tabularnewline
15798     }%
15799     \renewcommand\subglossentry[3]{%
15800         \glslongextraSubNameFmt{##1}{##2} &
15801         \glslongextraSubSymbolFmt{##1}{##2} &
15802         \glslongextraSubDescFmt{##1}{##2}\tabularnewline
15803     }%
15804     \ifglsgroupskip
15805         \renewcommand*\glsgroupskip{}%
15806     \else
15807         \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15808     \fi
15809 }

```

cLocationHeader

```

15810 \newcommand{\glslongextraNameSymDescLocationHeader}{%
15811     \glslongextraNameSymDescLocationTabularHeader\endhead
15812     \glslongextraNameSymDescLocationTabularFooter\endfoot
15813 }

```

onTabularHeader

```

15814 \newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
15815     \toprule
15816     \glslongextraHeaderFmt\entryname &
15817     \glslongextraHeaderFmt\symbolname &
15818     \glslongextraHeaderFmt\descriptionname &
15819     \glslongextraHeaderFmt\pagelistname\tabularnewline
15820     \midrule
15821 }

```

onTabularFooter

```

15822 \newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
15823     \bottomrule
15824 }

```

me-sym-desc-loc Four column style with symbol in the second column.

```

15825 \newglossarystyle{long-name-sym-desc-loc}%
15826 {}%

```

```

15827 \ifGlsLongExtraUseTabular
15828   \renewenvironment{theglossary}%
15829   {%
15830     \glslongextraSymLocSetDescWidth
15831     \edef\@glslongextra@begintab{%
15832       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15833         \expandonce\glslongextraNameAlign
15834         \expandonce\glslongextraSymbolAlign
15835         \expandonce\glslongextraDescAlign
15836         \expandonce\glslongextraLocationAlign
15837       }}%
15838     \@glslongextra@begintab
15839   }%
15840   {%
15841     \glslongextraNameSymDescLocationTabularFooter
15842     \end{tabular}%
15843   }%
15844   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationTabularHeader}%
15845 \else
15846   \renewenvironment{theglossary}%
15847   {%
15848     \glspatchLToutput
15849     \glslongextraSymLocSetDescWidth
15850     \edef\@glslongextra@begintab{%
15851       \noexpand\begin{longtable}{%
15852         \expandonce\glslongextraNameAlign
15853         \expandonce\glslongextraSymbolAlign
15854         \expandonce\glslongextraDescAlign
15855         \expandonce\glslongextraLocationAlign
15856       }}%
15857     \@glslongextra@begintab
15858   }%
15859   {\end{longtable}}%
15860   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationHeader}%
15861 \fi
15862 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
15863 \renewcommand{\glossentry}[2]{%
15864   \glslongextraNameFmt{##1} &
15865   \glslongextraSymbolFmt{##1} &
15866   \glslongextraDescFmt{##1} &
15867   \glslongextraLocationFmt{##1}{##2}\tabularnewline
15868 }%
15869 \renewcommand{\subglossentry}[3]{%
15870   \glslongextraSubNameFmt{##1}{##2} &
15871   \glslongextraSubSymbolFmt{##1}{##2} &
15872   \glslongextraSubDescFmt{##1}{##2} &
15873   \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
15874 }%
15875 \ifglsnogroupskip

```

```

15876 \renewcommand*{\glsgroupskip}{}%
15877 \else
15878 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15879 \fi
15880 }

```

mDescNameHeader

```

15881 \newcommand{\glslongextraSymDescNameHeader}{%
15882 \glslongextraSymDescNameTabularHeader\endhead
15883 \glslongextraSymDescNameTabularFooter\endfoot
15884 }

```

meTabularHeader

```

15885 \newcommand{\glslongextraSymDescNameTabularHeader}{%
15886 \toprule
15887 \glslongextraHeaderFmt\symbolname &
15888 \glslongextraHeaderFmt\descriptionname &
15889 \glslongextraHeaderFmt\entryname\tabularnewline
15890 \midrule
15891 }

```

meTabularFooter

```

15892 \newcommand{\glslongextraSymDescNameTabularFooter}{%
15893 \bottomrule
15894 }

```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```

15895 \newglossarystyle{long-sym-desc-name}%
15896 {%
15897 \ifGlsLongExtraUseTabular
15898 \renewenvironment{theglossary}%
15899 {%
15900 \glslongextraSymSetDescWidth
15901 \edef\@glslongextra@begintab{%
15902 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15903 \expandonce\glslongextraSymbolAlign
15904 \expandonce\glslongextraDescAlign
15905 \expandonce\glslongextraNameAlign
15906 }}%
15907 \@glslongextra@begintab
15908 }%
15909 {%
15910 \glslongextraSymDescNameTabularFooter
15911 \end{tabular}%
15912 }%
15913 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
15914 \else
15915 \renewenvironment{theglossary}%

```

```

15916   {%
15917       \glspatchLToutput
15918       \glslongextraSymSetDescWidth
15919       \edef\@glslongextra@begintab{%
15920           \noexpand\begin{longtable}{%
15921               \expandonce\glslongextraSymbolAlign
15922               \expandonce\glslongextraDescAlign
15923               \expandonce\glslongextraNameAlign
15924           }}%
15925       \@glslongextra@begintab
15926   }%
15927   {\end{longtable}}}%
15928   \renewcommand*{\glossaryheader}{\glslongextraSymDescNameHeader}%
15929   \fi
15930   \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15931   \renewcommand{\glossentry}[2]{%
15932       \glslongextraSymbolFmt{##1} &
15933       \glslongextraDescFmt{##1} &
15934       \glslongextraNameFmt{##1}\tabularnewline
15935   }%
15936   \renewcommand{\subglossentry}[3]{%
15937       \glslongextraSubSymbolFmt{##1}{##2} &
15938       \glslongextraSubDescFmt{##1}{##2} &
15939       \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15940   }%
15941   \ifglsgnogroupskip
15942       \renewcommand*{\glsgroupskip}{}%
15943   \else
15944       \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15945   \fi
15946 }

```

meDescNameHeader

```

15947 \newcommand{\glslongextraLocationSymDescNameHeader}{%
15948   \glslongextraLocationSymDescNameTabularHeader\endhead
15949   \glslongextraLocationSymDescNameTabularFooter\endfoot
15950 }

```

meTabularHeader

```

15951 \newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
15952   \toprule
15953   \glslongextraHeaderFmt\pagelistname &
15954   \glslongextraHeaderFmt\symbolname &
15955   \glslongextraHeaderFmt\descriptionname &
15956   \glslongextraHeaderFmt\entryname\tabularnewline
15957   \midrule
15958 }

```

meTabularFooter

```

15959 \newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
15960 \bottomrule
15961 }

```

c-sym-desc-name Four column style with location list, symbol, description and name.

```

15962 \newglossarystyle{long-loc-sym-desc-name}%
15963 {%
15964 \ifGlsLongExtraUseTabular
15965 \renewenvironment{theglossary}%
15966 {%
15967 \glslongextraSymLocSetDescWidth
15968 \edef\@glslongextra@begintab{%
15969 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15970 \expandonce\glslongextraLocationAlign
15971 \expandonce\glslongextraSymbolAlign
15972 \expandonce\glslongextraDescAlign
15973 \expandonce\glslongextraNameAlign
15974 }}%
15975 \@glslongextra@begintab
15976 }%
15977 {%
15978 \glslongextraLocationSymDescNameTabularFooter
15979 \end{tabular}%
15980 }%
15981 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameTabularHeader}%
15982 \else
15983 \renewenvironment{theglossary}%
15984 {%
15985 \glspatchLToutput
15986 \glslongextraSymLocSetDescWidth
15987 \edef\@glslongextra@begintab{%
15988 \noexpand\begin{longtable}{%
15989 \expandonce\glslongextraLocationAlign
15990 \expandonce\glslongextraSymbolAlign
15991 \expandonce\glslongextraDescAlign
15992 \expandonce\glslongextraNameAlign
15993 }}%
15994 \@glslongextra@begintab
15995 }%
15996 {\end{longtable}}%
15997 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameHeader}%
15998 \fi
15999 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16000 \renewcommand{\glossentry}[2]{%
16001 \glslongextraLocationFmt{##1}{##2} &
16002 \glslongextraSymbolFmt{##1} &
16003 \glslongextraDescFmt{##1} &
16004 \glslongextraNameFmt{##1}\tabularnewline
16005 }%

```

```

16006 \renewcommand{\subglossentry}[3]{%
16007   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16008   \glslongextraSubSymbolFmt{##1}{##2} &
16009   \glslongextraSubDescFmt{##1}{##2} &
16010   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16011 }%
16012 \ifglsnogroupskip
16013   \renewcommand*{\glsgroupskip}{}%
16014 \else
16015   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16016 \fi
16017 }

```

scSymNameHeader

```

16018 \newcommand{\glslongextraDescSymNameHeader}{%
16019   \glslongextraDescSymNameTabularHeader\endhead
16020   \glslongextraDescSymNameTabularFooter\endfoot
16021 }

```

meTabularHeader

```

16022 \newcommand{\glslongextraDescSymNameTabularHeader}{%
16023   \toprule
16024   \glslongextraHeaderFmt\descriptionname &
16025   \glslongextraHeaderFmt\symbolname &
16026   \glslongextraHeaderFmt\entryname\tabularnewline
16027   \midrule
16028 }

```

meTabularFooter

```

16029 \newcommand{\glslongextraDescSymNameTabularFooter}{%
16030   \bottomrule
16031 }

```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```

16032 \newglossarystyle{long-desc-sym-name}%
16033 {%
16034   \ifGlsLongExtraUseTabular
16035     \renewenvironment{theglossary}%
16036     {%
16037       \glslongextraSymSetDescWidth
16038       \edef\@glslongextra@begintab{%
16039         \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
16040         \expandonce\glslongextraDescAlign
16041         \expandonce\glslongextraSymbolAlign
16042         \expandonce\glslongextraNameAlign
16043       }}%
16044       \@glslongextra@begintab
16045     }%

```

```

16046   {%
16047       \glslongextraDescSymNameTabularFooter
16048       \end{tabular}%
16049   }%
16050   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameTabularHeader}%
16051 \else
16052   \renewenvironment{theglossary}%
16053   {%
16054       \glspatchLToutput
16055       \glslongextraSymSetDescWidth
16056       \edef\@glslongextra@begintab{%
16057         \noexpand\begin{longtable}{%
16058           \expandonce\glslongextraDescAlign
16059           \expandonce\glslongextraSymbolAlign
16060           \expandonce\glslongextraNameAlign
16061         }}%
16062       \@glslongextra@begintab
16063   }%
16064   {\end{longtable}}%
16065   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameHeader}%
16066 \fi
16067 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16068 \renewcommand{\glossentry}[2]{%
16069   \glslongextraDescFmt{##1} &
16070   \glslongextraSymbolFmt{##1} &
16071   \glslongextraNameFmt{##1}\tabularnewline
16072 }%
16073 \renewcommand{\subglossentry}[3]{%
16074   \glslongextraSubDescFmt{##1}{##2} &
16075   \glslongextraSubSymbolFmt{##1}{##2} &
16076   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16077 }%
16078 \ifglsgnogroupskip
16079   \renewcommand*{\glsgroupskip}{}%
16080 \else
16081   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16082 \fi
16083 }

```

scSymNameHeader

```

16084 \newcommand{\glslongextraLocationDescSymNameHeader}{%
16085   \glslongextraLocationDescSymNameTabularHeader\endhead
16086   \glslongextraLocationDescSymNameTabularFooter\endfoot
16087 }

```

meTabularHeader

```

16088 \newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
16089   \toprule
16090   \glslongextraHeaderFmt\pagelistname &

```

```

16091 \glslongextraHeaderFmt\descriptionname &
16092 \glslongextraHeaderFmt\symbolname &
16093 \glslongextraHeaderFmt\entryname\tabularnewline
16094 \midrule
16095 }

```

meTabularFooter

```

16096 \newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
16097 \bottomrule
16098 }

```

c-desc-sym-name Four column style with location list, description, symbol and name.

```

16099 \newglossarystyle{long-loc-desc-sym-name}%
16100 {%
16101 \ifGlsLongExtraUseTabular
16102 \renewenvironment{theglossary}%
16103 {%
16104 \glslongextraSymLocSetDescWidth
16105 \edef\@glslongextra@begintab{%
16106 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16107 \expandonce\glslongextraLocationAlign
16108 \expandonce\glslongextraDescAlign
16109 \expandonce\glslongextraSymbolAlign
16110 \expandonce\glslongextraNameAlign
16111 }}%
16112 \@glslongextra@begintab
16113 }%
16114 {%
16115 \glslongextraLocationDescSymNameTabularFooter
16116 \end{tabular}%
16117 }%
16118 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameTabularHeader}%
16119 \else
16120 \renewenvironment{theglossary}%
16121 {%
16122 \glspatchLToutput
16123 \glslongextraSymLocSetDescWidth
16124 \edef\@glslongextra@begintab{%
16125 \noexpand\begin{longtable}%
16126 \expandonce\glslongextraLocationAlign
16127 \expandonce\glslongextraDescAlign
16128 \expandonce\glslongextraSymbolAlign
16129 \expandonce\glslongextraNameAlign
16130 }}%
16131 \@glslongextra@begintab
16132 }%
16133 {\end{longtable}}%
16134 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameHeader}%
16135 \fi

```

```

16136 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
16137 \renewcommand{\glossentry}[2]{%
16138   \glslongextraLocationFmt{##1}{##2} &
16139   \glslongextraDescFmt{##1} &
16140   \glslongextraSymbolFmt{##1} &
16141   \glslongextraNameFmt{##1}\tabularnewline
16142 }%
16143 \renewcommand{\subglossentry}[3]{%
16144   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16145   \glslongextraSubDescFmt{##1}{##2} &
16146   \glslongextraSubSymbolFmt{##1}{##2} &
16147   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16148 }%
16149 \ifglsnogroupskip
16150   \renewcommand*{\glsgroupskip}{}%
16151 \else
16152   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16153 \fi
16154 }

```

Glossary

bib2gls A command line Java application that selects entries from a .bib file and converts them to glossary definitions (like bibtex but also performs hierarchical sorting and collation, thus omitting the need for **xindy** or **makeindex**). Further details at: [http://www.dickimaw-books.com/software/bib2gls/..](http://www.dickimaw-books.com/software/bib2gls/)

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	82
General: Initial experimental release	5	\@Glsymbolplural@: added redefinition	82
0.2 (2015-11-30)		\@Glstext@: added redefinition	78
\Glsfmtshort: new	334	\@Glsuseri@: added redefinition	83
\glsfmtshort: new	333	\@Glsuserii@: added redefinition	83
\Glsfmtshortpl: new	334	\@Glsuseriii@: added redefinition	83
\glsfmtshortpl: new	333	\@Glsuseriv@: added redefinition	84
short: switched inline full form to short (long)	236	\@Glsuserv@: added redefinition	84
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	84
\@ACRlong: added redefinition	87	\@acrlong: added redefinition	87
\@ACRlongpl: added redefinition	88	\@acrlongpl: added redefinition	88
\@ACRshort: added redefinition	85	\@acrshort: added redefinition	85
\@ACRshortpl: added redefinition	86	\@acrshortpl: added redefinition	86
\@Acrlong: added redefinition	87	\@gls@field@link: added optional argument	69
\@Acrlongpl: added redefinition	88	\@glsgdescplural@: added redefinition	81
\@Acrshort: added redefinition	85	\@glsgfirst@: added redefinition	78
\@Acrshortpl: added redefinition	86	\@glsgfirstplural@: added redefinition	79
\@GLSdesc@: added redefinition	81	\@glsgplural@: added redefinition	79
\@GLSdescplural@: added redefinition	82	\@glssymbolplural@: added redefinition	82
\@GLSfirst@: added redefinition	79	\@glsgxtr@defaultnoglossarywarning: new	141
\@GLSfirstplural@: added redefinition	80	\@glsgxtr@field@linkdefs: new	77
\@GLSname@: added redefinition	81	\@glsgxtr@insertdots: new	204
\@GLSplural@: added redefinition	79	\@print@glossary: added redefinition	138
\@GLSSymbol@: added redefinition	82	\glsabbrvdefaultfont: renamed from \abbrvdefaultfont	210
\@GLSSymbolplural@: added redefinition	83	\glsaccessdesc: new	168
\@GLStext@: added redefinition	78	\glsaccessdescplural: new	168
\@GLSuseri@: added redefinition	83	\glsaccessfirst: new	165
\@GLSuserii@: added redefinition	83	\glsaccessfirstplural: new	166
\@GLSuseriii@: added redefinition	84	\Glsaccesslong: new	170
\@GLSuseriv@: added redefinition	84	\glsaccesslong: new	170
\@GLSuserv@: added redefinition	84	\glsaccessname: new	164
\@GLSuservi@: added redefinition	84	\glsaccessplural: new	165
\@GLsgdesc@: added redefinition	81	\Glsaccessshort: new	169
\@GLsgdescplural@: added redefinition	81	\glsaccessshort: new	169
\@GLsgfirst@: added redefinition	78	\Glsaccessshortpl: new	170
\@GLsgfirstplural@: added redefinition	80		
\@GLsgname@: added redefinition	80		
\@GLsgplural@: added redefinition	79		

\glsaccesssshortpl: new	170	\cGLSpl: new	114
\glsaccessssymbol: new	167	\cGLSpl@: new	114
\glsaccessssymbolplural: new	167	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	164	new	109
\glsentryfmt: added check for short ..	68	\cGLS: new	114
\glslongpltok: new	204	\cGLSformat: new	114
\glsshortpltok: new	204	\cGLSpl: new	114
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	114
name in \setkeys	206	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	18
for plural	201	\glsenableentrycount: new	110
\GLSxtrlongpl: new	220	\glsfirstabbrvdefaultfont: new ..	210
\Glsxtrlongpl: new	220	\glsfirstlongdefaultfont: new ...	210
\glsxtrlongpl: new	219	\Glsfmtfirst: new	336
\glsxtrNoGlossaryWarning: new	23	\glsfmtfirst: new	336
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	337
new	200	\glsfmtfirstpl: new	336
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	336
new	201	\glsfmtplural: new	335
\glsxtrpostlinkendsentence: new ..	200	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	219	\Glsxtrtitleshort	334
\Glsxtrshortpl: new	218	renamed from \Glsentryfmtshort ..	334
\glsxtrshortpl: new	217	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\Glsxtrtitleshort	333
\glslabeltok	231	renamed from \glsentryfmtshort ..	333
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	229	\Glsxtrtitleshortpl	334
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glsentryfmtshortpl	334
redefinition of \acronymtype	20	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\Glsxtrtitleshortpl	333
\Glsxtrshort	334	renamed from	
\glsfmtshort: changed to use		\Glsentryfmtshortpl	333
\glsxtrshort	333	\Glsfmttext: new	335
\Glsfmtshortpl: changed to use		\glsfmttext: new	335
\glsxtrshortpl	334	\glshasattribute: new	179
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	178
\glsxtrshortpl	333	\glsxtremsuffix: new	272
\glsxtrifemptyglossary: new	31	\GlsXtrEnableEntryCounting: new ..	109
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	112
argument	182	\glsxtrscfont: new	244
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	244
argument	182	\glsxtrsmfont: new	258
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	259
default type to \acronymtype	123	short-em: new	279
\newterm: fixed name argument	181	short-em-desc: new	281
0.5 (2015-12-07)		short-em-footnote: new	290
\@cGLS: new	114	short-em-long: new	276
\@cGLS@: new	114	short-em-long-desc: new	277

short-em-postfootnote: new	292	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	255	attribute	324
short-sc-postfootnote: new	256	\Glsxtrheadtext: now uses headuc	
short-sm: new	262	attribute	327
short-sm-desc: new	263	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	269	attribute	326
short-sm-long: new	260	short-em-footnote: switch off regular	
short-sm-long-desc: new	261	attribute if set	290
short-sm-postfootnote: new	270	short-long: switch off regular attribute	
long-noshort-em: new	283	if set	230
long-noshort-em-desc: new	287	short-long-desc: switch off regular	
long-noshort-sm: new	265	attribute if set	231
long-noshort-sm-desc: new	267	short-sc-footnote: switch off regular	
long-short-em: new	273	attribute if set	255
long-short-em-desc: new	274	short-sm-footnote: switch off regular	
long-short-sm: new	259	attribute if set	269
long-short-sm-desc: new	260	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	228
\Glsaccessstext: new	165	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	229
\@glxtr@doaccsupp: new	23	long-short-sc-desc: switch off regular	
General: removed \ifglxtruseuchead	324	attribute if set	246
\Glsaccessdesc: new	168	footnote: switch off regular attribute if	
\Glsaccessdescplural: new	169	set	232
\Glsaccessfirst: new	166	postfootnote: switch off regular	
\Glsaccessfirstplural: new	166	attribute if set	234
\Glsaccessname: new	164	0.5.2 (2015-12-08)	
\Glsaccessplural: new	165	\@GLSdesc@: added accessibility support	81
\Glsaccesssymbol: new	167	\@GLSdescplural@: added accessibility	
\Glsaccesssymbolplural: new	167	support	82
\glxtrheadfirst: now uses headuc		\@GLSfirst@: added accessibility	
attribute	328	support	79
\glxtrheadfirst: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	328	support	80
\Glsxtrheadfirstplural: now uses		\@GLSname@: added accessibility support	81
headuc attribute	329	\@GLSplural@: added accessibility	
\glxtrheadfirstplural: now uses		support	79
headuc attribute	329	\@GLSsymbol@: added accessibility	
\Glsxtrheadplural: now uses headuc		support	82
attribute	327	\@GLSsymbolplural@: added	
\glxtrheadplural: now uses headuc		accessibility support	83
attribute	327	\@GLStext@: added accessibility support	78
\Glsxtrheadshort: now uses headuc		\@GLSdesc@: added accessibility support	81
attribute	325	\@GLSdescplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	81
attribute	324	\@GLSfirst@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	78
attribute	325	\@GLSfirstplural@: added accessibility	
		support	80

\@Glsname@: add accessibility support ..	80	\GLSaccesssymbolplural: new ..	168, 176
\@Glsplural@: added accessibility support	79	\GLSaccessstext: new	165, 174
\@Glsymbol@: added accessibility support	82	\glsentryfmt: moved	
\@Glsymbolplural@: added accessibility support	82	\glssetabbrvfmt from	
\@Glstext@: added accessibility support	78	\glsxtrabbrvfmt to here	68
\@glsdesc@: added accessibility support	81	\GlsXtrEnableInitialTagging: new	196
\@glsdescplural@: added accessibility support	81	\glsxtrfieldtitlecase: new	183
\@glsfirst@: added accessibility support	78	\GlsXtrFormatLocationList: new ...	66
\@glsfirstplural@: added accessibility support	79	\glsxtrnewabbrevpresetkeyhook:	
\@Glsname@: added accessibility support	80	new	208
\@Glsplural@: added accessibility support	79	\glsxtrtagfont: new	198
\@Glsymbol@: added accessibility support	82	\KV@printgloss@nonumberlist: added	68
\@Glsymbolplural@: added accessibility support	82	\mfu@checkword@do: added	197
\@Glstext@: added accessibility support	78	\setabbreviationstyle: added check	
\@glsxtr@activate@initialtagging:		for post-definition style switch	224
new	198	0.5.3 (2015-12-09)	
\@glsxtr@do@titlecaps@warn: new ..	198	\@glsxtr@autoindex@at: new	194
\@glsxtr@tag: new	198	\@glsxtr@autoindex@encap: new ...	194
General: fixed typo in glossaries-accsupp		\@glsxtr@autoindex@esc: new	195
and tidied up code to use just one		\@glsxtr@autoindex@level: new ...	194
\@ifpackageloaded	164	\@glsxtr@autoindex@setname: new ..	192
removed \glsxtrabbrvfmt	221	\@glsxtr@doabbreviationsdef: new ..	20
\glossaryentrynumbers: added	66	General: removed	
\Glossentrydesc: added	196	\GlsXtrNoGlsWarningNoAutoMakeMain	
\Glossentryname: added	187	140
\Glossentrysymbol: added	196	\glsdescwidth: added	65
\glossentrysymbol: added	196	\glspagelistwidth: added	65
\GLSaccessdesc: new	168, 176	\glsxtrdoautoindexname: new	192
\GLSaccessdescplural: new ...	169, 176	\glsxtrpostnamehook: new	189
\GLSaccessfirst: new	166, 175	\if@glsxtr@format@override: new ..	191
\GLSaccessfirstplural: new ..	166, 175	\ProvidesGlossariesExtraLang: new	339
\GLSaccesslong: new	170, 177	\RequireGlossariesExtraLang: new	339
\GLSaccesslongpl: new	171, 177	0.5.4 (2015-12-15)	
\Glsaccesslongpl: new	171	\@newglossaryentry@defunitcounters:	
\glsaccesslongpl: new	171	new	115
\GLSaccessname: new	164, 174	\@GLSxtr@p@acrlong@: new	101
\GLSaccessplural: new	165, 175	\@GLSxtr@p@acrlongpl@: new	101
\GLSaccessshort: new	169, 176	\@GLSxtr@p@acrshort@: new	100
\GLSaccessshortpl: new	170, 177	\@GLSxtr@p@acrshortpl@: new	100
\GLSaccesssymbol: new	167, 175	\@GLSxtr@p@long@: new	100
		\@GLSxtr@p@longpl@: new	100
		\@GLSxtr@p@plural@: new	99
		\@GLSxtr@p@short@: new	99
		\@GLSxtr@p@shortpl@: new	99
		\@GLSxtr@p@text@: new	98
		\@GlsXtrEnableOnTheFly: new	62
		\@Glsxtr: new	62
		\@Glsxtr@p@acrlong@: new	101
		\@Glsxtr@p@acrlongpl@: new	101

\@Glsxtr@p@acrshort@: new	100	\glsenableentryunitcount: new	117
\@Glsxtr@p@acrshortpl@: new	100	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	100	entry's existence	179
\@Glsxtr@p@longpl@: new	100	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	99	entry's existence	179
\@Glsxtr@p@short@: new	99	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	99	check	199
\@Glsxtr@p@text@: new	98	\Glsxtr: new	62
\@Glsxtrpl: new	63	\glxtr: new	62
\@alt@glshyp@opt: new	94	\glxtrcat: new	62
\@glscalt@hyp@opt: new	94	\glxtrdowrglossaryhook: new	94
\@glscalt@hyp@opt@char: new	94	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	95	new	120
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	61
new	116	\Glsxtrpl: new	63
\@glsc@local@increment@currunitcount:		\glxtrpl: new	63
new	117	\glxtrpostlocalreset: new	108
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	108
new	92	\glxtrpostreset: new	108
\@glsxtr: new	62	\glxtrpostunset: new	107
\@glsxtr@addunitcounter: new	116	\glxtrprotectlinks: new	97
\@glsxtr@currunitcount: new	117	\GlsXtrSetAltModifier: new	95
\@glsxtr@ifunitcounter: new	116	\GlsXtrSetDefaultGlsOpts: new	93
\@glsxtr@p@acrlong@: new	100	\glxtrstarflywarn: new	61
\@glsxtr@p@acrlongpl@: new	101	\GlsXtrWarning: new	63
\@glsxtr@p@acrshort@: new	100	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new	100	disables \setacronymstyle	123
\@glsxtr@p@long@: new	100	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new	100	\@glsxtr@autoindexcrossrefs: new	17
\@glsxtr@p@plural@: new	98	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new	99	new	132
\@glsxtr@p@shortpl@: new	99	\@glsxtr@idx@entrynumberlist: new	133
\@glsxtr@p@text@: new	98	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new	117	new	132
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new	121	new	133
\@glsxtr@unitcountlist: new	115	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new	63	new	132
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new	124
empty see value if not set and added		\makeglossaries: new	124
'see' to field key map	50	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	61	\glxtrdiscardperiod: added check	
\cGlsformat: added	115	for first use	201
\cGlsformat: added	115	short-desc: fixed typo in	
\cGlsplformat: added	115	\glxtrinlinfullformat and	
\cGlsplformat: added	115	added missing second argument	238
\glstdisablehyper: added	97	1.02 (2016-04-25)	
\glstdohyperlink: added	95	\@glsxtr@current@style: new	64
\glstdonohyperlink: added	97	\Glsfmtfull: new	338

\glsfmtfull: new	338	\@GLSdescplural@: set abbreviation and regular format	82
\Glsfmtfullpl: new	339	\@GLSfirst@: set abbreviation format ..	79
\glsfmtfullpl: new	339	\@GLSfirstplural@: set abbreviation and regular format	80
\Glsfmtlong: new	337	\@GLSname@: set abbreviation and regular format	81
\glsfmtlong: new	337	\@GLSplural@: set abbreviation and regular format	79
\Glsfmtlongpl: new	338	\@GLSsymbol@: set regular format	82
\glsfmtlongpl: new	338	\@GLSsymbolplural@: set regular format	83
\Glsxtrheadfull: new	332	\@GLStext@: set abbreviation and regular format	78
\glsxtrheadfull: new	331	\@GLSuseri@: set regular format	83
\Glsxtrheadfullpl: new	332	\@GLSuserii@: set regular format	83
\glsxtrheadfullpl: new	332	\@GLSuseriii@: set regular format	84
\Glsxtrheadlong: new	330	\@GLSuseriv@: set regular format	84
\glsxtrheadlong: new	329	\@GLSuservi@: set regular format	84
\Glsxtrheadlongpl: new	331	\@GLSdesc@: set abbreviation and regular format	81
\glsxtrheadlongpl: new	330	\@GLSdescplural@: set abbreviation and regular format	81
\Glsxtrtitlefull: new	332	\@GLSfirst@: set abbreviation and regular format	78
\glsxtrtitlefull: new	331	\@GLSfirstplural@: set abbreviation and regular format	80
\Glsxtrtitlefullpl: new	333	\@GLSname@: set abbreviation and regular format	80
\glsxtrtitlefullpl: new	332	\@GLSplural@: set abbreviation and regular format	79
\Glsxtrtitlelong: new	331	\@GLSsymbol@: set regular format	82
\glsxtrtitlelong: new	330	\@GLSsymbolplural@: set regular format	82
\Glsxtrtitlelongpl: new	331	\@GLStext@: set abbreviation and regular format	78
\glsxtrtitlelongpl: new	330	\@GLSuseri@: set regular format	83
\ifglxtrinsertinside: new	227	\@GLSuserii@: set regular format	83
postfootnote: added redef of		\@GLSuseriii@: set regular format	83
\glxtrsetupfulldefs	235	\@GLSuseriv@: set regular format	84
stylemods: new	24	\@GLSuservi@: set regular format	84
1.03 (2016-04-27)		\@GLSdesc@: set abbreviation and regular format	81
\@GLSfirstplural@: bug fix: misspelt cs name	80	\@GLSdescplural@: set abbreviation and regular format	79
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	79	\@GLSsymbol@: set regular format	82
\@GLSfirstplural@: bug fix: misspelt cs name	80	\@GLSsymbolplural@: set regular format	82
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	79	\@GLStext@: set abbreviation and regular format	78
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	79	\@GLSuseri@: set regular format	83
\glsxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	330	\@GLSuserii@: set regular format	83
\glsxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	324	\@GLSuseriii@: set regular format	83
1.04 (2015-04-30)		\@GLSuseriv@: set regular format	84
short-em-footnote: renamed from “footnote-em”	290	\@GLSuservi@: set regular format	84
1.04 (2016-05-02)		\@gls@preglossaryhook: added check for entry’s existence	198
\@@glxtrpostloctag: new	67	\@GLSdesc@: set abbreviation and regular format	81
\@GLSdesc@: set abbreviation and regular format	81	\@GLSdescplural@: set abbreviation and regular format	81
		\@GLSfirst@: set abbreviation and regular format	78

\@glsfirstplural@: set abbreviation and regular format	79	\glsxtruserparen: new	294
\@glsname@: set abbreviation and regular format	80	\glsxtrusersuffix: new	295
\@glsplural@: set abbreviation and regular format	79	\GlsXtrWarnDeprecatedAbbrStyle: new	226
\@glsymbol@: set regular format	82	short-em-long-em: new	278
\@glsymbolplural@: set regular format	82	short-em-long-em-desc: new	279
\@glstext@: set abbreviation and regular format	78	short-em-nolong: new	281
\@glsxtr@deprecated@abbrstyle: new	226	short-em-nolong-desc: new	282
\@glsxtr@do@style: new	25	short-em-postfootnote: renamed from “postfootnote-em”	292
\@glsxtr@doloctag: new	68	short-footnote: new	234
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand	133	short-long-user: new	301
\@glsxtr@pagetag: new	67	short-long-user-desc: new	302
\@glsxtr@pagetag: new	67	short-nolong: new	237
\@glsxtr@preloctag: new	68	short-nolong-desc: new	239
\@glsxtr@postloctag: new	68	short-postfootnote: new	236
\@glsxtr@preloctag: new	67	short-sc-footnote: renamed from “footnote-sc”	255
\glossentrydesc: added glossdescfont attribute check	183	short-sc-nolong: new	249
\Glossentryname: added glossnamefont attribute check	187	short-sc-nolong-desc: new	250
\glossentryname: added glossnamefont attribute check	185	short-sc-postfootnote: renamed from “postfootnote-sc”	256
moved post name hook inside condition	187	short-sm-footnote: renamed from “footnote-sm”	269
\glsabbrvmfont: new	272	short-sm-nolong: new	263
\glsabbrvuserfont: new	294	short-sm-nolong-desc: new	265
\glsfirstabbrvmfont: new	272	short-sm-postfootnote: renamed from “postfootnote-sm”	270
\glsfirstabbrvuserfont: new	294	\letabbreviationstyle: new	226
\glsfirstlongemfont: new	272	\newabbreviationstyle: bug fix: corrected test for existence	225
\glsfirstlonguserfont: new	295	long-em-noshort-em: new	285
\glsifnotregularcategory: new ...	180	long-em-noshort-em-desc: new	288
\glslongdefaultfont: new	210	long-em-short-em: new	274
\glslongemfont: new	273	long-em-short-em-desc: new	275
\glslongfont: new	210	long-noshort: new	243
\glslonguserfont: new	294	long-noshort-desc: new	242
\glsxtrassignfieldfont: new	77	long-noshort-em: renamed from “long-em”	283
\GlsXtrEnablePreLocationTag: new .	66	long-noshort-em-desc: renamed from “long-desc-em”	287
\glsxtrfirstscfont: new	244	long-noshort-sc: renamed from “long-sc”	251
\glsxtrfirstsmfont: new	258	long-noshort-sc-desc: renamed from “long-desc-sc”	253
\glsxtrlongshortdescsort: new ...	229	long-noshort-sm: renamed from “long-sm”	265
\glsxtrpostnamehook: added category check	189	long-noshort-sm-desc: renamed from \long-desc-sm	267
\glsxtrregularfont: new	69		
\glsxtruserfield: new	294		

long-short-user: new	295	docdef option changed to choice	16
long-short-user-desc: new	301	\glxtr@usesee: new	51
\renewabbreviationstyle: new	226	\glxtrusesee: new	51
style: new	25	\glxtruseseeformat: new	51
1.05 (2016-06-10)		\if@glxtrdocdefrestricted: new ..	17
\eglssetwidest: new	404	1.07 (2016-08-15)	
\glFindWidestAnyName: new	406	\@@glxtrp: new	101
\glFindWidestAnyNameLocation:		\@GLSfirst@: added check for	
new	412	nohyperfirst attribute	79
\glFindWidestAnyNameSymbol: new	409	\@GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute	80
new	411	\@GLSxtrp: new	102
\glFindWidestLevelTwo: new	408	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	406	nohyperfirst attribute	78
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new	411	nohyperfirst attribute	80
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new	102
new	409	\@gls@preglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts	199
new	410	\@glsfirst@: added check for	
\glFindWidestUsedLevelTwo: new .	407	nohyperfirst attribute	78
\glFindWidestUsedTopLevelName:		\@glsfirstplural@: added check for	
new	405	nohyperfirst attribute	80
\glfirstlongfootnotefont: new ..	232	\@glxtrinmark: new	321
\glsgetwidestname: new	405	\@glxtrnotinmark: new	322
\glsgetwidestsubname: new	405	\@glxtrp: new	101
\glslongfootnotefont: new	232	\@glxtrp@opt: new	101
\glxtrAltTreeIndent: new	403	\glossxtrsetpopts: new	101
\glxtralttreeInit: new	404	\glsp: new	104
\glxtrAltTreePar: new	403	\glsp: new	104
\glxtrAltTreeSetHangIndent: new	413	\glxtr@entry@p: new	102
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new	232
new	413	\glxtrchecknohyperfirst: new	78
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new	183
new	403	\glxtrifinmark: new	321
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new	105
new	403	\Glsxtrp: new	104
\glxtrComputeTreeIndent: new ...	412	\glxtrp: new	103
\glxtrComputeTreeSubIndent: new	413	\glxtrsetpopts: new	101
\glxtrtreetopindent: new	404	short-long-desc: added text key	231
short-em-long: fixed incorrect font used		fixed misspelling of \glabbrvfont in	
by long form	277	plural key	231
\xglsetwidest: new	404	long-short-desc: added missing text	
1.06 (2016-06-18)		key	229
\@glsoifexistsorwarn: new	17	fixed misspelling of \glabbrvfont .	229
\@glxtr@docdefval: new	16	footnote: changed first forms to use	
\@glxtr@usesee: new	51	\glfirstlongfootnotefont ...	232
General: disabled docdef key at the start		postfootnote: removed \footnote	
of the document	30	from first keys	234

switched from \glsfirstlongfont to	1.10 (2016-12-17)
\glsfirstlongfootnotefont ... 235	\@GLSp1@: fixed bug caused by typo in
\RestoreAcronyms: modified	command name 70
\@gls@link@checkfirsthyper to	1.11 (2017-01-19)
set \glsxtrifwasfirstuse 123	\@glsxtr@do@redef@forglsentries:
1.08 (2016-12-13)	new 6
\@glsxtr@record: new 8	\@glsxtr@noidx@do: new 153
\@GLS@: added \@glsxtr@record 70	\@glsxtr@redef@forglsentries: new . 6
\@GLSp1@: added \@glsxtr@record ... 70	\@glsxtr@shortcutsval: new 22
\@Gls@: added \@glsxtr@record 70	\@glsxtr@unsrt@getgrouptitle: new 152
\@GLspl@: added \@glsxtr@record ... 70	\@print@noidx@glossary: added
\@gls@: added \@glsxtr@record 69	redefinition 135
\@gls@@link@: added	\glsxtr@addloclistfield: added
\@glsxtr@record 71	group key 14
\@gls@field@link: added	added location key 14
\@glsxtr@record 69	\glsxtr@fields: new 144
\@gls@saveentrycounter: new 29	\glsxtr@linkprefix: new 144
\@glsdisp: added \@glsxtr@record .. 70	\glsxtr@org@newignoredglossary:
\@glspl@: added \@glsxtr@record ... 70	new 46
\@glsxtr@dorecord: new 10	\glsxtr@s@newignoredglossary: new 46
\@glsxtr@err@undefaction: new 6	\glsxtr@shortcutsval: new 144
\@glsxtr@record: new 7	\glsxtr@texencoding: new 144
\@glsxtr@warn@onexistsordo: new ... 6	\glsxtr@writefields: new 144
\@glsxtr@warn@undefaction: new 6	\GlsXtrLoadResources: new 144
\@print@unsrt@glossary: new 149	\glsxtrpageref: new 43
record: added record package option ... 15	\glsxtrresourcefile: changed
\glsadd: added \@glsxtr@record 76	extension to .gls tex 143
\glsdoifexists: now defines	\newignoredglossary: added starred
\glslabel 49	version 46
\glsxtr@do@wrglossary: new 29	1.12 (2017-02-03)
\glsxtr@addloclistfield: new 13	\@glsxtr@recordcounter: new 12
\glsxtr@indexonly@saveentrycounter:	\@gls@preglossaryhook: check for
new 13	definition 198
\glsxtr@record: new 146	\@glsxtr@counterrecordhook: new . 146
\glsxtr@resource: new 144	\@glsxtr@display@loc: new 136
\glsxtr@saveentrycounter: new 29	\@glsxtr@docounterrecord: new ... 147
\glsxtr@setup@record: new 13	\@glsxtr@longnewglossaryentry:
\glsxtrassignfieldfont: added check	new 45
for existence 77	\@glsxtr@noop@recordcounter: new . 13
\glsxtrresourcefile: new 143	\@glsxtr@op@recordcounter: new ... 13
\printunsrtglossaries: new 149	\@glsxtr@provide@storagekey: new . 31
\printunsrtglossary: new 149	\@glsxtr@s@longnewglossaryentry:
1.09 (2016-12-16)	new 45
\@glsxtr@gettype: new 131	\@glsxtrentryfmt: new 34
\@glsxtr@mixed@assign@sortkey:	\@glsxtrindexaliased: new 92
new 131	\@glsxtrsetaliasnoindex: new 92
\@printglossary: redefined to save	\@newglossaryentryposthook: added
options 130	check for alias key 57
\glsxtr@makeglossaries: new 131	\@no@glsxtrindexaliased: new 93

\@printunsrtglossary: new	149	\GlsXtrLoadResources: removed	
General: added target key to printgloss		restriction on only one per document	144
family	130	\glxtrlocrangefmt: new	137
\apptoglossarypreamble: new	44	\glxtrpostlongdescription: new ..	46
\csGlsXtrLetField: new	39	\glxtrprovidestoragekey: new	31
\eGlsXtrSetField: new	39	\GlsXtrRecordCounter: new	146
\gGlsXtrSetField: new	39	\glxtrresourcecount: new	144
\glsdohyperlink: added check for alias		\glxtrresourcefile: added catcode	
field	96	change for @	143
\glsnoidxdisplayloc: added		\glxtrsetaliasnoindex: new	92
redefinition	135	\GlsXtrSetField: new	38
\glissettoctitle: added patch	47	\glxtrsetfieldifexists: new	38
\glxtr@counterrecord: new	146	\glxtrunsrtdo: new	152
\glxtr@langtag: new	144	\GlsXtrusefield: new	37
\glxtr@newabbreviation: new	206	\glxtrusefield: new	37
\glxtr@org@newignoredglossary:		short-postlong-user: new	298
Added check for existence	46	short-postlong-user-desc: new ...	300
\glxtr@pluralsuffixes: new	144	\longnewglossaryentry: added starred	
\glxtr@provideignoredglossary:		version	45
new	48	long-postshort-user: new	296
\glxtr@s@newignoredglossary:		long-postshort-user-desc: new ...	298
Added check for existence	46	postdot: new	19
\glxtr@s@provideignoredglossary:		\pretoglossarypreamble: new	44
new	48	\print@noop@unsrtglossaryunit:	
\glxtrabbrvpluralsuffix: new ...	210	new	152
\glxtralias: new	56	\print@op@unsrtglossaryunit: new	151
\glxtrcopytoglossary: new	49	\printunsrtglossary: added starred	
\glxtrdeffield: new	38	form	149
\glxtrdisplayendloc: new	136	\printunsrtglossaryhandler: new .	151
\glxtrdisplayendloohook: new ...	137	\printunsrtglossaryunit: new	13
\glxtrdisplaysingleloc: new	136	\printunsrtglossaryunitsetup: new	151
\glxtrdisplaystartloc: new	136	\provideignoredglossary: new	47
\glxtrdeffield: new	38	\s@glxtr@provide@storagekey: new	32
\glxtrentryfmt: new	34	\s@printunsrtglossary: new	149
\glxtrfieldddolistloop: new	35	\xGlsXtrSetField: new	39
\glxtrfieldforlistloop: new	35		
\glxtrfielddifylist: new	35	1.13 (2017-02-07)	
\glxtrfieldlistadd: new	34	\@glsdisp: removed	
\glxtrfieldlistead: new	34	\@glxtr@org@glsdisp	70
\glxtrfieldlistgadd: new	34	\glxtrsetaliasnoindex: switched to	
\glxtrfieldlistxadd: new	34	\providecommand	92
\glxtrfieldxifylist: new	35		
\glxtrfmt: new	32	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	32	\@gls@link: added redefinition	73
\GlsXtrFmtField: new	32	\@gls@noidx@getgrouptitle: new ..	133
\glxtrifkeydefined: new	31	\@gls@removespaces: new	137
\glxtrindexaliased: new	93	\@glxtr@do@automake@err: new ...	146
\GlsXtrLetField: new	39	\@glxtr@org@gloautosee: new	28
\GlsXtrLetFieldToField: new	39	\@glxtr@record: added third arg	7
		\@glxtr@recordsee: new	13

General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	76	\@glo@autosee: added redefinition	28
added \glsadd option thevalue	76	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	97	bug	133
\glsenableentrycount: fixed		\@glsxtr@addunusedxrefs: added	
assignment of \@cGls@	111	check for seealso field	57
\glsenableentryunitcount: fixed		\@glsxtr@checkgroup: use \csuse	
assignment of \@cGls@	119	instead of \csname	153
\glsnavigation: new	134	\@glsxtr@dorecordnodefer: new	11
\glsxtr@org@getgrouptitle: new ..	134	\@glsxtr@record@only@setup: added	
\glsxtr@recordsee: new	7	check for \@gls@setupsort@none ..	15
\glsxtr@writefields: added check for		\@print@unsrt@glossary: corrected	
automake	146	misspelt command	150
\glsxtrdisplayendloc: added check		\@printunsrt@glossary@handler:	
for empty format	136	new	151
\glsxtrgetgrouptitle: new	134	\gls@checkseeallowed: added	
\glsxtrinitwrgloss: new	71	redefinition	28
\glsxtrlocationhyperlink: new ...	137	\glsxtr@writefields: added	
\glsxtrsetgrouptitle: new	134	\providecommand lines	145
\glsxtrsupphypernumber: new	138	\glsxtrautoindex: new	192
\ifglsxtrwrglossbefore: new	71	\glsxtrautoindexassignsort: new .	193
1.15 (2017-05-10)		\glsxtrautoindexentry: new	192
\@glsxtr@dorecord: corrected		\glsxtrindexseealso: new	54
premature expansion of \@glslocref	10	\glsxtrseealsolabels: new	56
short-em-long-em: fixed spelling of		\glsxtrseelist: new	54
\glsabbrvfont	278	\glsxtruseseealso: new	53
short-long: fixed spelling of		\glsxtruseseealsoformat: new	54
\glsabbrvfont	230	\seealso name: new	54
short-long-user: fixed spelling of		autoseeindex: new	18
\glsabbrvfont	302	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@glsxtr@mark@wordseps: new	205
\glsabbrvfont	298	\@glsxtr@markwordseps: new	205
short-postlong-user-desc: fixed		\@glsxtr@noidx@displaynumberlist:	
spelling of \glsabbrvfont	300	replace hard-coded ?? with	
long-em-short-em: fixed spelling of		\glsxtrundeftag	132
\glsabbrvfont	275	\@glsxtr@noidx@entrynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	296	\glsxtrundeftag	133
long-postshort-user-desc: fixed		\@glsxtr@noidx@numberlistloop:	
spelling of \glsabbrvfont	298	replace hard-coded ?? with	
long-short: fixed spelling of		\glsxtrundeftag	133
\glsabbrvfont	228	\@glsxtrifhyphenstart: new	303
long-short-user: fixed spelling of		General: removed some inconsistencies	
\glsabbrvfont	295	in the abbreviation styles	227
footnote: fixed spelling of		\glsabbrvhyphenfont: new	304
\glsabbrvfont	232	\glsabbrvonlyfont: new	317
postfootnote: fixed spelling of		\glsabbrvscfont: new	244
\glsabbrvfont	234	\glsabbrvsmfont: new	258

\glsabbrvuserfont: initialised to default font	294	short-long-user-desc: corrected first forms	303
\glsfirstabbrvhyphenfont: new ...	304	short-nolong-desc-noreg: new ...	239
\glsfirstabbrvonlyfont: new	317	short-nolong-noreg: new	237
\glsfirstabbrvscfont: new	244	long-em-noshort-em-desc-noreg: new	290
\glsfirstabbrvsmfont: new	258	long-em-noshort-em-noreg: new ...	286
\glsfirstlonghyphenfont: new ...	304	long-hyphen-noshort-desc-noreg: new	306
\glsfirstlongonlyfont: new	317	long-hyphen-noshort-noreg: new ..	308
\glslonghyphenfont: new	304	long-hyphen-postshort-hyphen: new	310
\glslongonlyfont: new	317	long-hyphen-postshort-hyphen-desc: new	311
\glslonguserfont: initialised to default font	294	long-hyphen-short-hyphen: new ...	304
\glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong	206	long-hyphen-short-hyphen-desc: new	305
\GlsXtrDefineAcShortcuts: new ...	21	long-noshort-desc-noreg: new ...	243
\glsxtrgenabbrvfmt: added check for \ifglsxtrininsertinside	221	long-noshort-noreg: new	243
\glsxtrhyphensuffix: new	304	long-only-short-only: new	318
\glsxtrifhyphenstart: new	303	long-only-short-only-desc: new ..	319
\glsxtrlonghyphen: new	309	long-short-user-desc: corrected first forms	301
\glsxtrlonghyphennoshort: new ...	306	1.18 (2017-08-10) stylemods: changed default value to “default”	24
\glsxtrlonghyphenshort: new	304	1.19 (2017-09-09) \@glsxtr@defaultnumberformat: new .	7
\glsxtrlongshortdescname: new ...	229	\@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref	10
\glsxtronlydescname: new	319	\@glsxtr@dorecordnodefer: Use \theglsentrycounter for the location rather than \@glslocref ..	11
\glsxtronlydescsort: new	319	\@glsxtr@record@setting: new	14
\glsxtronlysuffix: new	318	\@glsxtr@record@setting@alsoindex: new	14
\glsxtrparen: new	208	\@glsxtrifhasfield: new	35
\glsxtrposthyphenlong: new	314	General: added \glslink option theHvalue	72
\glsxtrposthyphenshort: new	309	added \glslink option thevalue ...	72
\glsxtrposthyphensubsequent: new	309	\glsxtr@writefields: removed double-quotes around \jobname ..	146
\glsxtrshortdescname: new	238	\glsxtrdoautoindexname: changed format test	192
\glsxtrshorthyphen: new	314	\glsxtrhyperlink: new	96
\glsxtrshorthyphenlong: new	312	\glsxtrifhasfield: new	35
\glsxtrshortlongdescname: new ...	231	\GlsXtrSetDefaultNumberFormat: new	7
\glsxtrshortlongdescsort: new ...	231	\s@glsxtrifhasfield: new	36
\Glsxtrsubsequentfmt: new	224		
\Glsxtrsubsequentplfmt: new	224		
\glsxtrsubsequentplfmt: new	224		
\glsxtrword: new	205		
\glsxtrwordsep: new	205		
short-hyphen-long-hyphen: new ...	312		
short-hyphen-long-hyphen-desc: new	313		
short-hyphen-postlong-hyphen: new	315		
short-hyphen-postlong-hyphen-desc: new	317		

1.20 (2017-09-11)			
\@glxstrhypernameprefix: new	130	
\glsdohypertarget: added redefinition		131	
\printunsertglossaryunitsetup:			
switched from redefining			
\glolinkprefix to			
\@glxstrhypernameprefix	152	
1.21 (2017-11-03)			
\@@glxstr@record: added check for			
default options	9	
\@@glxstrwrglossmark: new	26	
\@glslink: changed \let to \def	97	
\@glxstr@checkgroup: new	152	
\@glxstr@defpostpunc: new	18	
\@glxstr@do@record@wrglossary:			
new	8	
\@glxstr@dosee@alsoindex@glossary:			
new	28	
\@glxstr@doseeglossary: new	28	
\@glxstr@noidx@do: removed code			
dealing with the group	154	
\@glxstr@record@setting@off: new	.	15	
\@glxstr@record@setting@only: new		14	
\@glxstr@rglstrigger@record: new		158	
\@glxstrglossentry: new	147	
\@glxstrnewgls: new	155	
\@glxstrsetaliasnoindex: changed to			
use \glxstrifhasfield instead of			
\ifglshasfield	92	
\@glxstrwrglossmark: new	26	
\@rGLS: new	160	
\@rGLS@: new	160	
\@rGLSpl: new	161	
\@rGLSpl@: new	161	
\@rGls: new	160	
\@rGls@: new	160	
\@rGlspl: new	160	
\@rGlspl@: new	160	
\@rgls: new	159	
\@rgls@: new	159	
\@rglspl: new	159	
\@rglspl@: new	159	
General: adjusted mcolalttree	418	
modified index to remove hard coded			
\space	398	
modified list to remove hard coded			
\space	387	
moved conditional outside of			
\glsgroupskip	390–397	
new	421	
redefined altlistgroup to discourage			
breaks after group headings	388	
redefined altlisthypergroup to			
discourage breaks after group			
headings	389	
redefined alttreegroup to discourage			
breaks after group headings	414	
redefined alttreehypergroup to			
discourage breaks after group			
headings	415	
redefined indexgroup to discourage			
breaks after group headings	399	
redefined indexhypergroup to			
discourage breaks after group			
headings	399	
redefined listgroup to discourage			
breaks after group headings	388	
redefined listhypergroup to			
discourage breaks after group			
headings	388	
redefined mcolalttreegroup to			
discourage breaks after group			
headings	419	
redefined mcolalttreehypergroup to			
discourage breaks after group			
headings	419	
redefined mcolalttreespannav to			
discourage breaks after group			
headings	420	
redefined mcolindexgroup to			
discourage breaks after group			
headings	415	
redefined mcolindexhypergroup to			
discourage breaks after group			
headings	415	
redefined mcolindexspannav to			
discourage breaks after group			
headings	416	
redefined mcoltreegroup to			
discourage breaks after group			
headings	416	
redefined mcoltreehypergroup to			
discourage breaks after group			
headings	417	
redefined mcoltreenamegroup to			
discourage breaks after group			
headings	417	
redefined			

mcoltreenonamehypergroup to	\glxstrbookindexcols: new	421
discourage breaks after group	\glxstrbookindexcolspread: new ..	423
headings	\glxstrbookindexfirstmark: new ..	427
redefined mcoltreenonamespannav to	\glxstrbookindexfirstmarkfmt: new	426
discourage breaks after group	\glxstrbookindexformatheader: new	422
headings	\glxstrbookindexgroupskip: new ..	422
redefined mcoltreepannav to	\glxstrbookindexlastmark: new ...	427
discourage breaks after group	\glxstrbookindexlastmarkfmt: new	427
headings	\glxstrbookindexmarkentry: new ..	426
redefined treegroup to discourage	\glxstrbookindexname: new	421
breaks after group headings	\glxstrbookindexparentchildsep:	
redefined treehypergroup to	new	422
discourage breaks after group	\glxstrbookindexparentschildsep:	
headings	new	422
redefined treenonamegroup to	\glxstrbookindexprelocation: new	421
discourage breaks after group	\glxstrbookindexsubatendgroup:	
headings	new	422
redefined treenonamehypergroup to	\glxstrbookindexsubbetween: new ..	422
discourage breaks after group	\glxstrbookindexsubname: new	421
headings	\glxstrbookindexsubprelocation:	
debug: new	new	421
\gglssetwidest: new	\glxstrbookindexsubsubatendgroup:	
\glsdisablehyper: added check for	new	422
existence	\glxstrbookindexsubsubbetween:	
changed to use \def rather than \let .	new	422
\glsenablehyper: changed to use \def	\glxstrbookindexthepage: new	426
rather than \let	\glxstrdetoklocation: new	157
\Glsfmtname: new	\glxtrenablerecordcount: new ...	157
\glsfmtname: new	\glxstrglossentry: new	147
\glshex: new	\glxstrgroupfield: new	152
\glslstchildpostlocation: new ..	\Glsxtrheadname: new	326
\glslstchildprelocation: new ...	\glxtrheadname: new	325
\glslstprelocation: new	\GlsXtrIfFieldEqStr: new	39
\glsnahyperlink: patched	\glxstriflabelinlist: new	151
\glssseeitemformat: new	\glxstrifrecordtrigger: new	158
\glsshowtarget: new	\glxstrindexseealso: added check	
\glstreechildprelocation: new ...	that the entry exists	54
\glstreeprelocation: new	\glxstrinithyperoutside: new	72
\glstriggerrecordformat: new	\GlsXtrLocationRecordCount: new ..	157
\glseuseabbrvfont: new	\glxstrnewgls: new	154, 156
\glseuselongfont: new	\glxstrnewGLSlike: new	156
\glxstr@do@alsoindex@wrglossary:	\glxstrnewglslike: new	156
new	\glxstrnewrgls: new	156
\glxstr@org@@do@wrglossary: new ..	\glxstrnewrGLSlike: new	156
\glxstr@org@dohyperlink: new	\glxstrnewrglslike: new	156
\glxstr@setbookindexmark: new ...	\glxstrprelocation: new	386, 421
\glxstrbookindexatendgroup: new ..	\GlsXtrRecordCount: new	157
\glxstrbookindexbetween: new	\glxstrrecordtriggervalue: new ..	157
\glxstrbookindexbookmark: new ...		

\glxtrresourcefile: now disables record key	143	\glossentrynameother: new	190
\glxtrresourceinit: new	144	\glseeitemformat: switched check from regular to short	51
\GlsXtrSetRecordCountAttribute: new	157	\glxtr@setaccessdisplay: new ...	189
\glxtrtitlename: new	326	\glxtr@writefields: provide \glxtr@record in aux file	145
\glxtrtitleorpdforheading: new .	322	\glxtractivatenopost: new	129
\GlsXtrTotalRecordCount: new	156	\glxtrbookindexprelocation: removed check for no post dot	421
\glxtrwrglossmark: new	26	\glxtrglossentryother: new	148
short-em: new	280	\glxtrnopostpunc: new	129
short-sc: corrected first letter uppercasing	249	1.23 (2017-11-12)	
short-sm: corrected first letter uppercasing	263	\@@glxtrfmt: added check for indexing	33
shortcuts: ac	23	added grouping	33
\ifglxtr@hyperoutside: new	72	new	33
all: new	385	\@glxtr@nopostpunc@postdesc: new	130
nolong-short: new	240	\@glxtr@restore@postpunc: new ..	130
nolong-short-em: new	282	\@glxtrentryfmt: fixed missing label argument	34
nolong-short-noreg: new	240	\@glxtrfmt: new	33
nolong-short-sc: new	251	\eglupdatewidest: new	405
nolong-short-sm: new	265	\gglupdatewidest: new	404
nopostdot: new	19	\glupdatewidest: new	404
postpunc: new	19	\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand	21
\printunsrtglossaryentryprocesshook: new	150, 151	\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand	21
\printunsrtglossarypredoglossary: new	151	\glxtrfmtdisplay: new	33
\printunsrtglossaryskipentry: new	150	\glxtrifcustomdiscardperiod: new	200
\rGLS: new	160	\GlsXtrIfFieldUndef: new	37
\rGls: new	159	\glxtrrestorepostpunc: new	130
\rgls: new	159	\s@glxtrfmt: new	33
\rGLSformat: new	162	\s@glxtrfmt: new	33
\rGlsformat: new	161	\xglupdatewidest: new	405
\rglsformat: new	161	1.24 (2017-11-14)	
\rGLSpl: new	161	\glsadd: added \@gls@setsort	77
\rGlspl: new	160	\glxtrforcsvfield: new	35
\rglspl: new	159	\glxtrlocalsetgrouptitle: new ..	134
\rGLSplformat: new	162	1.25 (2017-11-14)	
\rGlsplformat: new	161	\glxtrbookindexmulticolsest: new	423
\rglsplformat: new	161	1.25 (2017-11-24)	
\s@glxtrifhasfield: switched from \ifdef to \ifundef	36	\glsextrapostnamehook: new	189
1.22 (2017-11-08)		\glxtrfootnotename: new	232
\@glxtr@nopostpunc: new	129	\glxtrlongnoshortdescname: new .	241
\@glxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glxtractivatenopost	129	\glxtrlongnoshortname: new	243
\@glxtrglossentryother: new	148	\glxtrlongshortname: new	227
		\glxtrlongshortuserdescname: new	297

\glxstronlyname:new	318	\glxstrGeneralLatinVIIrules:new	364
\glxstrpostlinkAddDescOnFirstUse:		\glxstrGeneralLatinVIrules:new ..	363
changed to use \glxstrparen	200	\glxstrGeneralLatinVrules:new ..	362
\glxstrpostlinkAddSymbolOnFirstUse:		\glxstrgeneralpuncIIrules:new ..	359
changed to use \glxstrparen	201	\glxstrgeneralpuncIrules:new ...	358
\glxstrshortlongname:new	230	\glxstrgeneralpuncrules:new	358
\glxstrshortlonguserdescname:new	300	\glxstrhyphenrules:new	358
\glxstrshortnolongname:new	236	\glxstrLatinA:new	365
1.26 (2018-01-05)		\glxstrLatinAA:new	367
\@glxstr@do@inc@linkcount:new ..	162	\glxstrLatinAELigature:new	367
\glslinkpresetkeys:new	72	\glxstrLatinE:new	365
\glxstr@inc@linkcount:new	72	\glxstrLatinEszettSs:new	367
\GlsXtrEnableLinkCounting:new ..	163	\glxstrLatinEszettSz:new	367
\GlsXtrIfLinkCounterDef:new	163	\glxstrLatinEth:new	367
\glxstrinclinkcounter:new	163	\glxstrLatinH:new	365
\GlsXtrLinkCounterName:new	163	\glxstrLatinI:new	366
\GlsXtrLinkCounterValue:new	163	\glxstrLatinInsularG:new	368
\GlsXtrTheLinkCounter:new	163	\glxstrLatinK:new	366
1.27 (2018-02-26)		\glxstrLatinL:new	366
\@glxstrdialecthook:new	30	\glxstrLatinLslash:new	368
General: added		\glxstrLatinM:new	366
glossaries-extra-bib2gls.sty	340	\glxstrLatinN:new	366
\Alpha:new	352	\glxstrLatinO:new	366
\Beta:new	352	\glxstrLatinOELigature:new	367
\Chi:new	352	\glxstrLatinOslash:new	368
\Digamma:new	353	\glxstrLatinP:new	366
\Epsilon:new	352	\glxstrLatinS:new	366
\Eta:new	352	\glxstrLatinSchwa:new	367
\glxstr@loaddialect:new	340	\glxstrLatinT:new	366
\glxstrBasicDigitrules:new	382	\glxstrLatinThorn:new	367
\glxstrcombiningdiacriticIIrules:		\glxstrLatinWynn:new	368
new	356	\glxstrLatinX:new	367
\glxstrcombiningdiacriticIrules:		\glxstrMathGreekIIrules:new	374
new	356	\glxstrMathGreekIrules:new	373
\glxstrcombiningdiacriticIrules:		\glxstrMathItalicAlpha:new	378
new	355	\glxstrMathItalicBeta:new	378
\glxstrcombiningdiacriticIVrules:		\glxstrMathItalicChi:new	381
new	357	\glxstrMathItalicDelta:new	379
\glxstrcombiningdiacriticrules:		\glxstrMathItalicEpsilon:new ...	379
new	355	\glxstrMathItalicEta:new	379
\glxstrcontrolrules:new	354	\glxstrMathItalicGamma:new	379
\glxstrcurrencyrules:new	359	\glxstrMathItalicGreekIIrules:	
\glxstrdigitrules:new	382	new	370
\glxstrfractionrules:new	383	\glxstrMathItalicGreekIrules:new	369
\glxstrGeneralLatinIIIrules:new	361	\glxstrMathItalicIota:new	379
\glxstrGeneralLatinIIrules:new ..	360	\glxstrMathItalicKappa:new	380
\glxstrGeneralLatinIrules:new ..	360	\glxstrMathItalicLambda:new	380
\glxstrGeneralLatinIVrules:new ..	362	\glxstrMathItalicLowerGreekIIrules:	
\glxstrGeneralLatinVIIIrules:new	365	new	372

<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau: new</code>	378
new	371	<code>\glxtrUpTheta: new</code>	376
<code>\glxtrMathItalicMu: new</code>	380	<code>\glxtrUpUpsilon: new</code>	378
<code>\glxtrMathItalicNabla: new</code>	382	<code>\glxtrUpXi: new</code>	377
<code>\glxtrMathItalicNu: new</code>	380	<code>\glxtrUpZeta: new</code>	376
<code>\glxtrMathItalicOmega: new</code>	381	<code>\Iota: new</code>	352
<code>\glxtrMathItalicOmicron: new</code>	380	<code>\Kappa: new</code>	352
<code>\glxtrMathItalicPartial: new</code>	382	<code>\Mu: new</code>	352
<code>\glxtrMathItalicPhi: new</code>	381	<code>\Nu: new</code>	352
<code>\glxtrMathItalicPi: new</code>	380	<code>\Omicron: new</code>	352
<code>\glxtrMathItalicPsi: new</code>	381	<code>\omicron: new</code>	353
<code>\glxtrMathItalicRho: new</code>	380	<code>\Rho: new</code>	352
<code>\glxtrMathItalicSigma: new</code>	381	<code>\Tau: new</code>	352
<code>\glxtrMathItalicTau: new</code>	381	<code>\Upalpha: new</code>	353
<code>\glxtrMathItalicTheta: new</code>	379	<code>\Upbeta: new</code>	353
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upchi: new</code>	354
new	371	<code>\Upsilon: new</code>	353
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta: new</code>	353
new	370	<code>\Upiota: new</code>	353
<code>\glxtrMathItalicUpsilon: new</code>	381	<code>\Upkappa: new</code>	353
<code>\glxtrMathItalicXi: new</code>	380	<code>\Upmu: new</code>	353
<code>\glxtrMathItalicZeta: new</code>	379	<code>\Upnu: new</code>	353
<code>\glxtrMathUpGreekIrules: new</code>	369	<code>\Upomicron: new</code>	353
<code>\glxtrMathUpGreekIrules: new</code>	368	<code>\upomicron: new</code>	354
<code>\glxtrnonprintablerules: new</code>	355	<code>\Uprho: new</code>	353
<code>\glxtrprovidecommand: new</code>	341	<code>\Uptau: new</code>	353
<code>\glxtrspacerules: new</code>	355	<code>\Upzeta: new</code>	353
<code>\glxtrSubScriptDigitrules: new</code>	382	<code>\Zeta: new</code>	352
<code>\glxtrSuperScriptDigitrules: new</code>	382		
<code>\glxtrUpAlpha: new</code>	375	1.28 (2018-03-06)	
<code>\glxtrUpBeta: new</code>	375	<code>\@glxtr@docdefval</code> : changed from	
<code>\glxtrUpChi: new</code>	378	count register to macro	16
<code>\glxtrUpDelta: new</code>	375	<code>\@glxtr@dialecthook</code> : save and restore	
<code>\glxtrUpDigamma: new</code>	376	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon: new</code>	376	383
<code>\glxtrUpEta: new</code>	376	<code>\glxtr@redeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma: new</code>	375	<code>\protected@csedef</code>	38
<code>\glxtrUpIota: new</code>	376	<code>\glxtr@localsetgrouptitle</code> : changed	
<code>\glxtrUpKappa: new</code>	376	<code>\csedef \protected@csedef</code>	134
<code>\glxtrUpLambda: new</code>	377	<code>\glxtr@rsetgrouptitle</code> : changed	
<code>\glxtrUpMu: new</code>	377	<code>\csxdef \protected@csxdef</code>	134
<code>\glxtrUpNu: new</code>	377	1.29 (2018-04-09)	
<code>\glxtrUpOmega: new</code>	378	<code>\@glxtr@removespaces</code> : added expansion	137
<code>\glxtrUpOmicron: new</code>	377	<code>\@glxtr@dorecord</code> : don't suppress	
<code>\glxtrUpPhi: new</code>	378	expansion of <code>\@glxtr@recordloc</code> if	
<code>\glxtrUpPi: new</code>	377	counter isn't page	11
<code>\glxtrUpPsi: new</code>	378	<code>\@glxtr@wrglossary@locationhyperlink</code> :	
<code>\glxtrUpRho: new</code>	377	new	25
<code>\glxtrUpSigma: new</code>	377	<code>\glxtr@inc@wrglossaryctr</code> : new	25
		<code>\glxtr@wrglossarylocation</code> : new	341

\GlsXtrBibTeXEntryAliases: new .. 342	\GLSxtrshort: added
\glxtrfieldforlistloop: corrected	\@glxtr@record 215
argument order in \forlistcsloop 35	\Glsxtrshort: added
\GlsXtrIndexCounterLink: new 341	\@glxtr@record 215
\GlsXtrInternalLocationHyperlink:	\glxtrshort: added
new 25	\@glxtr@record 214
\GlsXtrProvideBibTeXFields: new . 342	\GLSxtrshortpl: added
indexcounter: new 25	\@glxtr@record 219
\setentrycounter: new 137	\Glsxtrshortpl: added
1.30 (2018-04-25)	\@glxtr@record 218
\@@glxtr@record: added check for	\glxtrshortpl: added
post-key hook 9	\@glxtr@record 218
added check for pre-key hook 9	\GlsXtrStartUnsetBuffering: new . 107
\@GLSxtr@fullpl: added	\GlsXtrStopUnsetBuffering: new .. 107
\@glxtr@record 213	indexcounter: added check for
\@GlsXtrStopUnsetBuffering: new . 107	wrglossary counter 26
\@Glsxtr@fullpl: added	\s@GlsXtrStopUnsetBuffering: new 108
\@glxtr@record 213	1.31 (2018-05-09)
\@glxtr@dorecord: don't suppress	\@GlsXtrStartUnsetBuffering: new 107
expansion of \@glxtr@recordlocref .. 11	\@glxtr@ifaccessattribute@set: new 172
\@glxtr@full: added	\@glxtr@initaccesskeys: new ... 171, 177
\@glxtr@record 211	\@glxtr@setup@default@short@access:
\@glxtr@fullpl: added	new 172, 177
\@glxtr@record 212	\@glxtr@record@noglossarywarning:
\@glxtr@glossadd@postkeys: new .. 10	new 142
\@glxtr@glossadd@prekeys: new ... 10	\@glxtr@buffer@nodup@unset: new . 107
\@glxtr@glslink@postkeys: new ... 10	General: added prefix key for glslink . 72
\@glxtr@glslink@prekeys: new 10	added prefix key for printgloss .. 131
\@glxtr@local@textformat: new ... 72	changed \let to \def 130
\@glxtr@unset: new 106	\glxaddeach: new 77
\@glxtr@buffer@unset: new 107	\glscapturedgroup: new 341
\glxtradd: added \glxtraddpostsetkeys . 76	\glxtrdefpostdesc: new 199
added \glxtraddpresetkeys 76	\glxtrdefpostlink: new 200
\glxtraddpostsetkeys: new 76	\glxtrdefpostname: new 189
\glxtraddpresetkeys: new 76	\glxtrdohypertarget: bug fix: ensure that
\glxtruserdescription: new 295	new version is picked up 131
\glxtrabbreviationfont: new 69	\glxtrlistdesc: new 387
\GlsXtrDualBackLink: new 342	\glxtrlocalreseteach: new 108
\GlsXtrDualField: new 342	\glxtrlocalunseteach: new 109
\GlsXtrExpandedFmt: new 73	\glxtrtreechilddesc: new 400
\GLSxtrlong: added \@glxtr@record 217	\glxtrtreechildsymbol: new 400
\Glsxtrlong: added \@glxtr@record 216	\glxtrtreedefaultnamefmt: new 397
\glxtrlong: added \@glxtr@record 216	\glxtrtreedesc: new 399
\GLSxtrlongpl: added	\glxtrtreegroupheaderfmt: added
\@glxtr@record 221	redefinition 398
\Glsxtrlongpl: added	\glxtrscreennamefmt: added redefinition . 397
\@glxtr@record 220	\glxtrscreenavigationfmt: added
\glxtrlongpl: added	redefinition 398
\@glxtr@record 220	\glxtrscreenonamechilddesc: new 401

\glstreenonamedesc: new	401	\@dGls: new	351
\glstreenonamesymbol: new	401	\@dGlspl: new	351
\glstreesymbol: new	400	\@dglS: new	350
\glsxtr@newabbreviation: added		\@dglSpl: new	350
\ExtraCustomAbbreviationFields		\@glS@getcounterprefix: new	29
.....	206	\@glSlongextrawidestname: new ...	430
\GlsXtrForUnsetBufferedList: new	108	\@glSxtr@bibglS@removespaces: new	345
\GlsXtrIfFieldCmpNum: new	36	\@glSxtr@check@bibglS@nameref:	
\GlsXtrIfFieldEqNum: new	36	new	143
\GlsXtrIfFieldEqXpStr: new	40	\@glSxtr@do@nameref@record: new ..	12
\GlsXtrIfFieldNonZero: new	36	\@glSxtr@get@prefixedlabel: new ..	350
\GlsXtrIfHasNonZeroChildCount:		\@glSxtr@if@record@only: new	14
new	341	\@glSxtr@ifnum@mmode: new	12
\GlsXtrIfXpFieldEqXpStr: new	40	\@glSxtr@labelprefixes: new	349
\glSxtrpostlinkAddSymbolDescOnFirstUse:		\@glSxtr@prefixlabellist: new ...	350
new	201	\@glSxtr@providenewglS: new	154
\GlsXtrRecordWarning: new	141	\@glSxtr@record@only@setup: new ..	15
\glSxtrRevertTocMarks: new	321	\@glSxtr@record@setting@nameref:	
\GlsXtrStandaloneGlossaryType:		new	14
new	147	\@glSxtr@use@equation@counter@or:	
\GlsXtrStandaloneSubEntryItem:		new	73
new	148	General: new	428
\s@GlsXtrStartUnsetBuffering: new	107	page: nameref	11
1.32 (2018-05-24)		\dGLS: new	351
\GlsXtrForeignText: new	41	\dglS: new	350
\GlsXtrForeignTextField: new	43	\dglSdisp: new	351
\GlsXtrUnknownDialectWarning: new	43	\dglSlink: new	351
1.33 (2018-07-26)		\dGLSpl: new	351
\ifglSused: added redefinition	44	\glSadd: added grouping	76
1.34 (2018-07-29)		ensure that \glSadd performs indexing	77
\glS@begindocdefs: atom	59	\glSlongextraDescAlign: new	429
\GlsXtrIfUnusedOrUndefined: new ..	30	\glSlongextraDescFmt: new	428
\glSxtrNoGlossaryWarning: added		\glSlongextraDescNameHeader: new	435
package warning	23	\glSlongextraDescNameTabularFooter:	
\if@glSxtrdocdefrestricted:		new	435
changed to allow for atom as well ...	17	\glSlongextraDescNameTabularHeader:	
docdef: atom	17	new	435
1.35 (2018-08-13)		\glSlongextraDescSymNameHeader:	
\@glS@@link@: initialise post-link hook		new	447
commands	71	\glSlongextraDescSymNameTabularFooter:	
1.36 (2018-08-18)		new	447
\glSxtrautoindexesc: new	192	\glSlongextraDescSymNameTabularHeader:	
\glSxtrdisplaysupplc: new	343	new	447
\glSxtrmultisupplcation: new ...	343	\glSlongextraGroupHeading: new ..	430
1.37 (2018-11-30)		\glSlongextraHeaderFormat: new ..	430
\@glSxtr@record: added check for		\glSlongextraLocationAlign: new ..	430
auto-add	9	\glSlongextraLocationDescNameHeader:	
\@dGLS: new	351	new	436
\@dGLSpl: new	351		

\glslongextraLocationDescNameTabularFooter:	\glslongextraNameSymDescLocationTabularHeader:
new 436	new 442
\glslongextraLocationDescNameTabularHeader:	\glslongextraNameSymDescTabularFooter:
new 436	new 441
\glslongextraLocationDescSymNameHeader:	\glslongextraNameSymDescTabularHeader:
new 448	new 441
\glslongextraLocationDescSymNameTabularFooter:	\glslongextraSetDescWidth: new .. 431
new 449	\glslongextraSetWidest: new 430
\glslongextraLocationDescSymNameTabularHeader:	\glslongextraSubDescFmt: new 429
new 448	\glslongextraSubLocationFmt: new 429
\glslongextraLocationFmt: new ... 428	\glslongextraSubNameFmt: new 429
\glslongextraLocationSymDescNameHeader:	\glslongextraSubSymbolFmt: new .. 429
new 445	\glslongextraSymbolAlign: new ... 429
\glslongextraLocationSymDescNameTabularFooter:	\glslongextraSymbolFmt: new 428
new 445	\glslongextraSymDescNameHeader:
\glslongextraLocationSymDescNameTabularHeader:	new 444
new 445	\glslongextraSymDescNameTabularFooter:
\glslongextraLocSetDescWidth: new 431	new 444
\glslongextraNameAlign: new 429	\glslongextraSymDescNameTabularHeader:
\glslongextraNameDescHeader: new 430	new 444
\glslongextraNameDescLocationHeader:	\glslongextraSymLocSetDescWidth:
new 433	new 432
\glslongextraNameDescLocationTabularFooter:	\glslongextraSymSetDescWidth: new 431
new 433	\glslongextraTabularVAlign: new . 432
\glslongextraNameDescLocationTabularHeader:	\glslongextraUpdateWidest: new .. 430
new 433	\glslongextraUpdateWidestChild:
\glslongextraNameDescSymHeader:	new 431
new 437	\glsrenewcommand: new 341
\glslongextraNameDescSymLocationHeader:	\glsseeitemformat: removed reference
new 439	to \glslabel 51
\glslongextraNameDescSymLocationTabularFooter:	\glsxtr@dblfloat: new 18
new 439	\glsxtr@do@autoadd: new 73
\glslongextraNameDescSymLocationTabularHeader:	\glsxtr@float: new 18
new 439	\glsxtr@record@nameref: new 146
\glslongextraNameDescSymTabularFooter:	\glsxtr@renewcommand: new 341
new 438	\glsxtr@writefields: provide
\glslongextraNameDescSymTabularHeader:	\glsxtr@record@nameref in aux
new 437	file 145
\glslongextraNameDescTabularFooter:	\glsxtraddlabelprefix: new 349
new 430	\GlsXtrAutoAddOnFormat: new 73
\glslongextraNameDescTabularHeader:	\glsxtrclearlabelprefixes: new .. 349
new 430	\glsxtrdisplaylocnameref: new ... 343
\glslongextraNameFmt: new 428	\glsxtrfmtexternalnameref: new .. 346
\glslongextraNameSymDescHeader:	\glsxtrfmtinternalnameref: new .. 345
new 440	\GLSXTRhiername: new 53
\glslongextraNameSymDescLocationHeader:	\GLSXtrhiername: new 53
new 442	\GlsXtrhiername: new 52
\glslongextraNameSymDescLocationTabularFooter:	\GLSXtrhiername: new 52
new 442	\glsxtrhiername: new 52

\glsxtrhiernamesep: new	53	\@GlsXtrIfXpFieldEqXpStr: new	40
\glsxtridentifyglsllike: new	154	\@glslremovespaces: changed \x to	
\glsxtrifinlabelprefixlist: new ..	349	\@glo@tmp	137
\GlsXtrLocationField: new	153	\@glsxtr@dorecord: added protection	
\glsxtrnameloclink: new	344	for fragile commands	11
\glsxtrnamerefink: new	344	General: added label key for	
\glsxtrprependlabelprefix: new ..	349	printgloss	131
\GlsXtrSetAltModifier: write modifier		\glsxtrbookindexlocation: new ...	421
to aux	95	\glsxtrbookindexsublocation: new	422
\glsxtrSetWidest: new	346	\glsxtrentryparentname: new	38
\glsxtrSetWidestFallback: new ...	348	\GlsXtrIfFieldCmpNum: added starred	
\GlsXtrStandaloneEntryName: new ..	147	version	36
\GlsXtrStandaloneEntryOther: new	149	\GlsXtrIfFieldEqNum: added starred	
\GLSxtrusefield: new	38	version	36
\Glsxtrusefield: fixed internal		\GlsXtrIfFieldEqStr: added starred	
command and added check for		form	39
\texorpdfstring	37	\GlsXtrIfFieldEqXpStr: added starred	
\ifGlsLongExtraUseTabular: new ..	432	form	40
floats: new	18	\GlsXtrIfFieldNonZero: added starred	
long-desc-name: new	435	version	36
long-desc-sym-name: new	447	\GlsXtrIfXpFieldEqXpStr: added	
long-loc-desc-name: new	436	starred form	40
long-loc-desc-sym-name: new	449	\glsxtrsetglossarylabel: new	131
long-loc-sym-desc-name: new	446	\glsxtrshortdescname: corrected to	
long-name-desc: new	432	show long form as advertised in the	
long-name-desc-loc: new	433	manual	238
long-name-desc-sym: new	438	short-desc: corrected to omit	
long-name-desc-sym-loc: new	439	description key as advertised in the	
long-name-sym-desc: new	441	manual	238
long-name-sym-desc-loc: new	442	short-em-desc: bug fix: omit description	
long-sym-desc-name: new	444	key as advertised in the manual ...	281
equations: new	18	short-sc-desc: bug fix: omit description	
1.38 (2018-12-01)		key as advertised in the manual ...	249
\glslongextraNameFmt: bug fix:		short-sm-desc: corrected to omit	
removed double param	428	description key as advertised in the	
1.39 (2019-03-22)		manual	263
\@GlsXtrIfFieldCmpNum: new	37	\s@GlsXtrIfFieldCmpNum: new	37
\@GlsXtrIfFieldEqNum: new	36	\s@GlsXtrIfFieldEqNum: new	36
\@GlsXtrIfFieldEqStr: new	39	\s@GlsXtrIfFieldEqStr: new	40
\@GlsXtrIfFieldEqXpStr: new	40	\s@GlsXtrIfFieldEqXpStr: new	40
\@GlsXtrIfFieldNonZero: new	36	\s@GlsXtrIfXpFieldEqXpStr: new ...	41

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	
\\$	341
\,	53
\.	19, 200, 397
\@	58, 143
\@cGLS@	111, 119
\@cGLSpl@	111, 119
\@cGLs@	111, 119
\@cGLspl@	111, 119
\@cgls@	111, 119
\@cglspl@	111, 119
\@do@wrglossary	8, 10, 126
\@do@wrglossary	13, 15, 16, 29, 77, 93
\@glo@assign@sortkey	131
\@glo@list	6
\@glo@type	149
\@glossarysec	422
\@glossaryseclabel	131
\@gls@expand@field	32
\@glslocalreset	108
\@glslocalunset	108
\@glsreset	108
\@glsunset	106
\@glsxtr@autoindex@escspch	194, 195
\@glsxtr@checkspch	193–196
\@glsxtr@disabledflycommand	64
\@glsxtr@org@postdescription	129
\@glsxtr@record	15, 16
\@glsxtr@recordcounter	15, 16, 146
\@glsxtrfmt	33
\@glsxtrp	101, 102
\@glsxtrpostloctag	66
\@glsxtrpreloctag	66, 67
\@glsxtrwrglossmark	8, 9, 13, 28, 29, 54, 59, 125
\@newglossaryentry@defcounters	110
\@newglossaryentry@defunitcounters	117
\@@par	403
\@ACRlong	98
\@ACRlongpl	98
\@ACRshort	98
\@ACRshortpl	98
\@Acrlong	98
\@Acrlongpl	98
\@Acrshort	98
\@Acrshortpl	98
\@GLS@	98, 113, 114, 161, 351
\@GLSdesc@	82
\@GLSpl@	98, 113, 114, 161, 351
\@GLSplural@	99
\@GLSsymbol@	83
\@GLStext@	98
\@GLSxtr@full	212
\@GLSxtr@fullpl	213
\@GLSxtr@p@acrlong@	98
\@GLSxtr@p@acrlongpl@	98
\@GLSxtr@p@acrshort@	98
\@GLSxtr@p@acrshortpl@	98
\@GLSxtr@p@long@	98
\@GLSxtr@p@longpl@	98
\@GLSxtr@p@plural@	98
\@GLSxtr@p@short@	98
\@GLSxtr@p@shortpl@	98
\@GLSxtr@p@text@	98
\@GLSxtrlong	98, 217
\@GLSxtrlongpl	98, 221
\@GLSxtrp	105, 106
\@GLSxtrshort	98, 215
\@GLSxtrshortpl	98, 219
\@Gls@	98, 113, 114, 160, 351
\@Gls@acrentryname	121
\@Gls@entry@field	38, 89, 104, 105, 190
\@Gls@entryname	121
\@GlsXtrEnableOnTheFly	61

\@GlsXtrIfFieldCmpNum	36	\@currentlabelname	131
\@GlsXtrIfFieldEqNum	36	\@dGLS	351
\@GlsXtrIfFieldEqStr	39	\@dGLSpl	351
\@GlsXtrIfFieldEqXpStr	40	\@dGls	351
\@GlsXtrIfFieldNonZero	36	\@dGlspl	351
\@GlsXtrIfXpFieldEqXpStr	40	\@dblfloat	18
\@GlsXtrStartUnsetBuffering	107	\@dglS	350
\@GlsXtrStopUnsetBuffering	107	\@dglSpl	350
\@Glspl@	98, 113, 114, 160, 351	\@disable@onlypremakeg	125
\@Glsplural@	99	\@do@auxoutstuff	138, 139
\@Glstext@	98	\@do@glS@getcounterprefix	11
\@Glsxtr	62, 64	\@do@glSsee	56, 57
\@Glsxtr@full	211	\@do@newglossaryentry	122, 208
\@Glsxtr@fullpl	213	\@do@seeglossary	15, 16, 27, 59, 125
\@Glsxtr@p@acrlong@	98	\@do@wrglossary	75, 76, 158
\@Glsxtr@p@acrlongpl@	98	\@empty	
\@Glsxtr@p@acrshort@	98		30, 77, 85–89, 130, 137, 193, 194, 211–221
\@Glsxtr@p@acrshortpl@	98	\@end@glSxtr@addunused	58
\@Glsxtr@p@long@	98	\@end@glSxtr@gettype	127, 131
\@Glsxtr@p@longpl@	98	\@end@glSxtr@usesee	51
\@Glsxtr@p@plural@	98	\@end@glSxtrifhyphenstart	303, 304
\@Glsxtr@p@short@	98	\@endfortrue	35, 189, 224, 350
\@Glsxtr@p@shortpl@	98	\@firstofone	77, 150, 184, 191, 197, 344, 345
\@Glsxtr@p@text@	98	\@firstofthree	70,
\@Glsxtrlong	98, 216		77, 85–88, 94, 211, 212, 214, 216, 218, 220
\@Glsxtrlongpl	98, 220	\@firstoftwo	78–83, 86–89, 91, 94, 123, 189,
\@Glsxtrp	104, 105		190, 202, 203, 211–213, 218–221, 321, 322
\@Glsxtrpl	63, 64	\@float	18
\@Glsxtrshort	98, 214	\@for	6, 24, 35, 58, 77, 108, 109, 121, 125,
\@Glsxtrshortpl	98, 218		128, 135, 150, 157, 163, 182, 189, 197, 350
\@acrlong	98	\@glo@alias	55, 56
\@acrlongpl	98	\@glo@assign@sortkey	127
\@acrshort	98	\@glo@autosee	28
\@acrshortpl	98	\@glo@autoseehook	56
\@addtoreset	162	\@glo@category	115
\@afterheading		\@glo@check@sortallowed	128
	388, 389, 399, 401–403, 415–418, 426	\@glo@counterprefix	10–12, 30, 137, 345
\@alt@glS@hyp@opt	94	\@glo@countunit	115
\@auxout	11–13, 59, 68, 95, 111, 120, 125,	\@glo@default@sorttype	127
	138, 139, 143, 145, 147, 154, 155, 350, 426	\@glo@desc	45
\@bibglS@restoreat	143	\@glo@descplural	45
\@cGLS	114	\@glo@group	14
\@cGLS@	111, 114, 119	\@glo@label	
\@cGLSpl	114		13, 14, 32, 51, 55–57, 89, 97, 406–412
\@cGLSpl@	111, 114, 119	\@glo@location	14
\@cGls@	111, 119	\@glo@loclist	13
\@cGlspl@	111, 119	\@glo@name	192
\@cglS@	111, 119	\@glo@no@assign@sortkey	131
\@cglSpl@	111, 119	\@glo@parent	407, 408

<code>\@glo@see</code>	50, 51, 53, 56–58	<code>\@gls@hyp@opt</code>	90,
<code>\@glo@seealso</code>	55, 56		95, 114, 155, 159–161, 210–221, 350, 351
<code>\@glo@sort</code>	192	<code>\@gls@hyp@opt@cs</code>	94
<code>\@glo@sorttype</code>	127, 135	<code>\@gls@ifaccessattribute@set</code>	172
<code>\@glo@text</code>	71	<code>\@gls@ifinlist</code>	151
<code>\@glo@thislettergrp</code>	153	<code>\@gls@increment@currcount</code>	110
<code>\@glo@thisvalue</code>	294	<code>\@gls@increment@currunitcount</code>	118
<code>\@glo@tmp</code>	30–32, 54, 89, 137	<code>\@gls@initaccesskeys</code>	206
<code>\@glo@type</code>	57, 95, 122, 125, 128, 129,	<code>\@gls@keymap</code> ..	13, 14, 31, 51, 55, 89, 145, 189
	131, 134, 135, 138, 139, 141, 142, 149, 150	<code>\@gls@label</code>	
<code>\@glo@types</code>	180, 181, 405–412		8, 9, 11, 12, 59, 94, 125, 126, 147, 224
<code>\@glossary@default@style</code> .	64, 65, 128, 420	<code>\@gls@levelchar</code>	193
<code>\@glossarystyle</code>	128, 129	<code>\@gls@link</code>	33, 69, 71, 85–89, 211–221
<code>\@gls@</code>	98, 112, 114, 159, 350	<code>\@gls@link@checkfirsthyper</code>	70, 123
<code>\@gls@@automake@immediate</code>	125	<code>\@gls@link@label</code>	74, 158
<code>\@gls@@link</code>	71	<code>\@gls@link@nocheckfirsthyper</code>	
<code>\@gls@ReturnAfterFi</code>	137, 345		69, 85–89, 211–221
<code>\@gls@actualchar</code>	193	<code>\@gls@link@opts</code>	73
<code>\@gls@adjustmode</code>	76	<code>\@gls@list</code>	135
<code>\@gls@alt@hyp@opt</code>	95	<code>\@gls@local@increment@currcount</code> ...	110
<code>\@gls@alt@hyp@opt@char</code>	94, 95	<code>\@gls@local@increment@currunitcount</code>	118
<code>\@gls@alt@hyp@opt@keys</code>	94, 95	<code>\@gls@location</code>	153, 154
<code>\@gls@automake</code>	128	<code>\@gls@loclist</code>	132, 133, 153, 154
<code>\@gls@between</code>	134, 135	<code>\@gls@long</code>	206
<code>\@gls@checkedmkidx</code>	193–196	<code>\@gls@longpl</code>	204, 206, 207
<code>\@gls@checkmkidxchars</code>	54, 192	<code>\@gls@map</code>	189
<code>\@gls@codepage</code>	139	<code>\@gls@nameaccess</code>	171, 173
<code>\@gls@counter</code>		<code>\@gls@nohyperlist</code>	46, 48
	9, 11, 12, 26, 30, 73, 74, 76, 93, 158	<code>\@gls@noidx@do</code>	135
<code>\@gls@currentlettergroup</code> ...	135, 150, 153	<code>\@gls@noidx@getgrouptitle</code>	150
<code>\@gls@declareoption</code>	5	<code>\@gls@noidx@nosanitizesort</code>	127
<code>\@gls@default@longpl</code>	206, 207	<code>\@gls@noidx@sanitizesort</code>	127
<code>\@gls@deffile</code>	59	<code>\@gls@noidx@loclist@finalsep</code>	132
<code>\@gls@doautomake</code>	128, 146	<code>\@gls@noidx@loclist@prev</code>	132
<code>\@gls@doautomake@err</code>	146	<code>\@gls@noidx@loclist@sep</code>	132
<code>\@gls@enablesavenonumberlist</code>	58, 59	<code>\@gls@noref@warn</code>	126, 135
<code>\@gls@encapchar</code>	193	<code>\@gls@org@glsnoidxdisplayloc</code> ..	132, 133
<code>\@gls@entry@count</code>	111	<code>\@gls@org@glssseeformat</code>	132, 133
<code>\@gls@entry@field</code>		<code>\@gls@preglossaryhook</code>	129, 197
	32, 37, 38, 56, 89, 103–106, 110, 148, 149	<code>\@gls@prevlevel</code>	413–415, 419, 420
<code>\@gls@entry@unitcount</code>	119, 120	<code>\@gls@quotechar</code>	193
<code>\@gls@field@font</code>	77–85	<code>\@gls@reference</code>	59, 60, 125
<code>\@gls@field@link</code>	78–85, 89, 90	<code>\@gls@restoreat</code>	58
<code>\@gls@firstaccess</code>	171, 174	<code>\@gls@saveentrycounter</code>	15, 16, 29, 74, 76, 158
<code>\@gls@get@counterprefix</code>	30	<code>\@gls@see@noindex</code>	28, 143
<code>\@gls@getcounterprefix</code>	11	<code>\@gls@setdefault@glslink@opts</code>	
<code>\@gls@getgrouptitle</code>	134, 150		9, 33, 74, 93
<code>\@gls@grptitle</code>	95, 135	<code>\@gls@setsort</code>	75, 77
		<code>\@gls@setup@default@short@access</code> ..	207

<code>\@gls@setupsort@none</code>	15	<code>\@glsstyle@listdotted</code>	386
<code>\@gls@short</code>	206, 207	<code>\@glsstyle@listgroup</code>	388
<code>\@gls@shortaccess</code>	171–174	<code>\@glsstyle@listhypergroup</code>	388
<code>\@gls@shortaccesspl</code>	171–173	<code>\@glsstyle@mcolalttree</code>	418
<code>\@gls@shortpl</code>	204, 207	<code>\@glsstyle@mcolalttreegroup</code>	419
<code>\@gls@sort</code>	153	<code>\@glsstyle@mcolalttreehypergroup</code> ..	419
<code>\@gls@textaccess</code>	171, 173	<code>\@glsstyle@mcolalttreespannav</code>	420
<code>\@gls@thisHloc</code>	29, 30	<code>\@glsstyle@mcolindexgroup</code>	415
<code>\@gls@thislabel</code>	77, 108, 109, 350, 351	<code>\@glsstyle@mcolindexhypergroup</code>	415
<code>\@gls@thisloc</code>	29, 30	<code>\@glsstyle@mcolindexspannav</code>	416
<code>\@gls@thisval</code>	189	<code>\@glsstyle@mcoltreegroup</code>	416
<code>\@gls@tmp</code>	40, 41, 135	<code>\@glsstyle@mcoltreehypergroup</code>	417
<code>\@gls@tmpb</code>	195, 196	<code>\@glsstyle@mcoltreenonamegroup</code>	417
<code>\@gls@type</code>	125, 126, 128, 224, 406–412	<code>\@glsstyle@mcoltreenonamehypergroup</code>	418
<code>\@gls@write@entrycounts</code>	111	<code>\@glsstyle@mcoltreenonamespannav</code> ..	418
<code>\@gls@write@entryunitcounts</code>	119	<code>\@glsstyle@mcoltreespannav</code>	417
<code>\@gls@write@entryunitcounts@do</code>	120	<code>\@glsstyle@tree</code>	399
<code>\@gls@writedef</code>	59	<code>\@glsstyle@treegroup</code>	400
<code>\@gls@xref</code>	13, 54, 55	<code>\@glsstyle@treehypergroup</code>	401
<code>\@glsabbrv@current@abbreviation</code> 206, 221		<code>\@glsstyle@treenoname</code>	401
<code>\@glsacronymlists</code>	121	<code>\@glsstyle@treenonamegroup</code>	402
<code>\@glsdoifexistsorwarn</code> ...	17, 185–188, 190	<code>\@glsstyle@treenonamehypergroup</code> ...	402
<code>\@glsentry</code>	59, 111, 120	<code>\@glstarget</code>	97, 130, 131
<code>\@glslink</code>	75, 95, 97	<code>\@glstext@</code>	98
<code>\@glslocalreset</code>	108	<code>\@glsunset</code>	107, 108
<code>\@glslocalunset</code>	108, 109	<code>\@glswidestname</code>	405, 413
<code>\@glslongextra@begintab</code>	432–449	<code>\@glsxtr</code>	62, 64
<code>\@glslongextrawidestname</code>	430, 431	<code>\@glsxtr@@do@wrglossary</code>	126
<code>\@glsnextpages</code>	129	<code>\@glsxtr@abbreviationsdef</code>	20, 28, 29
<code>\@glsnonextpages</code>	129	<code>\@glsxtr@accessdisplay</code>	189–191
<code>\@glsnumberformat</code>	9, 11, 12, 73, 74, 76, 92, 158, 189, 192, 345	<code>\@glsxtr@activate@initialtagging</code> ..	197, 198
<code>\@glsorder</code>	125	<code>\@glsxtr@addunitcounter</code>	115
<code>\@glspl@</code>	98, 112, 114, 159, 350	<code>\@glsxtr@addunused</code>	58
<code>\@glsplural@</code>	98	<code>\@glsxtr@addunusedxrefs</code>	57, 58
<code>\@glsplunc@token</code>	202, 203	<code>\@glsxtr@altmodifier</code>	95
<code>\@glsrecordlocref</code>	10, 11	<code>\@glsxtr@attrval</code>	75, 183–188, 190, 192
<code>\@glsshowtarget</code>	96	<code>\@glsxtr@autoindex@at</code>	192–194
<code>\@glsstyle@altlist</code>	387	<code>\@glsxtr@autoindex@doextra@esc</code>	192
<code>\@glsstyle@altlistgroup</code>	388	<code>\@glsxtr@autoindex@encap</code>	192–194
<code>\@glsstyle@altlisthypergroup</code>	389	<code>\@glsxtr@autoindex@esc</code>	193, 195, 196
<code>\@glsstyle@alttree</code>	403	<code>\@glsxtr@autoindex@escat</code>	193, 194
<code>\@glsstyle@alttreegroup</code>	414	<code>\@glsxtr@autoindex@escencap</code> ...	193, 194
<code>\@glsstyle@alttreehypergroup</code>	415	<code>\@glsxtr@autoindex@esclevel</code> ...	193, 194
<code>\@glsstyle@index</code>	398	<code>\@glsxtr@autoindex@escquote</code> ...	193, 195
<code>\@glsstyle@indexgroup</code>	399	<code>\@glsxtr@autoindex@level</code>	193, 194
<code>\@glsstyle@indexhypergroup</code>	399	<code>\@glsxtr@autoindex@setname</code>	192
<code>\@glsstyle@inline</code>	397	<code>\@glsxtr@autoindexcrossrefs</code> 15, 17, 51, 55	
<code>\@glsstyle@list</code>	387	<code>\@glsxtr@autoseeindexfalse</code>	15

\@glsxtr@autoseeindextrue	18	\@glsxtr@doaccsupp	23, 27
\@glsxtr@bibgls@removespaces	345	\@glsxtr@docdefsetting	17, 59
\@glsxtr@bookindex@atendgroup	423, 424, 426	\@glsxtr@docdefval	17, 58–60
\@glsxtr@bookindex@atsubendgroup ..	425	\@glsxtr@doccounterrecord	13
\@glsxtr@bookindex@atsubsubendgroup	425	\@glsxtr@doglossary	150
\@glsxtr@bookindex@between .	423, 424, 426	\@glsxtr@doiflabelinlist	151
\@glsxtr@bookindex@sep	423–425	\@glsxtr@doloctag	67
\@glsxtr@bookindex@subatendgroup ..	423–426	\@glsxtr@dorecord	8, 10
\@glsxtr@bookindex@subbetween .	423–425	\@glsxtr@dorecordnodefer	8, 10
\@glsxtr@bookindex@subsep	423–425	\@glsxtr@dosee@alsoindex@glossary ..	16
\@glsxtr@bookindex@subsubatendgroup	423–426	\@glsxtr@doseeglossary	16, 28
\@glsxtr@bookindex@subsubbetween ..	423–425	\@glsxtr@dostylewarn	224
\@glsxtr@bookindexgroupskip ...	423, 426	\@glsxtr@enabletagging	196
\@glsxtr@cat	109, 121, 157, 197	\@glsxtr@end@	61
\@glsxtr@check@bibgls@nameref	143	\@glsxtr@endescspch	193–196
\@glsxtr@checkgroup	150	\@glsxtr@entrycount@org@localreset	110
\@glsxtr@counterrecordhook	11–13	\@glsxtr@entrycount@org@localunset	110
\@glsxtr@csname	116–119	\@glsxtr@entrycount@org@reset	110
\@glsxtr@current@style	64, 420	\@glsxtr@entrycount@org@unset	110
\@glsxtr@currentunitcount	116–119	\@glsxtr@entryunitcount@org@localreset	119
\@glsxtr@currunitcount	118, 120	\@glsxtr@entryunitcount@org@localunset	118
\@glsxtr@debugnr	26	\@glsxtr@entryunitcount@org@reset .	118
\@glsxtr@debugval	26	\@glsxtr@entryunitcount@org@unset .	118
\@glsxtr@declareoption ...	5, 18–20, 23, 26	\@glsxtr@equationsfalse	18
\@glsxtr@defaultnoglossarywarning	23, 24	\@glsxtr@err@undefaction	7, 16
\@glsxtr@defaultnumberformat	7, 9, 74, 76, 92, 189, 192	\@glsxtr@field@linkdefs	69, 71
\@glsxtr@defpostpunc	19, 27	\@glsxtr@floatsfalse	18
\@glsxtr@deprecated@abbrstyle	253, 255, 256, 258, 267, 269, 270, 272, 285, 288, 292, 294	\@glsxtr@format@overridefalse	191
\@glsxtr@dialect	41, 42	\@glsxtr@format@overridetrue	191
\@glsxtr@disabledflycommand	64	\@glsxtr@foundinlist	203
\@glsxtr@display@loc	136	\@glsxtr@full	210
\@glsxtr@do@wrindex	94	\@glsxtr@fullpl	212
\@glsxtr@do@autoadd	73	\@glsxtr@get@prefixedlabel	350, 351
\@glsxtr@do@glsgl@disablehyperinlist ..	91	\@glsxtr@gettype	127
\@glsxtr@do@inc@linkcount	163	\@glsxtr@glossdescfont	183–185
\@glsxtr@do@nameref@record	11	\@glsxtr@glossnamefont .	185–188, 190, 191
\@glsxtr@do@record@wrglossary	8, 15	\@glsxtr@gobbleto@endescspch	195
\@glsxtr@do@redef@for@glsgl@entries	7	\@glsxtr@groupheading	150, 152, 153
\@glsxtr@do@style	25, 340	\@glsxtr@idx@displaynumberlist	126
\@glsxtr@do@titlecaps@warn	184–187, 190, 197, 198	\@glsxtr@idx@entrynumberlist	126
\@glsxtr@do@abbreviationsdef	20	\@glsxtr@if@record@only	124
		\@glsxtr@ifcsstart	61
		\@glsxtr@ifnum@mmode	73
		\@glsxtr@ifpunctoken	203
		\@glsxtr@ifunitcounter	115
		\@glsxtr@insert@dots	204
		\@glsxtr@insert@dots@next	205

<code>\@glsxtr@insertdots</code>	172, 206	<code>\@glsxtr@org@gls@</code>	69, 70
<code>\@glsxtr@label</code>	35, 58, 163, 182, 183	<code>\@glsxtr@org@glsdohypertarget</code>	131
<code>\@glsxtr@labelprefixes</code>	349, 350	<code>\@glsxtr@org@glsignore</code>	67
<code>\@glsxtr@loadstyles</code>	385, 386	<code>\@glsxtr@org@glspl@</code>	70
<code>\@glsxtr@local@textformat</code>	74, 75	<code>\@glsxtr@org@glsxtrtitlefirst</code> ..	322, 323
<code>\@glsxtr@locale</code>	41, 42	<code>\@glsxtr@org@glsxtrtitlefirstplural</code>	322, 323
<code>\@glsxtr@longnewglossaryentry</code>	45	<code>\@glsxtr@org@glsxtrtitlefull</code> ..	322, 323
<code>\@glsxtr@mark@wordseps</code>	205	<code>\@glsxtr@org@glsxtrtitlefullpl</code>	322, 324
<code>\@glsxtr@mark@wordseps@next</code>	205	<code>\@glsxtr@org@glsxtrtitlelong</code> ..	322, 323
<code>\@glsxtr@markwordseps</code>	206, 207	<code>\@glsxtr@org@glsxtrtitlelongpl</code>	322, 323
<code>\@glsxtr@mixed@assign@sortkey</code>	127	<code>\@glsxtr@org@glsxtrtitlename</code> ..	322, 323
<code>\@glsxtr@newglslike</code>	154, 155	<code>\@glsxtr@org@glsxtrtitleorpdforheading</code>	322, 323
<code>\@glsxtr@noidx@displaynumberlist</code> ..	126	<code>\@glsxtr@org@glsxtrtitleplural</code>	322, 323
<code>\@glsxtr@noidx@do</code>	152	<code>\@glsxtr@org@glsxtrtitleshort</code> ..	322, 323
<code>\@glsxtr@noidx@entrynumberlist</code>	127	<code>\@glsxtr@org@glsxtrtitleshortpl</code>	322, 323
<code>\@glsxtr@noidx@numberlistloop</code>	127	<code>\@glsxtr@org@glsxtrtitletext</code> ..	322, 323
<code>\@glsxtr@nomissinggls@textnr</code>	24	<code>\@glsxtr@org@makeglossaries</code>	124
<code>\@glsxtr@nomissinggls@textval</code>	24	<code>\@glsxtr@org@markboth</code>	321
<code>\@glsxtr@noop@recordcounter</code>	13, 16	<code>\@glsxtr@org@markright</code>	320, 321
<code>\@glsxtr@nopostpunc</code>	129	<code>\@glsxtr@org@newacronymstyle</code>	123
<code>\@glsxtr@nopostpunc@postdesc</code> ..	129, 130	<code>\@glsxtr@org@postdescription</code> ..	130, 198
<code>\@glsxtr@notfoundinlist</code>	203	<code>\@glsxtr@org@see@noindex</code>	143
<code>\@glsxtr@op@recordcounter</code>	15, 16	<code>\@glsxtr@org@setacronymstyle</code>	123
<code>\@glsxtr@optlist</code>	63	<code>\@glsxtr@org@theHvalue</code>	8, 9
<code>\@glsxtr@org@GLS@</code>	70	<code>\@glsxtr@org@unset@buffer</code>	107
<code>\@glsxtr@org@GLSpl@</code>	70	<code>\@glsxtr@org@prefix</code>	10, 11
<code>\@glsxtr@org@gls@</code>	70	<code>\@glsxtr@org@printglossary</code>	64, 130
<code>\@glsxtr@org@GLspl@</code>	70	<code>\@glsxtr@org@warndep</code>	204
<code>\@glsxtr@org@glsxtrtitlefirst</code> ..	322, 323	<code>\@glsxtr@p@acrlong@</code>	98
<code>\@glsxtr@org@glsxtrtitlefirstplural</code>	322, 323	<code>\@glsxtr@p@acrlongpl@</code>	98
<code>\@glsxtr@org@glsxtrtitlefull</code> ..	322, 324	<code>\@glsxtr@p@acrshort@</code>	98
<code>\@glsxtr@org@glsxtrtitlefullpl</code>	322, 324	<code>\@glsxtr@p@acrshortpl@</code>	98
<code>\@glsxtr@org@glsxtrtitlelong</code> ..	322, 323	<code>\@glsxtr@p@long@</code>	98
<code>\@glsxtr@org@glsxtrtitlelongpl</code>	322, 323	<code>\@glsxtr@p@longpl@</code>	98
<code>\@glsxtr@org@glsxtrtitlename</code> ..	322, 323	<code>\@glsxtr@p@plural@</code>	98
<code>\@glsxtr@org@glsxtrtitleplural</code>	322, 323	<code>\@glsxtr@p@short@</code>	98
<code>\@glsxtr@org@glsxtrtitleshort</code> ..	322, 323	<code>\@glsxtr@p@shortpl@</code>	98
<code>\@glsxtr@org@glsxtrtitleshortpl</code>	322, 323	<code>\@glsxtr@p@text@</code>	98
<code>\@glsxtr@org@glsxtrtitletext</code> ..	322, 323	<code>\@glsxtr@pagetag</code>	66, 67
<code>\@glsxtr@org@MakeUppercase</code>	322, 323	<code>\@glsxtr@pagetag</code>	66, 67
<code>\@glsxtr@org@checkfirsthyper</code> ...	91, 123	<code>\@glsxtr@prefix</code>	350
<code>\@glsxtr@org@currentfieldvalue</code>	41	<code>\@glsxtr@prevunitcount</code>	118
<code>\@glsxtr@org@delimN</code>	67	<code>\@glsxtr@printglossnr</code>	130
<code>\@glsxtr@org@delimR</code>	67	<code>\@glsxtr@printglossopts</code>	64, 127, 130
<code>\@glsxtr@org@doseeglossary</code>	28, 125	<code>\@glsxtr@printglossval</code>	130
<code>\@glsxtr@org@gloautosee</code>	28	<code>\@glsxtr@printunsrtglossaryskipentry</code>	150, 151
<code>\@glsxtr@org@glolinkprefix</code>	74, 76		

<code>\@glxstr@provide@addstoragekey</code>	32	<code>\@glxstr@wrglossnr</code>	71, 72
<code>\@glxstr@provide@storagekey</code>	31	<code>\@glxstr@wrglossval</code>	71
<code>\@glxstr@providenewgls</code>	155	<code>\@glxstrbuffer@nodup@unset</code>	107
<code>\@glxstr@record</code> .. 15, 16, 69–71, 76, 211–221		<code>\@glxstrbuffer@unset</code>	107
<code>\@glxstr@record@noglossarywarning</code> ..	15	<code>\@glxstrdialecthook</code>	340
<code>\@glxstr@record@only@setup</code>	16	<code>\@glxstrdocdeffalse</code>	60
<code>\@glxstr@record@setting</code> 8, 10, 11,		<code>\@glxstrentryfmt</code>	34
14, 15, 30, 54, 59, 60, 95, 124, 143, 145, 154		<code>\@glxstrfmt</code>	32
<code>\@glxstr@record@setting@alsoindex</code> .		<code>\@glxstrglossentry</code>	147
..... 8, 10, 54, 124		<code>\@glxstrglossentryother</code>	148
<code>\@glxstr@record@setting@nameref</code> ...		<code>\@glxstrhypernameprefix</code> 130, 131, 152	
..... 11, 15, 30, 143, 145		<code>\@glxstrifhasfield</code>	35, 39, 40
<code>\@glxstr@record@setting@off</code> .. 59, 95, 154		<code>\@glxstrifhyphenstart</code>	303
<code>\@glxstr@record@setting@only</code> 14, 30		<code>\@glxstrindexaliased</code>	92
<code>\@glxstr@recordsee</code>	15, 28, 54	<code>\@glxstrindexcrossrefsfalse</code>	17
<code>\@glxstr@redef@forglsentries</code>	7, 29	<code>\@glxstrindexcrossrefstrue</code>	17
<code>\@glxstr@redefstyles</code>	24, 25, 340	<code>\@glxstrinmark</code>	321
<code>\@glxstr@reg@glosslist</code>	125–128, 131	<code>\@glxstrlong</code>	98, 216
<code>\@glxstr@restore@postpunc</code>	129	<code>\@glxstrlongpl</code>	98, 219
<code>\@glxstr@rglstrigger@record</code> ... 159–161		<code>\@glxstrnewgls</code>	156
<code>\@glxstr@s@longnewglossaryentry</code> 45		<code>\@glxstrnewgls@inner</code>	154, 155
<code>\@glxstr@savepreloctag</code>	66, 68	<code>\@glxstrnewgls@innercsname</code>	155
<code>\@glxstr@setentrycountunsetattr</code> ... 109		<code>\@glxstrnotinmark</code>	321
<code>\@glxstr@setentryunitcountunsetattr</code> 121		<code>\@glxstrp</code>	103, 104
<code>\@glxstr@setupshortcuts</code>	22, 23, 29	<code>\@glxstrp@opt</code>	101
<code>\@glxstr@shortcutsnr</code>	22	<code>\@glxstrpl</code>	63, 64
<code>\@glxstr@shortcutsval</code>	22, 145	<code>\@glxstrpostloctag</code>	66, 68
<code>\@glxstr@swaptwo</code>	203	<code>\@glxstrpreloctag</code>	66, 68
<code>\@glxstr@tag</code>	197	<code>\@glxstrsetaliasnoindex</code>	92, 93
<code>\@glxstr@taggingcs</code>	197	<code>\@glxstrshort</code>	98, 214
<code>\@glxstr@textformat</code>	75	<code>\@glxstrshortpl</code>	98, 218
<code>\@glxstr@theHentrycounter</code>	11	<code>\@glxstrundeftag</code>	6, 30
<code>\@glxstr@theHvalue</code> ... 8–10, 72, 74–76, 158		<code>\@glxstrwrglossmark</code>	26, 27
<code>\@glxstr@theentrycounter</code>	11	<code>\@gobble</code> .. 7, 16, 18, 77, 151, 152, 205, 423–426	
<code>\@glxstr@thevalue</code> 8–10, 72, 74–76, 158		<code>\@gobbletwo</code>	204
<code>\@glxstr@thisloctag</code>	67, 68	<code>\@gtempa</code>	341
<code>\@glxstr@titlelabel</code>	133, 134, 152	<code>\@ifdefinable</code>	341
<code>\@glxstr@tmp</code>	24, 73, 136	<code>\@ifnextchar</code>	94
<code>\@glxstr@type</code>	183	<code>\@ifpackageloaded</code>	5, 20, 145,
<code>\@glxstr@unitcountlist</code>	116	164, 183, 185, 187, 189, 191, 340, 353, 384	
<code>\@glxstr@unset</code>	106–108	<code>\@ifstar</code>	31, 32,
<code>\@glxstr@unset@buffer</code>	107, 108	35, 36, 39, 40, 45–47, 61, 94, 107, 149, 196	
<code>\@glxstr@unsrt@getgroupptitle</code>	150	<code>\@ifundefined</code>	339, 341
<code>\@glxstr@use@equation@counter</code> . 9, 73, 74		<code>\@ignored@glossaries</code>	46–48
<code>\@glxstr@usesee</code>	51	<code>\@input</code>	143
<code>\@glxstr@warn@onexistsordo</code> 7, 15, 16		<code>\@input@</code>	138
<code>\@glxstr@warn@undefaction</code> 7, 15, 16		<code>\@istfilename</code>	125
<code>\@glxstr@wrglossary@locationhyperlink</code>		<code>\@makeglossary</code>	125
..... 26		<code>\@mfu@domakefirstuc</code>	197, 198

<code>\acfp</code>	21	<code>\AS</code>	21
<code>\ACL</code>	21	<code>\As</code>	20
<code>\Acl</code>	21	<code>\as</code>	20
<code>\acl</code>	21	<code>\ASP</code>	21
<code>\ACLP</code>	21	<code>\Asp</code>	20
<code>\Aclp</code>	21	<code>\asp</code>	20
<code>\aclp</code>	21	<code>\AtBeginDocument</code>	26, 30, 65, 66, 145, 350
<code>\ACP</code>	21	<code>\AtEndDocument</code>	57, 111, 119, 138, 139
<code>\Acp</code>	21		
<code>\acp</code>	21	B	
<code>\ACRfullfmt</code>	122	<code>babel package</code>	192, 194, 202
<code>\Acrfullfmt</code>	122	<code>\begin</code>	135, 140, 150, 387, 389–396, 416–420, 423, 432, 434–441, 443–449
<code>\acrfullfmt</code>	122	<code>\begingroup</code>	8, 9, 33, 73, 76, 92, 147–149, 162, 341, 344, 345, 350
<code>\ACRfullplfmt</code>	122	<code>\bgroup</code>	45, 128
<code>\Acrfullplfmt</code>	122	<code>bib2gls</code>	25, 33, 95, 143, 152, 154, 158, 159, 340, 341, 343, 346, 348, 350, 352, 451
<code>\acrfullplfmt</code>	122	<code>\bibglshrefchar</code>	143
<code>\acronymentry</code>	122	<code>\bottomrule</code>	430, 433, 435, 436, 438, 439, 441, 442, 444, 446, 447, 449
<code>\acronymfont</code>	85–89, 100, 123		
<code>\acronymname</code>	20	C	
<code>\acronymsort</code>	122	<code>\c@wrglossary</code>	26
<code>\acronymtype</code>	20, 122, 123	<code>\catcode</code>	58, 143
<code>\acrpluralsuffix</code>	122, 145	<code>category attributes:</code>	
<code>\ACS</code>	21	<code>accessinsertdots</code>	172
<code>\Acs</code>	21	<code>aposplural</code>	207
<code>\acs</code>	21	<code>discardperiod</code>	201
<code>\ACSP</code>	21	<code>entrycount</code>	106, 109–111, 120, 121
<code>\Acsp</code>	21	<code>externallocation</code>	343
<code>\acsp</code>	21	<code>firstshortaccess</code>	174
<code>\actualchar</code>	195	<code>firsttuc</code>	187
<code>\addtolength</code>	414	<code>glossdesc</code>	183
<code>\advance</code>	111, 120, 144, 155	<code>glossdescfont</code>	183
<code>\AF</code>	21	<code>glossname</code>	185
<code>\Af</code>	21	<code>glossnamefont</code>	185, 187
<code>\af</code>	20	<code>headuc</code>	324
<code>\AFP</code>	21	<code>indexname</code>	192
<code>\Afp</code>	21	<code>indexonlyfirst</code>	93
<code>\afp</code>	20	<code>insertdots</code>	206
<code>\AL</code>	21	<code>linkcount</code>	162
<code>\Al</code>	21	<code>linkcountmaster</code>	162
<code>\al</code>	20	<code>markshortwords</code>	206
<code>\ALP</code>	21	<code>markwords</code>	206, 207, 303, 304, 312
<code>\Alp</code>	21	<code>nameshortaccess</code>	173
<code>\alp</code>	20	<code>nohyper</code>	91
<code>amsmath package</code>	12, 26	<code>nohyperfirst</code>	78–80
<code>\AnyTrackedLanguages</code>	340, 384	<code>noshortplural</code>	207
<code>\appto</code>	13, 14, 25, 31, 50, 51, 54–57, 89, 94, 110, 117, 150, 152, 191, 202, 205, 349, 385		
<code>\apptoglossarypreamble</code>	346–348		
<code>\arabic</code>	26, 426		

regular	68, 115, 227–232, 234, 237, 239, 240, 242–244, 246, 247, 249, 250, 253–255, 257, 260–264, 267–269, 271, 274–280, 282, 284, 286, 288–290, 292, 295, 301–303, 305–308, 313, 314, 318, 320
textformat	75
textshortaccess	173
\cdot	26
\centering	422
\cGLS	21, 109, 121
\cGls	20, 21, 109, 121
\cglS	20, 21, 109, 121
\cGLSformat	113
\cGlsformat	113
\cglSformat	112, 114
\cGLSpl	21, 109, 121
\cGlspl	20, 21, 109, 121
\cglSpl	20, 21, 109, 121
\cGLSplformat	113
\cGlsplformat	113
\cglSplformat	112, 114
\changes	326
\char	133
\columnwidth	65, 66
\count@	111, 120
\csappto	44
\csdef	31, 38, 44, 89, 90, 111, 116, 117, 119, 178, 189, 199, 200, 225, 226, 234, 257, 271, 292, 296, 298–300, 310, 312, 315, 317, 386, 404
\cseappto	49
\csedef	117
\csgdef	39, 46–48, 60, 67, 110, 111, 116, 119, 404, 426
\cslet	39, 45, 128
\csletcs	39, 226
\csname	6, 32, 47, 54, 59, 64, 68, 71, 74, 76, 85–90, 93, 101, 116, 117, 125, 126, 135, 138, 141, 142, 150, 155–158, 163, 183, 204, 211–221, 227, 413
\cspreto	44
\csuse	9, 12, 33, 34, 38, 47, 57, 67, 89, 90, 103–105, 115–119, 128, 133, 135, 136, 147, 148, 151–153, 178, 189, 199, 200, 225, 226, 345, 346, 348, 404, 405, 407, 408, 430
\csxdef	51, 55, 56, 116, 119
\currentglossary	129, 147, 148, 426
\CurrentOption	27, 385, 386
\CurrentTrackedLanguage	383
\CurrentTrackedLanguageTag	145
\CurrentTrackedScript	383, 384
\CurrentTrackedTag	340, 384
\CustomAbbreviationFields	208, 228–232, 234, 236, 238, 241, 243–249, 251, 255, 256, 259–262, 264, 266, 269, 270, 273, 274, 276–281, 283, 285, 288, 290, 292, 295, 296, 298, 300, 301, 303, 305, 306, 308, 310, 311, 313, 315, 317–319
D	
datatool-base package	12
\DeclareAcronymList	122
\DeclareOption	5, 385
\DeclareOptionX	5, 27
\def	9, 10, 13–17, 27, 30, 33, 37, 45, 47, 51, 54, 55, 58, 61–64, 66, 68–76, 78–89, 95, 97–101, 107, 112–114, 122, 126–128, 130–132, 134–138, 150, 152, 153, 155, 158–161, 171–173, 193–198, 202–207, 210–221, 224, 304, 306, 309, 312, 314, 315, 343, 345, 349, 350, 384, 398, 413–415, 419, 420, 423, 424, 430, 431
\defglSentryfmt	46–48
\define@boolkey	17, 18, 72, 92
\define@choicekey	7, 15, 17, 19, 22, 24, 26, 71, 130
\define@key	13, 14, 19, 24, 25, 31, 55, 72, 76, 89, 130, 131, 171, 204
\DefineAcronymSynonyms	22
\delimN	67
\delimR	67
\descriptionname	430, 433, 435, 436, 438, 439, 441, 442, 444, 445, 447, 449
\detokenize	61
\dimen@	124, 404–412, 430, 431
\dimen@i	407, 408
\dimen@ii	404, 405, 407, 408, 430
\dimexpr	65, 66, 403, 431, 432
\disable@keys	20, 30, 60, 143
\do	6, 24, 35, 58, 77, 108, 109, 121, 125, 128, 135, 150, 157, 163, 182, 189, 197, 350
\do@glS@link@checkfirsthyper	33, 69–71, 74, 85–89, 211–221
\do@glSdisablehyperinlist	74, 92
doc package	195
\dolistcsloop	35
\DTLifinlist	73, 125–127, 131, 350
\DTLifint	133

E	
<code>\eappto</code>	13, 24, 46–48, 150, 153, 172–174, 192, 385
<code>\edef</code>	6, 8, 9, 11, 42, 46–49, 54, 56–59, 73–76, 91, 93, 95, 97, 115–119, 125, 126, 131, 133, 134, 136–139, 143, 147, 148, 158, 162, 183–186, 188–190, 193, 195, 196, 204, 345, 350, 383, 407, 408, 423–425, 432, 434–441, 443–449
<code>\eglssetwidest</code>	406–412
<code>\egroup</code>	45, 46, 129
<code>\else</code>	8–15, 17–20, 22–24, 27, 28, 30, 33, 37, 43, 44, 59–61, 66, 68, 71, 75, 76, 92, 93, 112, 124, 127–130, 134, 136, 137, 140, 142, 144, 158, 159, 191–193, 195, 196, 203, 205, 207, 214–221, 223, 224, 228, 230, 231, 233–242, 245, 247–261, 263–275, 277–293, 296, 297, 299, 300, 302–304, 306, 309, 311, 312, 315, 316, 318, 319, 345–348, 387, 390–400, 402, 413, 414, 420, 422–425, 432–450
<code>\emph</code>	272, 273
<code>\empty</code>	136, 137, 345, 349, 350
<code>\encapchar</code>	195
<code>\end</code>	135, 140, 141, 150, 387, 389–396, 416–420, 424, 432–438, 440–446, 448, 449
<code>\end@getprefix</code>	30
<code>\end@glxtr@display@loc</code>	136
<code>\endcsname</code>	6, 32, 47, 54, 59, 64, 68, 71, 74, 76, 85–90, 93, 101, 116, 117, 125, 126, 135, 138, 141, 142, 150, 155–158, 163, 183, 204, 211–221, 227, 413
<code>\endfoot</code>	430, 433, 435–437, 439, 441, 442, 444, 445, 447, 448
<code>\endgroup</code>	8, 10, 33, 73, 77, 93, 147, 149, 162, 341, 344, 345, 350
<code>\endhead</code>	430, 433, 435–437, 439, 441, 442, 444, 445, 447, 448
<code>\ensuremath</code>	26
entry categories:	
abbreviation	221
general	177, 179
index	181
<code>\entryname</code>	430, 431, 433, 435, 436, 438, 439, 441, 442, 444, 445, 447, 449
<code>\epreto</code>	192
<code>\equal</code>	141, 142, 200
equation (counter)	18, 73
<code>\escapechar</code>	341
etoolbox package	5
<code>\expandafter</code> ..	27, 32, 33, 35, 41, 51, 53, 57, 58, 61–63, 73, 89, 90, 94, 101, 107, 114, 115, 125–127, 131, 135–137, 150, 153, 155, 162, 172, 173, 183, 186, 187, 189, 191, 192, 195, 203, 206, 207, 303, 341, 345
<code>\expandonce</code>	122, 153, 172–174, 193, 229, 307, 432–438, 440–449
<code>\ExtraCustomAbbreviationFields</code>	172–174, 206, 208
F	
<code>\fi</code>	7–13, 15–20, 22–24, 27, 28, 30, 33, 37, 43, 44, 51, 54, 55, 57, 59–61, 65, 66, 68, 71, 72, 74–76, 93, 112, 119, 120, 124, 127, 128, 130, 131, 134, 136, 137, 139, 140, 142, 144–146, 158, 159, 191–194, 196, 203, 205, 207, 214–221, 223, 224, 228, 230, 231, 233–242, 245, 247–261, 263–275, 277–293, 295–297, 299, 300, 302–304, 306, 309, 311, 312, 315, 316, 318, 319, 345–349, 387, 390–400, 402, 404–414, 420, 422–425, 431, 433, 434, 436–440, 442–450
first use	451
flag	451
text	451
<code>\firstacronymfont</code>	123, 124
fontspec package	145
<code>\footnote</code>	232
<code>\forall glossaries</code> ..	57, 149, 181, 183, 406–412
<code>\forall glsentries</code>	59, 111, 120
<code>\ForEachTrackedDialect</code>	340, 384
<code>\foreignlanguage</code>	41, 42
<code>\forall glsentries</code>	6, 57, 181, 183, 406–412
<code>\forall listcsloop</code>	35, 120, 135
<code>\forall listloop</code>	107, 108, 132, 198
<code>\futurelet</code>	202
G	
<code>\gdef</code>	67, 194, 195
<code>\Genacrfullformat</code>	122
<code>\genacrfullformat</code>	122
<code>\GenericAcronymFields</code>	122
<code>\Genplacrfullformat</code>	122
<code>\genplacrfullformat</code>	122
<code>\GetTrackedDialectFromLanguageTag</code> ..	41
<code>\GetTrackedDialectToMapping</code>	42
<code>\glo@grabfirst</code>	153
<code>\glo@name</code>	186, 187, 191

<code>\gloaliaslabel</code>	96	<code>mcoltreenonamespannav</code>	418
<code>\global</code>	10, 45, 59, 129, 153	<code>mcoltreespannav</code>	417
<code>\glolinkprefix</code> ...	72, 74–76, 96, 97, 131, 145	<code>name-desc</code>	435
<code>glossaries package</code>		<code>sublistdotted</code>	387
	12, 15, 27–29, 43, 50, 54–56, 125, 128, 386	<code>tree</code>	399
<code>glossaries-accsupp package</code>	23, 27, 164, 207	<code>treegroup</code>	400
<code>glossaries-extra package</code>	2, 384	<code>treehypergroup</code>	401
<code>glossaries-extra-bib2gls package</code>		<code>treenoname</code>	401
	15, 16, 30, 340, 384, 431	<code>treenonamegroup</code>	402
<code>glossaries-extra-stylemods package</code>		<code>treenonamehypergroup</code>	402
	24, 199, 340, 347	<code>glossary-bookindex package</code>	386
<code>glossaries-prefix package</code>	350	<code>glossary-hypernav package</code>	95
<code>glossaries-stylemods package</code>	421	<code>glossary-long package</code>	391
<code>glossaries.sty package</code>	45	<code>glossary-longbooktabs package</code>	391
<code>\GlossariesExtraWarning</code>		<code>glossary-longextra package</code>	346–348
	6, 16, 18, 23, 30, 42–44,	<code>glossary-tree package</code>	347, 348, 398
	61, 63, 75, 123, 126, 136, 140, 143, 149,	<code>\glossaryentrynumbers</code> 68, 128, 129, 153, 154	
	183–186, 188, 190, 197, 226, 341, 348, 349	<code>\glossaryheader</code>	
<code>\GlossariesExtraWarningNoLine</code> 18, 111, 120			135, 150, 387–396, 398–402, 413,
<code>\GlossariesWarning</code> 67, 126, 128, 132, 133, 224			415–419, 424, 432–438, 440–446, 448, 449
<code>\GlossariesWarningNoLine</code>	126, 139	<code>\glossaryname</code>	128
<code>glossary styles:</code>		<code>\glossarypostamble</code>	135, 150, 152
<code>altlist</code>	387	<code>\glossarypreamble</code>	135, 149
<code>altlistgroup</code>	388	<code>\glossarysection</code> ...	135, 141, 143, 149, 152
<code>altlisthypergroup</code>	389	<code>\glossarytitle</code> ...	47, 128, 135, 141, 143, 149
<code>alttree</code>	346, 347, 403, 404, 413, 430	<code>\glossarytoctitle</code>	
<code>alttreegroup</code>	414		47, 128, 131, 135, 141, 143, 149
<code>alttreehypergroup</code>	415	<code>\glossentry</code>	129, 154, 386–396,
<code>index</code>	398		398, 400, 402, 413, 424, 433, 434, 436,
<code>indexgroup</code>	399		437, 439, 440, 442, 443, 445, 446, 448, 450
<code>indexhypergroup</code>	399	<code>\glossentrydesc</code>	
<code>inline</code>	397		386, 387, 389–397, 399–401, 403, 428
<code>list</code>	387	<code>\glossentryname</code>	147,
<code>listdotted</code>	386		386–396, 398, 400, 402, 413, 414, 421, 428
<code>listdottedstyle</code>	387	<code>\glossentrynameother</code>	149
<code>listgroup</code>	388	<code>\glossentrysymbol</code> 390–397, 400, 401, 403, 428	
<code>listhypergroup</code>	388	<code>\glossxtrsetpopts</code>	199
<code>longragged-booktabs</code>	432	<code>\GLS</code>	109, 121, 157
<code>mcolalttree</code>	418	<code>\Gls</code>	62, 109, 121, 157
<code>mcolalttreegroup</code>	419	<code>\gls</code>	43, 44, 62, 64, 109, 121, 126, 140, 157
<code>mcolalttreehypergroup</code>	419	<code>\gls@assign@desc</code>	45
<code>mcolalttreespannav</code>	420	<code>\gls@assign@field</code>	13, 14, 32, 89
<code>mcolindexgroup</code>	415	<code>\gls@checkseeallowed</code>	59, 61, 125
<code>mcolindexhypergroup</code>	415	<code>\gls@codepage</code>	139
<code>mcolindexspannav</code>	416	<code>\gls@defdocnewglossaryentry</code> .	59, 110, 117
<code>mcoltreegroup</code>	416	<code>\gls@defglossaryentry</code> ...	45, 46, 59, 62, 63
<code>mcoltreehypergroup</code>	417	<code>\gls@dotocitle</code>	128, 129
<code>mcoltreenonamegroup</code>	417	<code>\gls@glossary</code>	54
<code>mcoltreenonamehypergroup</code>	418	<code>\gls@grplabel</code>	95

<code>\gls@ifnotmeasuring</code>	12, 108, 109	273, 275, 277, 279–293, 295, 297, 299,
<code>\gls@level</code>	153	300, 302, 305, 307, 308, 311, 313, 318, 319
<code>\gls@noidxglossary</code>	126	<code>\Glsaccesslongpl</code>
<code>\gls@org@glossaryentryfield</code>	129	.. 88, 209, 220, 228, 237, 242, 245, 251,
<code>\gls@org@glossarysubentryfield</code>	129	252, 254, 260, 265–268, 274, 275, 283–
<code>\gls@org@TrackLangRequireDialectPrefix</code>		289, 296, 297, 305, 307, 308, 311, 313, 319
.....	383, 384	<code>\glsaccesslongpl</code>
<code>\gls@save@numberlist</code>	66, 68	88, 89, 209,
<code>\gls@set@xr@key</code>	55	220, 221, 228, 231, 233–237, 239–242,
<code>\gls@tmplen</code>	406–412, 414, 431	245, 247–256, 258, 259, 261, 263–270,
<code>\gls@type</code>	125	272, 273, 275, 277, 279–293, 296, 297,
<code>\glsabbrvdefaultfont</code> ...	210, 228, 230,	300, 302, 305, 307, 308, 311, 313, 318, 319
233, 235, 236, 238, 241, 294, 304, 307, 317		<code>\GlsAccessname</code>
<code>\glsabbrvfont</code>		53, 81
....	272–281, 283, 285, 287, 288, 290–293	<code>\Glsaccessname</code>
<code>\glsabbrvfont</code>		52, 81
.. 99, 100, 123, 210, 214, 215, 218, 219,		<code>\glsaccessname</code>
221, 223, 224, 227–236, 238, 241, 243,		51–53, 80
245, 247, 248, 250, 252, 253, 255, 257,		<code>\GlsAccessplural</code>
259, 261, 262, 264, 266, 267, 269, 271,		79
273, 275, 277, 278, 280, 281, 283, 285,		<code>\Glsaccessplural</code>
287, 288, 291, 293, 295, 297, 299, 301–		79
303, 305, 307, 309–311, 313, 315, 316, 318		<code>\GlsAccessshort</code>
<code>\glsabbrvhyphenfont</code>		53
.....	304–306, 310, 311, 313–315, 317	<code>\Glsaccessshort</code> ...
<code>\glsabbrvonlyfont</code>	317–320	52, 85, 215, 224, 231,
<code>\glsabbrvscfont</code> .	244–250, 252, 253, 255–257	233–235, 239, 240, 247, 249, 250, 256–
<code>\glsabbrvsmfont</code>		258, 261, 263, 264, 270–272, 277, 279,
.....	258–262, 264, 266, 267, 269, 271	280, 282, 291–293, 299, 300, 302, 310, 316
<code>\glsabbrvuserfont</code>	294–300, 302	<code>\glsaccessshort</code>
<code>\GlsAccessdesc</code>	81	52, 53, 85, 86, 208, 209, 214,
<code>\Glsaccessdesc</code>	81, 184, 196	215, 223, 228, 230, 233, 235–240, 242,
<code>\glsaccessdesc</code>	81, 184, 200, 201	245, 247–252, 254–259, 261–273, 275,
<code>\GlsAccessdescplural</code>	82	277, 278, 280–289, 291, 293, 296, 297,
<code>\Glsaccessdescplural</code>	81	299, 302, 305, 307, 308, 310, 313, 316, 319
<code>\glsaccessdescplural</code>	81	<code>\Glsaccessshortpl</code> 86, 218, 224, 231, 233–
<code>\GlsAccessfirst</code>	79	236, 239, 240, 247, 249, 250, 256, 258,
<code>\Glsaccessfirst</code>	79	261, 263, 264, 270, 272, 277, 279, 280,
<code>\glsaccessfirst</code>	78	282, 291–293, 299, 300, 302, 311, 316, 319
<code>\GlsAccessfirstplural</code>	80	<code>\glsaccessshortpl</code>
<code>\Glsaccessfirstplural</code>	80	86, 87, 209, 218, 219, 224, 228–
<code>\glsaccessfirstplural</code>	80	230, 233, 235–240, 242, 245, 247–252,
<code>\GlsAccesslong</code>	87, 209, 217,	254–261, 263–268, 270–275, 277, 279–
228, 237, 241, 242, 245, 251, 252, 254,		284, 286–289, 291, 293, 296, 297, 299,
259, 265–268, 273, 275, 283–287, 289,		300, 302, 305, 307, 308, 310, 313, 316, 319
296, 297, 305, 307, 308, 311, 313, 318, 319		<code>\GlsAccesssymbol</code>
<code>\glsaccesslong</code>	87, 88, 208, 216, 217,	82
228, 230, 231, 233–236, 238–242, 245,		<code>\Glsaccesssymbol</code>
247–256, 258, 259, 261, 263–270, 272,		82, 196
		<code>\glsaccesssymbol</code>
		82, 196, 201
		<code>\GlsAccesssymbolplural</code>
		83
		<code>\Glsaccesssymbolplural</code>
		82
		<code>\glsaccesssymbolplural</code>
		82
		<code>\GlsAccessstext</code>
		78
		<code>\Glsaccessstext</code>
		78
		<code>\glsaccessstext</code>
		51, 78
		<code>\glsacrshortcutstrue</code>
		22, 23
		<code>\glsacspacemax</code>
		124

<code>\glsadd</code>	33, 58, 73, 77, 140	<code>\glsentrydesc</code>	168, 176, 185
<code>\glsadd options</code>		<code>\Glsentrydescplural</code>	169, 176
<code>theHvalue</code>	9	<code>\glsentrydescplural</code>	168, 169, 176
<code>thevalue</code>	9, 10	<code>\Glsentryfirst</code>	115, 161, 166, 175
<code>\glsaddpostsetkeys</code>	10, 76	<code>\glsentryfirst</code>	115, 161, 166, 175, 336
<code>\glsaddpresetkeys</code>	10, 76	<code>\Glsentryfirstplural</code> ...	115, 162, 166, 175
<code>\glsaddstoragekey</code>	56, 177, 342, 343	<code>\glsentryfirstplural</code>	
<code>\glsbackslash</code>	61	115, 161, 166, 167, 175, 336, 337
<code>\glscapscase</code>	70, 77–90, 211–223	<code>\glsentryfmt</code>	46–48
<code>\glscategory</code> .	68, 77, 91, 99, 100, 178–180,	<code>\Glsentryfull</code>	122
	183–190, 196, 199, 200, 211–215, 218, 219	<code>\glsentryfull</code>	122
<code>\glscategorylabel</code>		<code>\Glsentryfullpl</code>	122
..	91, 172–174, 204, 206, 207, 234, 257,	<code>\glsentryfullpl</code>	122
	271, 292, 296, 298–300, 310, 312, 315, 317	<code>\glsentryitem</code>	147,
<code>\glsclosebrace</code>	54, 141, 142		148, 386–396, 398, 400, 402, 413, 424, 428
<code>\glscounter</code>	9, 18	<code>\Glsentrylong</code> ...	100, 101, 115, 161, 170, 177
<code>\glscurrententrylabel</code>	66–	<code>\glsentrylong</code> ...	100, 101, 115, 161, 170,
	68, 129, 138, 147, 148, 150, 151, 198, 199		177, 234, 257, 271, 292, 299, 300, 315, 337
<code>\glscurrentfieldvalue</code>		<code>\Glsentrylongpl</code>	100, 101, 115, 171, 177
.....	33–37, 39–41, 52, 53, 294, 341, 342	<code>\glsentrylongpl</code> .	100, 101, 115, 171, 177, 338
<code>\glscustomtext</code> ..	69–71, 85–89, 211–221, 223	<code>\Glsentrylongplural</code>	162
<code>\glsdefaulttype</code>		<code>\glsentrylongplural</code>	161
.....	6, 20, 44, 127, 128, 139, 149, 151	<code>\Glsentryname</code>	164, 174, 185, 187, 188
<code>\glsdescriptionaccessdisplay</code> ..	168, 184	<code>\glsentryname</code>	
<code>\glsdescriptionpluralaccessdisplay</code>			147, 164, 174, 192, 334, 406–412, 426, 427
.....	168, 169	<code>\glsentrynumberlist</code>	126, 133, 410–412
<code>\glsdescwidth</code>	389–396, 429, 431, 432	<code>\Glsentryplural</code>	165, 175
<code>\glsdetoklabel</code> .	8, 9, 31, 34–39, 43, 45, 49–	<code>\glsentryplural</code>	165, 174, 175, 335, 336
	51, 53, 57–61, 73, 76, 93, 96, 110, 111,	<code>\glsentryprevcount</code>	110, 112, 118
	116–120, 125, 129, 132, 133, 147, 148,	<code>\glsentryprevmaxcount</code>	118
	153, 156–158, 183, 186, 187, 191, 407, 408	<code>\glsentryprevtotalcount</code>	118
<code>\glsdisp</code>	351	<code>\Glsentryshort</code>	99, 100, 169, 176
<code>\glsdisplaynumberlist</code>	126, 132	<code>\glsentryshort</code>	99,
<code>\glsdohyperlink</code>	95, 97, 345		100, 124, 169, 176, 296, 298, 309, 333, 334
<code>\glsdohypertarget</code>	97, 130	<code>\Glsentryshortpl</code>	99, 100, 170, 176
<code>\glsdoifexists</code>	17, 28,	<code>\glsentryshortpl</code>	
	38, 45, 49, 51–54, 69, 70, 76, 85–89, 97,	99, 100, 170, 176, 177, 333, 334
	108, 109, 125, 132, 133, 147, 148, 211–221	<code>\Glsentrysymbol</code>	167, 175
<code>\glsdoifexistsordo</code>	33, 34, 71	<code>\glsentrysymbol</code>	167, 175, 409–411
<code>\glsdoifexistsorwarn</code>	17, 183, 184, 196	<code>\Glsentrysymbolplural</code>	168, 175
<code>\glsdoifnoexists</code>	45	<code>\glsentrysymbolplural</code> ..	167, 168, 175, 176
<code>\glsdonohyperlink</code>	75, 97	<code>\Glsentrytext</code>	165, 174
<code>\glsdosanitizesort</code>	127	<code>\glsentrytext</code>	97, 164, 165, 174, 335
<code>\glsenableentrycount</code>	109, 111, 119	<code>\glsentrytype</code>	148
<code>\glsenableentryunitcount</code> ...	111, 120, 121	<code>\Glsentryuseri</code>	83
<code>\glsentrycounter</code>	26, 137, 345	<code>\glsentryuseri</code>	83
<code>\GlsEntryCounterLabelPrefix</code>	43	<code>\Glsentryuserii</code>	83
<code>\glsentrycurrcount</code>	110, 111, 118	<code>\glsentryuserii</code>	83
<code>\Glsentrydesc</code>	168, 176, 184	<code>\Glsentryuseriii</code>	83

<code>\glsentryuseriii</code>	84	<code>\Glsfirstplural</code>	329
<code>\Glsentryuseriv</code>	84	<code>\glsfirstplural</code>	329
<code>\glsentryuseriv</code>	84	<code>\glsfirstpluralaccessdisplay</code> ..	166, 167
<code>\Glsentryuserv</code>	84	<code>\glsforeachincategory</code>	224
<code>\glsentryuserv</code>	84	<code>\glsgenentryfmt</code>	68, 69
<code>\Glsentryuservi</code>	84	<code>\glsgetattribute</code>	75, 96, 112, 116– 118, 138, 158, 162, 183–186, 188, 190, 192
<code>\glsentryuservi</code>	85	<code>\glsgetcategoryattribute</code>	178
<code>\glsextrapostnamehook</code>	189	<code>\glsgetgrouptitle</code>	388, 389, 399, 401–403, 414–420
<code>\glsfieldfetch</code>	96	<code>\glsgetwidestname</code>	404
<code>\glsfieldxdef</code>	182, 183	<code>\glsgroupheading</code>	153, 387–396, 398–403, 413–420, 425, 433, 434, 436, 437, 439, 440, 442, 443, 445, 446, 448, 450
<code>\glsFindWidestLevelTwo</code>	348	<code>\glsgroupskip</code>	153, 387, 389–397, 399, 400, 402, 414, 425, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447, 448, 450
<code>\glsFindWidestTopLevelName</code>	348	<code>\glshasattribute</code>	75, 95, 96, 111, 112, 116– 120, 138, 158, 162, 183–188, 190, 192, 228–232, 234, 238–240, 243–246, 248, 255, 257, 259–262, 269, 271, 273–276, 278, 279, 286, 290–292, 295, 296, 298– 303, 305–308, 310, 312–315, 317, 318, 320
<code>\GLSfirst</code>	328	<code>\glshascategoryattribute</code>	179
<code>\Glsfirst</code>	328, 329	<code>\glshex</code> ..	354–360, 362–368, 370–373, 375–383
<code>\glsfirst</code>	328	<code>\glshyperlink</code>	97, 342
<code>\glsfirstabbrvdefaultfont</code>	210, 228, 230, 233, 235, 236, 238, 241, 307	<code>\glshypernavsep</code>	135
<code>\glsfirstabbrvfont</code>	273–293	<code>\glshypernumber</code>	138, 191, 343–345
<code>\glsfirstabbrvfont</code>	123, 208, 209, 228–242, 245, 247, 248, 250, 252, 253, 255, 257, 259, 261, 262, 264, 266, 267, 269, 271, 273, 275, 277, 278, 280, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 302, 305, 307, 308, 310, 313, 315, 318	<code>\glsifattribute</code>	71, 72, 78, 91, 93, 103, 162, 181, 184–187, 190, 191, 198, 201, 202, 324–332
<code>\glsfirstabbrvhyphenfont</code>	304–306, 309, 310, 312–317	<code>\glsifcategory</code>	181
<code>\glsfirstabbrvonlyfont</code>	318, 319	<code>\glsifcategoryattribute</code>	91, 172–174, 179, 180, 206, 207
<code>\glsfirstabbrvscfont</code>	244–258	<code>\glsifnotregular</code>	77
<code>\glsfirstabbrvsmfont</code>	259–272	<code>\glsifnotregularcategory</code>	180
<code>\glsfirstabbrvuserfont</code>	295–303	<code>\glsifplural</code>	70, 77, 79–83, 85–89, 201, 202, 211–222
<code>\glsfirstaccessdisplay</code>	166	<code>\glsifregular</code>	68, 77, 115, 161, 162
<code>\glsfirstlongdefaultfont</code>	228, 230, 236, 238, 241, 244–254, 259– 269, 273, 274, 276, 277, 280–284, 287, 288	<code>\glsifregularcategory</code>	180
<code>\glsfirstlongemfont</code>	274–276, 278, 279, 285, 286, 288–290	<code>\glsifusetranslator</code>	47
<code>\glsfirstlongfont</code> ...	208, 209, 228–231, 233, 235–243, 245, 247, 248, 250, 252, 253, 255, 257, 259, 261, 262, 264, 266, 267, 269, 271, 273, 275, 277, 278, 280, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 302, 305, 307, 310, 313, 315, 318	<code>\glsignore</code>	67
<code>\glsfirstlongfootnotefont</code>	232–236, 255–258, 269–272, 290–293	<code>\glsinlinedescformat</code>	397
<code>\glsfirstlonghyphenfont</code>	304–315	<code>\glsinlinesubdescformat</code>	397
<code>\glsfirstlongonlyfont</code>	318, 319	<code>\glsinsert</code>	71, 77, 85–89, 211–223, 303, 310, 312, 315, 317
<code>\glsfirstlonguserfont</code>	295–303	<code>\glskeylisttok</code>	122, 206, 208
<code>\GLSfirstplural</code>	329	<code>\glslabel</code>	8, 9, 33, 49, 68, 69, 71–75, 91–93, 95–97, 123, 158,

162, 163, 199–201, 221–223, 234, 257, 271, 292, 296, 298–300, 310, 312, 315, 317	\glslongextraGroupHeading 433, 434, 436, 437, 439, 440, 442, 443, 445, 446, 448, 450
\glslabeltok . 122, 206, 208, 228–234, 236, 238–241, 243–249, 252, 255, 257, 259– 262, 264, 266, 269, 271, 273–281, 283, 285, 286, 288, 290–292, 295, 296, 298– 303, 305–308, 310, 312–315, 317, 318, 320	\glslongextraHeaderFmt 430, 431, 433, 435, 436, 438, 439, 441, 442, 444, 445, 447–449
\glsletentryfield 193	\glslongextraLocationAlign 434, 436, 437, 440, 443, 446, 449
\glslink 122, 351	\glslongextraLocationDescNameHeader 437
\glslink options	\glslongextraLocationDescNameTabularFooter 436, 437
counter 10	\glslongextraLocationDescNameTabularHeader 436, 437
format 191	\glslongextraLocationDescSymNameHeader 449
hyper 320	\glslongextraLocationDescSymNameTabularFooter 448, 449
hyperoutside 72	\glslongextraLocationDescSymNameTabularHeader 448, 449
noindex 8, 9, 92, 320	\glslongextraLocationFmt 434, 437, 440, 443, 446, 450
textformat 75	\glslongextraLocationSymDescNameHeader 446
theHvalue 74	\glslongextraLocationSymDescNameTabularFooter 445, 446
thevalue 74, 154	\glslongextraLocationSymDescNameTabularHeader 445, 446
wrgloss 8, 71	\glslongextraLocSetDescWidth 434, 436, 437
\glslinkcheckfirsthyperhook 91	\glslongextraNameAlign 432–435, 437, 438, 440, 441, 443–449
\glslinkpostsetkeys 10, 74, 158	\glslongextraNameDescHeader 433
\glslinkpresetkeys 10, 74, 158	\glslongextraNameDescLocationHeader 434
\glslinkvar 94	\glslongextraNameDescLocationTabularFooter 433, 434
\glslistchildpostlocation 387	\glslongextraNameDescLocationTabularHeader 433, 434
\glslistchildprelocation 387, 388	\glslongextraNameDescSymHeader 438
\glslistdesc 387, 388	\glslongextraNameDescSymLocationHeader 440
\glslistdottedwidth 386	\glslongextraNameDescSymLocationTabularFooter 439, 440
\glslistgroupheaderfmt 388, 389	\glslongextraNameDescSymLocationTabularHeader 439, 440
\glslistnavigationitem 388, 389	\glslongextraNameDescSymTabularFooter 437, 438
\glslistprelocation 387, 388	\glslongextraNameDescSymTabularHeader 437, 438
\glslocalunset 71, 158	\glslongextraNameDescTabularFooter 430, 432
\glslongaccessdisplay 170	
\glslongdefaultfont 210, 228, 230, 232, 236, 238, 241, 245, 247, 248, 250– 254, 259, 261, 262, 264, 266–268, 273, 277, 280, 281, 283, 284, 287, 294, 304, 317	
\glslongemfont .. 272, 275, 278, 285, 288, 289	
\glslongextraDescAlign 432–435, 437, 438, 440–449	
\glslongextraDescFmt 429, 433, 434, 436, 437, 439, 440, 442, 443, 445, 446, 448, 450	
\glslongextraDescNameHeader 436	
\glslongextraDescNameTabularFooter 435	
\glslongextraDescNameTabularHeader 435	
\glslongextraDescSymNameHeader 448	
\glslongextraDescSymNameTabularFooter 447, 448	
\glslongextraDescSymNameTabularHeader 447, 448	

<code>\glslongextraNameDescTabularHeader</code>	<code>\glslongfootnotefont</code>
..... 430, 432	232, 233, 235, 255, 257, 269, 271, 291, 293
<code>\glslongextraNameFmt</code> ... 433, 434, 436,	<code>\glslonghyphenfont</code>
437, 439, 440, 442, 443, 445, 446, 448, 450 304, 305, 307, 308, 310, 313, 315
<code>\glslongextraNameSymDescHeader</code> 442	<code>\glslongonlyfont</code>
<code>\glslongextraNameSymDescLocationHeader</code>	<code>\glslongpltok</code>
..... 443	207, 208, 228–232, 241, 243, 244, 246, 247,
<code>\glslongextraNameSymDescLocationTabularFooter</code>	251, 255, 259–262, 266, 269, 273–279,
..... 442, 443	283, 285, 288, 290, 295, 296, 298, 300–
<code>\glslongextraNameSymDescLocationTabularHeader</code>	303, 305–308, 310, 311, 313, 314, 318, 319
..... 442, 443	<code>\glslongpluralaccessdisplay</code>
<code>\glslongextraNameSymDescTabularFooter</code>	<code>\glslongtok</code>
..... 441 122, 206, 208, 228–232, 234, 236,
<code>\glslongextraNameSymDescTabularHeader</code>	238, 241, 243–248, 251, 255, 256, 259,
..... 441	260, 262, 266, 269, 271, 273–280, 283,
<code>\glslongextraSetDescWidth</code> .. 431, 432, 435	285, 288, 290, 292, 295, 296, 298, 300–
<code>\glslongextraSubDescFmt</code> 433, 434, 436,	303, 305–308, 310, 311, 313–315, 318, 319
437, 439, 440, 442, 443, 445, 447, 448, 450	<code>\glslonguserfont</code>
<code>\glslongextraSubLocationFmt</code>	295, 297–300, 302
..... 434, 437, 440, 443, 447, 450	<code>\glsmcols</code>
<code>\glslongextraSubNameFmt</code> 433, 434, 436,	416–420
437, 439, 440, 442, 443, 445, 447, 448, 450	<code>\GLSname</code>
<code>\glslongextraSubSymbolFmt</code>	326
.... 439, 440, 442, 443, 445, 447, 448, 450	<code>\Glsname</code>
<code>\glslongextraSymbolAlign</code>	326
..... 438, 440, 441, 443–449	<code>\glsname</code>
<code>\glslongextraSymbolFmt</code>	326
429, 439, 440, 442, 443, 445, 446, 448, 450	<code>\glsnameaccessdisplay</code> .. 164, 185, 186, 188
<code>\glslongextraSymDescNameHeader</code> 445	<code>\glsnamefont</code>
<code>\glslongextraSymDescNameTabularFooter</code>	185, 187, 188, 190, 431
..... 444	<code>\glsnavhyperlink</code>
<code>\glslongextraSymDescNameTabularHeader</code>	135
..... 444	<code>\glsnavhyperlinkname</code>
<code>\glslongextraSymLocSetDescWidth</code> ...	95
..... 439, 440, 443, 446, 449	<code>\glsnavhypertarget</code>
<code>\glslongextraSymSetDescWidth</code> 388, 389, 399, 401, 403, 415–420
..... 432, 438, 441, 444, 445, 447, 448	<code>\glsnavigation</code>
<code>\glslongextraTabularVAlign</code> . 432, 434– 388, 389, 399, 401, 403, 415–420
436, 438, 439, 441, 443, 444, 446, 447, 449	<code>\glsnextpages</code>
<code>\glslongextraUpdateWidest</code> 346–348	129
<code>\glslongextraUpdateWidestChild</code> 346–348	<code>\glsnoidxdisplayloc</code>
<code>\GlsLongExtraUseTabularfalse</code>	132, 133, 343
432	<code>\glsnoidxdisplaylocloclisthandler</code> 132
<code>\glslongfont</code>	<code>\glsnoidxloclist</code>
100, 210, 216, 217,	133, 153, 154
220, 221, 228–231, 233, 235, 236, 238,	<code>\glsnoidxnumberlistloophandler</code> 132
241–243, 245, 247, 248, 250, 252, 253,	<code>\glsnonextpages</code>
255, 257, 259, 261, 262, 264, 266, 267,	129
269, 271, 273, 275, 277, 278, 280, 281,	<code>\glsnonumberlistfalse</code>
283, 285, 287, 289, 291, 293, 295, 297,	66
299, 302, 305, 307, 310, 313, 315, 318, 319	<code>\glsnonumberlisttrue</code>
	66
	<code>\glsnopostdotfalse</code>
	130
	<code>\glsnopostdottrue</code>
	129
	<code>\glsnumberlistloop</code>
	127
	<code>\glsnumlistlastsep</code>
	132
	<code>\glsnumlistsep</code>
	132
	<code>\glsopenbrace</code>
	54, 141, 142
	<code>\glsorder</code>
	125
	<code>\glspagelistwidth</code> 390, 392, 394, 396, 430, 432
	<code>\glspar</code>
	152
	<code>\glspatchLToutput</code>
	432, 434, 435,
	437, 438, 440, 441, 443, 445, 446, 448, 449

<code>\glspenaltygroupskip</code> ...	433, 434, 436, 437, 439, 440, 442, 444, 445, 447, 448, 450	<code>\glstarget</code>	147, 149, 386– 398, 400, 402, 413, 414, 424, 425, 428, 429
<code>\GLSpl</code>	109, 121, 157	<code>\GLStext</code>	326, 327
<code>\Glspl</code>	63, 109, 121, 157	<code>\Glstext</code>	327
<code>\glspl</code>	63, 109, 121, 157	<code>\glstext</code>	326, 327
<code>\GLSplural</code>	327, 328	<code>\glstextaccessdisplay</code>	164, 165
<code>\Glsplural</code>	328	<code>\glstextformat</code>	71, 75
<code>\glsplural</code>	327	<code>\glstextup</code>	244
<code>\glspluralaccessdisplay</code>	165	<code>\glstreechilddesc</code>	399, 400
<code>\glspluralsuffix</code>	145, 206, 210	<code>\glstreechildpredesc</code>	400
<code>\glspostdescription</code>	19, 129, 198, 386, 387, 389–397, 399–401, 403, 428	<code>\glstreechildprelocation</code> ...	399, 400, 402
<code>\glspostinline</code>	397	<code>\glstreechildsymbol</code>	398, 400
<code>\glspostlinkhook</code>	69, 71, 85–89, 101, 211–221	<code>\glstreedefaultnamefmt</code>	397, 398
<code>\glsprestandardsort</code>	127	<code>\glstreedesc</code>	398, 400
<code>\glsresetentrylist</code>	135, 150	<code>\glstreegroupheaderfmt</code>	399, 401–403, 414–420, 422
<code>\glssee</code>	55–57	<code>\glstreeindent</code>	400, 402, 412–414
<code>\glsseeformat</code>	51, 54, 60, 125, 132, 133	<code>\glstreeitem</code>	398, 416, 424
<code>\glsseelist</code>	54	<code>\glstreenamebox</code>	413, 414
<code>\glssetabbrvfmt</code>	68, 77, 99, 100, 183–188, 190, 196, 211–215, 218, 219, 221	<code>\glstreenamefmt</code>	397, 398, 400, 402, 404, 406–414
<code>\glssetattribute</code>	228– 234, 236, 238–241, 243–246, 248, 249, 252, 255, 257, 259–262, 264, 266, 269, 271, 273–276, 278–281, 283, 285, 286, 288, 290–292, 295, 296, 298, 299, 301– 303, 305–308, 310, 312–315, 317, 318, 320	<code>\glstreenavigationfmt</code>	399, 401, 403, 415–420
<code>\glssetcategoryattribute</code>	109, 121, 123, 157, 163, 178, 179, 181, 182, 197	<code>\glstreenonamechilddesc</code>	402
<code>\glssetnoexpandfield</code>	14	<code>\glstreenonamedesc</code>	402
<code>\glssettoctitle</code>	128	<code>\glstreenonamesymbol</code>	402
<code>\glssetwidest</code>	347	<code>\glstreepredesc</code>	399, 401
<code>\glsshortaccessdisplay</code>	169	<code>\glstreeprelocation</code> ...	398, 400, 402, 403
<code>\glsshortpltok</code>	207, 208, 228–232, 234, 236, 238, 245–249, 255, 256, 259–262, 264, 269, 271, 273– 281, 290, 292, 295, 296, 298, 300–303, 305, 306, 310, 311, 313–315, 317, 318, 320	<code>\glstreesubitem</code>	398, 425
<code>\glsshortpluralaccessdisplay</code>	170	<code>\glstreesubsubitem</code>	398, 425
<code>\glsshorttok</code>	122, 206–208, 227–232, 234, 236, 238, 243, 244, 246– 249, 251, 255, 256, 259, 260, 262, 264, 266, 269, 271, 273, 274, 276–281, 283, 285, 290, 292, 295, 296, 298, 300–303, 305, 306, 308, 310, 311, 313–315, 317–319	<code>\glstreesymbol</code>	398, 400
<code>\glssubentryitem</code>	148, 386–396, 398, 400, 402, 414, 425, 429	<code>\GLstrLetField</code>	39
<code>\glssymbolaccessdisplay</code>	167	<code>\glstype</code>	71, 74, 85–89, 158, 211–221
<code>\glssymbolpluralaccessdisplay</code> .	167, 168	<code>\glsunset</code>	58, 71, 112, 113, 159
		<code>\glsupdatewidest</code>	346, 347
		<code>\gluserdescription</code> ...	295, 296, 298, 301
		<code>\glswrite</code>	54, 125
		<code>\glswriteentry</code>	8, 9
		<code>\Glsxtr</code>	64
		<code>\glxtr</code>	64
		<code>\glxtr@do@wrglossary</code>	8, 10, 13, 16
		<code>\glxtr@addloclistfield</code>	15, 16
		<code>\glxtr@addunused</code>	58
		<code>\glxtr@applyabbrvfmt</code>	221
		<code>\glxtr@applyabbrvstyle</code> ...	204, 206, 225
		<code>\glxtr@beginbookindex</code>	423
		<code>\glxtr@counterrecord</code>	147
		<code>\glxtr@dblfloat</code>	18
		<code>\glxtr@do@alsoindex@wrglossary</code> ...	16
		<code>\glxtr@do@autoadd</code>	9, 73, 74

<code>\glsxtr@doooption</code>	5, 18, 19, 26, 27, 29	<code>\glsxtr@writefields</code>	143
<code>\glsxtr@endbookindex</code>	424	<code>\glsxtrabbreviationfont</code>	69
<code>\glsxtr@fields</code>	145	<code>\glsxtrabbrvfootnote</code>	
<code>\glsxtr@float</code>	18	232–234, 255–257, 269–271, 290–292
<code>\glsxtr@headentry@p</code>	103, 104	<code>\glsxtrabbrvpluralsuffix</code>	145,
<code>\glsxtr@hyperoutsidedefalse</code>	72	210, 228, 230, 233, 235, 236, 238, 241,	
<code>\glsxtr@hyperoutsidettrue</code>	72	244, 259, 272, 295, 304, 307, 310, 315, 318	
<code>\glsxtr@ifnextpunc</code>	202	<code>\glsxtrabbrvtype</code>	20, 208
<code>\glsxtr@ifpunctoken</code>	203	<code>\glsxtractivatenopost</code>	129
<code>\glsxtr@inc@linkcount</code>	74, 163	<code>\glsxtraddallcrossrefs</code>	57
<code>\glsxtr@inc@wrglossaryctr</code>	8, 9, 26, 29	<code>\glsxtralias</code>	93
<code>\glsxtr@indexonly@saveentrycounter</code>		<code>\glsxtrAltTreeIndent</code>	403, 404
.....	16, 29	<code>\glsxtralttreeInit</code>	413, 419, 420
<code>\glsxtr@keylist</code>	62, 63	<code>\glsxtrAltTreePar</code>	403
<code>\glsxtr@label</code>	427	<code>\glsxtrAltTreeSetHangIndent</code> ...	403, 413
<code>\glsxtr@langtag</code>	145	<code>\glsxtrAltTreeSetSubHangIndent</code>	414
<code>\glsxtr@linkprefix</code>	145	<code>\glsxtralttreeSubSymbolDescLocation</code>	414
<code>\glsxtr@loaddialect</code>	340, 384	<code>\glsxtralttreeSymbolDescLocation</code> ..	
<code>\glsxtr@locationhypertext</code>	345	403, 414
<code>\glsxtr@makeglossaries</code>	125	<code>\glsxtrassignfieldfont</code>	78–85
<code>\glsxtr@newabbreviation</code>	123, 205	<code>\glsxtrautoindex</code>	192
<code>\glsxtr@next</code>	203	<code>\glsxtrautoindexassignsort</code>	192
<code>\glsxtr@org@do@wrglossary</code>	29	<code>\glsxtrautoindexentry</code>	192
<code>\glsxtr@org@dohyperlink</code>	95	<code>\glsxtrautoindexesc</code>	192
<code>\glsxtr@org@getgrouptitle</code>	134	<code>\glsxtrbibaddress</code>	342
<code>\glsxtr@org@newignoredglossary</code>	46	<code>\glsxtrbibauthor</code>	342
<code>\glsxtr@orgmakenoidxglossaries</code>	59	<code>\glsxtrbibbooktitle</code>	342
<code>\glsxtr@pluralsuffixes</code>	145	<code>\glsxtrbibchapter</code>	342
<code>\glsxtr@process</code>	150, 151	<code>\glsxtrbibedition</code>	342
<code>\glsxtr@provideignoredglossary</code>	47	<code>\glsxtrbibhowpublished</code>	342
<code>\glsxtr@punclist</code>	202, 203	<code>\glsxtrbibinstitution</code>	342
<code>\glsxtr@record</code>	11, 12, 145	<code>\glsxtrbibjournal</code>	342
<code>\glsxtr@record@nameref</code>	12, 145	<code>\glsxtrbibmonth</code>	343
<code>\glsxtr@record@nr</code>	15	<code>\glsxtrbibnote</code>	343
<code>\glsxtr@recordsee</code>	13	<code>\glsxtrbibnumber</code>	343
<code>\glsxtr@renewcommand</code>	341	<code>\glsxtrbiborganization</code>	343
<code>\glsxtr@resource</code>	143, 145	<code>\glsxtrbibpages</code>	343
<code>\glsxtr@s@newignoredglossary</code>	46	<code>\glsxtrbibpublisher</code>	343
<code>\glsxtr@s@provideignoredglossary</code> ...	47	<code>\glsxtrbibschooll</code>	343
<code>\glsxtr@saveentrycounter</code>	8, 10, 13, 93	<code>\glsxtrbibseries</code>	343
<code>\glsxtr@setaccessdisplay</code>	190	<code>\glsxtrbibtitle</code>	343
<code>\glsxtr@setbookindexmark</code>	426	<code>\glsxtrbibtype</code>	343
<code>\glsxtr@setup@record</code>	15, 16, 29, 30	<code>\glsxtrbibvolume</code>	343
<code>\glsxtr@shortcutsval</code>	145	<code>\glsxtrbookindexatendgroup</code>	424
<code>\glsxtr@texencoding</code>	145	<code>\glsxtrbookindexatsubendgroup</code>	425
<code>\glsxtr@undefaction@nr</code>	7	<code>\glsxtrbookindexatsubsubendgroup</code> ..	425
<code>\glsxtr@undefaction@val</code>	7	<code>\glsxtrbookindexbetween</code>	424
<code>\glsxtr@usesee</code>	51	<code>\glsxtrbookindexbookmark</code>	426
<code>\glsxtr@warnonexistsordo</code>	7, 15, 16, 50	<code>\glsxtrbookindexcols</code>	423, 424

\glxtrbookindexcolspread	423	\GlsXtrEnableEntryUnitCounting	109
\glxtrbookindexfirstmarkfmt	427	\GlsXtrEnableOnTheFly	61, 64
\glxtrbookindexformatheader	426	\glxtrtrendfor	35
\glxtrbookindexgroupskip	426	\glxtrtrfieldlistgadd	146
\glxtrbookindexlastmarkfmt	427	\glxtrtrfieldtitlecase	184–187, 190
\glxtrbookindexlocation	422, 424	\glxtrtrfieldtitlecasecs	183
\glxtrbookindexmulticolseenv ..	423, 424	\glxtrtrfieldxifinlist	151
\glxtrbookindexname	421, 424	\glxtrtrfirstscfont	244
\glxtrbookindexparentchildsep	422–424	\glxtrtrfirstsmfont	258
\glxtrbookindexparentsubchildsep .		\GlsXtrFmtDefaultOptions	33
.....	423–425	\glxtrtrfmtdisplay	33
\glxtrbookindexprelocation ...	421, 424	\glxtrtrfmtexternalnameref	344, 345
\glxtrbookindexsubbetween	425	\GlsXtrFmtField	33, 34
\glxtrbookindexsublocation	425	\glxtrtrfmtinternalnameref	344, 345
\glxtrbookindexsubname	425	\glxtrtrfootnotename	
\glxtrbookindexsubprelocation ...	425	232, 234, 255, 256, 269, 270, 290, 292
\glxtrbookindexsubsubbetween	425	\GlsXtrForeignTextField	41
\glxtrbookindexthepage	426, 427	\GlsXtrFormatLocationList	66, 68, 410–412
\glxtrcat	62, 63	\GLSxtrfull	21, 331, 332
\glxtrchecknohyperfirst	78–80	\Glsxtrfull	21, 332
\glxtrcombiningdiacriticIIrules .	355	\glxtrfull	20, 21, 331
\glxtrcombiningdiacriticIrules ..	355	\Glsxtrfullformat	
\glxtrcombiningdiacriticIVrules ..	355	209, 223, 225, 226, 228, 231, 233, 235,
\glxtrComputeTreeIndent	413, 414		237, 239, 242, 245, 247, 249, 250, 253,
\glxtrComputeTreeSubIndent	414		254, 256, 257, 259, 261, 263, 265, 267,
\glxtrcounterprefix	137		268, 270, 271, 273, 275, 277, 279, 281,
\glxtrcurrencyrules	358		282, 284, 286, 288, 290, 291, 293, 296,
\Glsxtrdefaultsubsequentfmt ...	224, 225		297, 299, 302, 305, 308, 311, 313, 316, 318
\glxtrdefaultsubsequentfmt ...	223, 225	\glxtrfullformat	
\Glsxtrdefaultsubsequentplfmt .	224, 225	209, 223, 225, 226, 228, 230,
\glxtrdefaultsubsequentplfmt .	224, 225		233, 235, 237, 239, 242, 245, 247, 249,
\GlsXtrDefineAbbreviationShortcuts			250, 253–255, 257, 259, 261, 263, 264,
.....	22, 23		267–269, 271, 273, 275, 277, 278, 280,
\GlsXtrDefineAcShortcuts	23		282, 284, 286, 288, 289, 291, 293, 295,
\GlsXtrDefineOtherShortcuts	23		297, 299, 302, 305, 308, 311, 313, 316, 318
\glxtrdetoklocation	157	\GLSxtrfullpl	21, 332, 333
\glxtrdiscardperiod	199	\Glsxtrfullpl	21, 333
\glxtrdisplayendloc	136	\glxtrfullpl	20, 21, 332
\glxtrdisplayendlochook	136	\Glsxtrfullplformat	
\glxtrdisplaysingleloc	136	209, 223, 225, 226, 228, 231, 233, 235,
\glxtrdisplaystartloc	136		237, 239, 242, 245, 247, 249, 250, 253,
\glxtrdoautoindexname	93, 94, 189		254, 256, 257, 260, 261, 263, 265, 267,
\glxtrdopostpunc	234, 257, 271, 292		269, 270, 272, 274, 275, 277, 279, 281,
\glxtrdowrglossaryhook	94		282, 284, 286, 288, 290, 291, 293, 296,
\GlsXtrDualField	342		297, 299, 302, 305, 308, 311, 313, 316, 319
\glxxtremsuffix	273, 275, 277,	\glxtrfullplformat	
	278, 280, 281, 283, 285, 287, 288, 291, 293	223, 225, 226, 228, 230, 233,
\GlsXtrEnableEntryCounting	121		235, 237, 239, 242, 245, 247, 249, 250,
			253–255, 257, 259, 261, 263, 264, 267,

268, 270, 271, 273, 275, 277, 279, 281, 282, 284, 286, 288, 289, 291, 293, 296, 297, 299, 302, 305, 308, 311, 313, 316, 318	\glxtrifrecordtrigger	159–161
\glxtrifullsep 208, 209, 228–231, 233–240, 242, 244–252, 254, 256, 258–268, 270, 272–289, 291–294, 304–309, 312–315, 319	\glxtrifwasfirstuse	77–80, 85–89, 91, 123, 200, 201, 211, 214–221, 234, 235, 257, 271, 292, 293, 296, 298–300, 310, 312, 315, 317
\glxtrngenabbrvfmt	69	
\glxtrngeneralpuncIIrules	358	
\glxtrngeneralpuncIrules	358	
\glxtrnggetgrouptitle	135, 426	
\glxtrngroupfield	153	
\Glsxtrheadfirst	323	
\glxtrheadfirst	323	
\Glsxtrheadfirstplural	323	
\glxtrheadfirstplural	323	
\Glsxtrheadfull	323	
\glxtrheadfull	323	
\Glsxtrheadfullpl	323	
\glxtrheadfullpl	323	
\Glsxtrheadlong	323	
\glxtrheadlong	323	
\Glsxtrheadlongpl	323	
\glxtrheadlongpl	323	
\Glsxtrheadname	323	
\glxtrheadname	147, 323	
\Glsxtrheadplural	323	
\glxtrheadplural	323	
\Glsxtrheadshort	323	
\glxtrheadshort	322	
\Glsxtrheadshortpl	323	
\glxtrheadshortpl	322	
\Glsxtrheadtext	323	
\glxtrheadtext	323	
\glxtrhiernamesep	52, 53	
\glxtrhyperlink	25, 26, 96	
\glxtrhyphensuffix	305, 313	
\glxtridentifyglslake	155	
\glxtrifcounttrigger	112, 113	
\glxtrifcustomdiscardperiod	200	
\glxtrifemptyglossary	135, 141, 149	
\GlsXtrIfFieldEqNum	148	
\GlsXtrIfFieldNonZero	341	
\glxtrifhasfield 41, 52, 53, 92, 341, 342, 421		
\glxtrifhyphenstart	304, 306, 309, 312, 314, 315	
\glxtrifindexing	93	
\glxtrifinmark	76, 103–106, 321–323	
\glxtrifnextpunc	202, 203	
\glxtrifperiod	200–202	
\glxtrinclinkcounter	163	
\glxtrinindexaliased	92, 93	
\glxtrinindexseealso	56, 57	
\glxtrinithyperoutside	74	
\glxtrinitwrgloss	74, 158	
\glxtrinitwrglossbeforefalse ...	71, 72	
\glxtrinitwrglossbeforetrue	71, 72	
\Glsxtrinlinefullformat 209, 211, 225, 226, 234, 235, 237, 239, 240, 242, 249– 252, 254, 256, 258, 263–265, 267, 268, 270, 272, 280, 282–284, 286, 287, 289, 292, 293, 297, 300, 308, 311, 316, 319, 338		
\glxtrinlinefullformat	209, 211, 212, 225, 226, 233, 235, 236, 238, 240, 242, 248, 250– 252, 254, 256, 258, 262, 264–266, 268, 270, 272, 280–282, 284, 285, 287, 289, 291, 293, 297, 299, 307, 311, 316, 319, 338	
\Glsxtrinlinefullplformat 210, 213, 225, 226, 234, 235, 237, 239, 240, 242, 249– 252, 254, 256, 258, 263–265, 267, 268, 270, 272, 280, 282–284, 286, 288, 289, 292, 293, 297, 300, 308, 311, 316, 319, 339		
\glxtrinlinefullplformat	209, 212, 213, 225, 226, 233, 235, 236, 238, 240, 242, 248, 250– 252, 254, 256, 258, 263–266, 268, 270, 272, 280, 281, 283, 284, 286, 287, 289, 291, 293, 297, 300, 307, 311, 316, 319, 339	
\glxtrininsertinsidefalse	227	
\GlsXtrInternalLocationHyperlink 26, 137		
\glxtrLatinA	360–365	
\glxtrLatinAELigature	362, 364, 365	
\glxtrLatinE	360–365	
\glxtrLatinEszettSs	361–363, 365	
\glxtrLatinEszettSz	361, 364	
\glxtrLatinEth	360–364	
\glxtrLatinH	360–365	
\glxtrLatinI	360–365	
\glxtrLatinInsularG	364	
\glxtrLatinK	360–365	
\glxtrLatinL	360–365	
\glxtrLatinM	360–365	
\glxtrLatinN	360–365	

<code>\glxstrLatinO</code>	360–365	<code>\glxstrMathItalicTau</code>	370, 374, 375
<code>\glxstrLatinOELigature</code>	362, 364, 365	<code>\glxstrMathItalicTheta</code> .	369, 370, 373, 374
<code>\glxstrLatinP</code>	360–365	<code>\glxstrMathItalicUpsilon</code> ...	370, 374, 375
<code>\glxstrLatinS</code>	360–365	<code>\glxstrMathItalicXi</code>	369, 370, 374, 375
<code>\glxstrLatinT</code>	360–365	<code>\glxstrMathItalicZeta</code> ..	369, 370, 373, 374
<code>\glxstrLatinThorn</code>	364	<code>\glxstrmultisupplcation</code>	343
<code>\glxstrLatinX</code>	360–365	<code>\glxstrnamereflink</code>	344
<code>\GlsXtrLocationField</code>	153	<code>\glxstrnewabbrevpresetkeyhook</code>	207
<code>\glxstrlocationhyperlink</code>	137	<code>\glxstrnewnumber</code>	22
<code>\glxstrlocrangefmt</code>	136	<code>\glxstrnewsymbol</code>	22
<code>\GLSxtrlong</code>	21, 330	<code>\glxstrNoGlossaryWarning</code>	15, 24, 138
<code>\Glsxtrlong</code>	21, 330, 331	<code>\GlsXtrNoGlsWarningAutoMake</code>	142
<code>\glxstrlong</code>	20, 21, 330	<code>\GlsXtrNoGlsWarningBuildInfo</code>	142
<code>\glxstrlonghyphen</code>	311	<code>\GlsXtrNoGlsWarningCheckFile</code>	142
<code>\glxstrlonghyphenno</code>	307, 308	<code>\GlsXtrNoGlsWarningEmptyMain</code> ..	141, 142
<code>\glxstrlonghyphenshort</code>	305	<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	142
<code>\glxstrlongnoshortdescname</code> .	241, 288, 307	<code>\GlsXtrNoGlsWarningEmptyStart</code>	141
<code>\glxstrlongnoshortname</code>		<code>\GlsXtrNoGlsWarningHead</code>	141
.....	243, 251, 266, 283, 285, 308	<code>\GlsXtrNoGlsWarningMisMatch</code>	142
<code>\GLSxtrlongpl</code>	21, 330, 331	<code>\GlsXtrNoGlsWarningNoOut</code>	142
<code>\Glsxtrlongpl</code>	21, 331	<code>\GlsXtrNoGlsWarningTail</code>	142, 143
<code>\glxtrlongpl</code>	20, 21, 330	<code>\glxstrnopostpunc</code>	129
<code>\glxstrlongshortdescname</code>		<code>\glxstronlydescname</code>	319
.....	229, 246, 260, 274, 276, 305, 311	<code>\glxstronlydescsort</code>	319
<code>\glxstrlongshortdescsort</code>		<code>\glxstronlyname</code>	318
....	229, 246, 260, 274, 276, 301, 306, 311	<code>\glxstronlysuffix</code>	318
<code>\glxstrlongshortname</code>		<code>\glxstrorg@ifKV@glslink@hyper</code>	69
	228, 244, 259, 273, 274, 295, 296, 305, 310	<code>\glxstrorglong</code>	206, 229, 307
<code>\glxstrlongshortuserdescname</code> ..	298, 301	<code>\glxstrorgshort</code>	206, 229
<code>\glxstrmarkhook</code>	320, 321	<code>\GLSxtrp</code>	102
<code>\glxstrMathItalicAlpha</code> .	369, 370, 373, 374	<code>\Glsxtrp</code>	102
<code>\glxstrMathItalicBeta</code> ..	369, 370, 373, 374	<code>\glxstrp</code>	102, 104
<code>\glxstrMathItalicChi</code>	370, 374, 375	<code>\glxstrparen</code>	
<code>\glxstrMathItalicDelta</code> .	369, 370, 373, 374	
<code>\glxstrMathItalicEpsilon</code> ..	369, 370, 373, 374	200, 201, 208, 209, 228–231, 233–240,
<code>\glxstrMathItalicEta</code> ...	369, 370, 373, 374	242, 244–252, 254, 256, 258–268, 270,
<code>\glxstrMathItalicGamma</code> .	369, 370, 373, 374	272–289, 291–294, 304–309, 312–315, 319
<code>\glxstrMathItalicIota</code> ..	369, 370, 373, 374	<code>\Glsxtrpl</code>	64
<code>\glxstrMathItalicKappa</code> .	369, 370, 373, 374	<code>\glxstrpl</code>	64
<code>\glxstrMathItalicLambda</code> ..	369, 370, 373, 374	<code>\glxstrpostdescription</code>	
<code>\glxstrMathItalicMu</code>	369, 370, 373, 374	129, 130, 181, 198, 397
<code>\glxstrMathItalicNu</code>	369, 370, 373, 375	<code>\glxstrposthyphenlong</code>	315, 317
<code>\glxstrMathItalicOmega</code>	370, 374, 375	<code>\glxstrposthyphenshort</code>	310, 312
<code>\glxstrMathItalicOmicron</code> ..	369, 370, 374, 375	<code>\glxstrposthyphensequent</code>	
<code>\glxstrMathItalicPhi</code>	370, 374, 375	310, 312, 315, 317
<code>\glxstrMathItalicPi</code>	369, 370, 374, 375	<code>\glxstrpostlink</code>	200
<code>\glxstrMathItalicPsi</code>	370, 374, 375	<code>\glxstrpostlinkendsentence</code>	200
<code>\glxstrMathItalicRho</code> ...	369, 370, 374, 375	<code>\glxstrpostlinkhook</code>	199
<code>\glxstrMathItalicSigma</code>	370, 374, 375	<code>\glxstrpostlocalreset</code>	108, 110, 119
		<code>\glxstrpostlocalunset</code>	108, 110, 118

\glsxtrpostlongdescription	45	\GLSxtrshortpl	21, 324, 325
\glsxtrpostnamehook	186–188, 191	\Glsxtrshortpl	20, 21, 325
\GlsXtrPostNewAbbreviation .	208, 225,	\glsxtrshortpl	20, 21, 324
226, 228–232, 234, 236–241, 243–246,		\glsxtrsmfont	258
248, 249, 252, 255, 257, 259–262, 264,		\glsxtrsmsuffix	
266, 269, 271, 273–276, 278–281, 283,		259, 261, 262, 264, 266, 267, 269, 271
285, 286, 288, 290, 292, 295, 296, 298–		\GlsXtrStandaloneEntryName	147
303, 305–308, 310, 312–315, 317, 318, 320		\GlsXtrStandaloneEntryOther	148
\glsxtrpostreset	108, 110, 118	\GlsXtrStandaloneGlossaryType .	147, 148
\glsxtrpostunset	106, 110, 118	\GlsXtrStandaloneSubEntryItem .	147, 148
\glsxtrprelocation		\Glsxtrsubsequentfmt	
.....	387, 389, 391, 393, 395, 398, 421	222, 225, 241, 252, 253,
\glsxtrprotectlinks	96, 97	266, 268, 284, 285, 287, 289, 307, 310, 316	
\GlsXtrRecordCounter	13	\glsxtrsubsequentfmt	
\glsxtrrecordtriggervalue	158	222, 225, 241, 252, 253,
\GlsXtrRecordWarning	143	266, 267, 283, 285, 287, 289, 307, 310, 316	
\glsxtrregularfont	68, 69, 77	\Glsxtrsubsequentplfmt	
\glsxtrresourcecount	144	222, 225, 242, 252, 254,
\glsxtrresourcefile	144	266, 268, 284, 285, 287, 289, 307, 310, 316	
\glsxtrresourceinit	143	\glsxtrsubsequentplfmt	
\glsxtrrestoremarkhook	321	222, 225, 241, 252, 253,
\glsxtrrestorepostpunc	129, 130	266, 268, 283, 285, 287, 289, 307, 310, 316	
\glsxtrscfont	244	\glsxtrsupplocationurl .	137, 138, 343, 345
\glsxtrscsuffix		\glsxtrtagfont	198
....	245, 247, 248, 250, 252, 253, 255, 257	\Glsxtrtitlefirst	322, 323, 336
\GlsXtrSetActualChar	195	\glsxtrtitlefirst	322, 323, 336
\glsxtrsetaliasnoindex	15, 16, 92, 93	\Glsxtrtitlefirstplural	322, 323, 337
\GlsXtrSetEncapChar	195	\glsxtrtitlefirstplural	322, 323, 336, 337
\GlsXtrSetEscChar	195	\Glsxtrtitlefull	322–324, 338, 339
\glsxtrsetfieldifexists	38, 39	\glsxtrtitlefull	322, 323, 338
\glsxtrsetglossarylabel	131	\Glsxtrtitlefullpl	322–324, 339
\GlsXtrSetLevelChar	195	\glsxtrtitlefullpl	322–324, 339
\glsxtrsetpopts	101	\Glsxtrtitlelong	322, 323, 337
\glsxtrsetupfulldefs		\glsxtrtitlelong	322, 323, 337
.....	211–213, 235, 257, 271, 293	\Glsxtrtitlelongpl	322, 323, 338
\GLSxtrshort	21, 105, 106, 324, 325	\glsxtrtitlelongpl	322, 323, 338
\Glsxtrshort	20, 21, 325	\Glsxtrtitlename	322, 323, 334, 335
\glsxtrshort	20, 21, 324	\glsxtrtitlename	322, 323, 334
\glsxtrshortdescname ...	238, 249, 264, 281	\glsxtrtitleorpdforheading	
\glsxtrshorthyphen	316	27, 147, 148, 322, 323
\glsxtrshorthyphenlong	313	\Glsxtrtitleplural	322, 323, 336
\glsxtrshortlongdescname		\glsxtrtitleplural	322, 323, 335
.....	231, 247, 261, 277, 279, 314, 317	\Glsxtrtitleshort	322, 323, 334
\glsxtrshortlongdescsort		\glsxtrtitleshort	322, 323, 333
....	231, 247, 262, 277, 279, 303, 314, 317	\Glsxtrtitleshortpl	322, 323, 334
\glsxtrshortlongname		\glsxtrtitleshortpl	322, 323, 333
230, 246, 260, 276, 278, 298, 301, 313, 315		\Glsxtrtitletext	322, 323, 335
\glsxtrshortlonguserdescname ..	300, 303	\glsxtrtitletext	322, 323, 335
\glsxtrshortnolongname .	236, 248, 262, 280	\GlsXtrTotalRecordCount	157

<code>\glxtrtreetopindent</code>	404, 412	
<code>\glxtrundefaction</code> 7, 15, 16, 31, 46, 47, 49, 50		
<code>\glxtrundeftag</code>	30, 132, 133	
<code>\GlsXtrUnknownDialectWarning</code>	42	
<code>\glxtrunsrtdo</code>	151	
<code>\glxtrUpAlpha</code>	368, 369, 373, 374	
<code>\glxtrUpBeta</code>	368, 369, 373, 374	
<code>\glxtrUpChi</code>	368, 369, 374, 375	
<code>\glxtrUpDelta</code>	368, 369, 373, 374	
<code>\glxtrUpDigamma</code>	368, 369, 373	
<code>\glxtrUpEpsilon</code>	368, 369, 373, 374	
<code>\glxtrUpEta</code>	368, 369, 373, 374	
<code>\glxtrUpGamma</code>	368, 369, 373, 374	
<code>\glxtrUpIota</code>	368, 369, 373, 374	
<code>\glxtrUpKappa</code>	368, 369, 373, 374	
<code>\glxtrUpLambda</code>	368, 369, 373, 374	
<code>\glxtrUpMu</code>	368, 369, 373, 375	
<code>\glxtrUpNu</code>	368, 369, 373, 375	
<code>\glxtrUpOmega</code>	368, 369, 374, 375	
<code>\glxtrUpOmicron</code>	368, 369, 374, 375	
<code>\glxtrUpPhi</code>	368, 369, 374, 375	
<code>\glxtrUpPi</code>	368, 369, 374, 375	
<code>\glxtrUpPsi</code>	368, 369, 374, 375	
<code>\glxtrUpRho</code>	368, 369, 374, 375	
<code>\glxtrUpSigma</code>	368, 369, 374, 375	
<code>\glxtrUpTau</code>	368, 369, 374, 375	
<code>\glxtrUpTheta</code>	368, 369, 373, 374	
<code>\glxtrUpUpsilon</code>	368, 369, 374, 375	
<code>\glxtrUpXi</code>	368, 369, 374, 375	
<code>\glxtrUpZeta</code>	368, 369, 373, 374	
<code>\GlsXtrUseAbbrStyleFmts</code>		
.....	229, 232, 238, 240, 241,	
	243, 244, 246, 248, 251, 260, 262, 265,	
	274, 276, 278, 279, 282, 287, 290, 298,	
	301, 303, 306, 307, 309, 312, 314, 317, 320	
<code>\GlsXtrUseAbbrStyleSetup</code>		
.....	237, 239, 240, 243,	
	244, 251, 253, 265, 267, 282, 286, 287, 290	
<code>\glxtruserfield</code>	294	
<code>\glxtruserparen</code>	295–303	
<code>\glxtrusersuffix</code>	295, 297, 299, 302	
<code>\glxtruseseealsoformat</code>	53, 54	
<code>\glxtruseseeformat</code>	51	
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	204, 226	
<code>\GlsXtrWarning</code>	62, 63	
<code>\glxtrword</code>	205	
<code>\glxtrwordsep</code>	205, 304, 306, 309, 312, 314, 315	
<code>\glxtrwrglossmark</code>	27	
		H
<code>\hangindent</code> .	400, 402, 403, 413–415, 419, 420	
<code>\hbox</code>	386	
<code>\hfill</code>	386	
<code>\href</code>	96	
<code>\hsize</code>	65	
<code>\hss</code>	386	
<code>\hyperlink</code>	16, 96, 343	
<code>\hyperpage</code>	191	
<code>\hyperref</code>	96, 137, 341, 346	
<code>hyperref package</code>	97, 191, 320, 333, 343	
		I
<code>\if</code>	61	
<code>\if@display</code>	12	
<code>\if@glxtr@autoseeindex</code>	28, 51, 55	
<code>\if@glxtr@equations</code>	9, 74	
<code>\if@glxtr@floats</code>	18	
<code>\if@glxtr@format@override</code>	192	
<code>\if@glxtrdocdefrestricted</code>	60	
<code>\if@glxtrindexcrossrefs</code>	17, 57	
<code>\ifblank</code>	32, 62, 63, 124	
<code>\ifbool</code>	31, 45	
<code>\ifcase</code> .	7, 15, 22, 24, 26, 60, 72, 130, 398, 424	
<code>\ifcsdef</code>	31, 38, 44,	
	46–49, 72, 75, 89, 90, 101–106, 116, 128,	
	134, 135, 148, 152, 155–157, 162, 183–	
	186, 188–190, 200, 204, 221, 225, 389–396	
<code>\ifcsstring</code>	31, 179, 224	
<code>\ifcsundef</code>	37, 42, 44, 46–48, 60, 64,	
	67, 97, 110, 116–119, 134, 138, 152, 163,	
	179, 224, 226, 227, 386, 404, 405, 413, 426	
<code>\ifcsvoid</code>	57, 178	
<code>\ifdef</code>	15, 22, 28, 34,	
	37, 38, 41, 43, 49, 50, 54, 55, 58, 65, 91,	
	92, 95, 96, 103–105, 125, 128, 132, 133,	
	143, 145, 155, 181, 182, 195, 198, 294,	
	322, 333–339, 341, 346–348, 383, 384,	
	386–389, 397–403, 414–420, 422, 426, 427	
<code>\ifdefempty</code>	7–9, 36, 37, 41, 42, 46–48, 51, 53,	
	74, 76, 109, 121, 125, 128, 136, 150, 153,	
	157, 158, 172–174, 197, 205, 221, 349, 423	
<code>\ifdefequal</code> .	40, 41, 59, 95, 142, 153, 154, 189	
<code>\ifdefstring</code>	6, 26, 39, 40, 192, 197, 422	
<code>\ifdefvoid</code>		
	50, 55–58, 96, 115, 133, 137, 153, 154, 345	
<code>\ifdim</code>	65, 66, 124, 404–412, 430, 431	
<code>\IfFileExists</code>	24, 138, 142, 143, 146, 384, 385	
<code>\ifglossaryexists</code>	50	

<code>\ifgl@sacronym</code>	20, 142	<code>\ifundef</code>	16,
<code>\ifgl@sacrshortcuts</code>	22		26, 36, 37, 41, 125, 197, 198, 343, 383, 430
<code>\ifgl@automake</code>	127, 142, 146	<code>\ifx</code> ...	8–11, 14, 15, 30, 54, 65, 66, 75, 124,
<code>\ifgl@sentrycounter</code>	43		128, 131, 136, 137, 143, 145, 146, 192,
<code>\ifgl@sentryexists</code>	9, 31,		193, 195, 203, 205, 207, 303, 304, 345, 420
	49, 50, 62, 63, 66, 77, 153, 179, 198, 199, 350	<code>\immediate</code>	59, 111, 120, 138, 139, 146
<code>\ifgl@sfieldeq</code>	177	<code>\index</code>	192
<code>\ifgl@shasfield</code>	33, 34, 294	<code>\indexspace</code> .	387, 399–403, 414–420, 422, 426
<code>\ifgl@shaslong</code>	115, 161, 162	<code>\input</code>	339
<code>\ifgl@shasparent</code> .	147, 148, 150, 153, 406–408	<code>\inputencodingname</code>	145
<code>\ifgl@shasshort</code>	51–53, 68, 69, 77	<code>\InputIfFileExists</code>	58
<code>\ifgl@shassymbol</code>	201, 400, 401, 403	<code>\istfilename</code>	125
<code>\ifgl@sindexonlyfirst</code>	93	<code>\item</code>	140, 141, 386–389, 398, 399, 415, 416
<code>\ifGlsLongExtraUseTabular</code>	432, 433, 435,		
	436, 438, 439, 441, 443, 444, 446, 447, 449		
<code>\ifgl@snogroupskip</code>	387, 389–397,		
	399, 400, 402, 414, 422, 433, 434, 436,		
	437, 439, 440, 442, 443, 445, 447, 448, 450		
<code>\ifgl@snonumberlist</code>	68		
<code>\ifgl@snopostdot</code>	19, 129		
<code>\ifgl@ssanitizesort</code>	127		
<code>\ifgl@ssubentrycounter</code>	43		
<code>\ifgl@sused</code>	57, 58, 91, 93,		
	111, 120, 123, 221, 406, 407, 409, 410, 412		
<code>\ifgl@sxindy</code>	138, 140		
<code>\ifgl@xtr@hyperoutside</code>	75		
<code>\ifgl@xtrinitwrglossbefore</code>	71, 75, 76, 158		
<code>\ifgl@xtrininsertinside</code>			
	214–221, 223, 224, 228,		
	230, 231, 233–242, 245, 247–261, 263–		
	275, 277–293, 295–297, 299, 300, 302,		
	304, 306, 309, 311, 312, 315, 316, 318, 319		
<code>\ifHy@hyperindex</code>	191		
<code>\ifinlist</code>	107		
<code>\ifinlistcs</code>	35, 60		
<code>\ifinner</code>	27		
<code>\ifKV@gl@slink@hyper</code>	69, 74–76		
<code>\ifKV@gl@slink@local</code>	71, 158		
<code>\ifKV@gl@slink@noindex</code> ..	8, 9, 13, 33, 92, 93		
<code>\ifmmode</code>	12, 27		
<code>\ifnum</code>	17,		
	37, 58, 59, 111, 112, 119, 120, 133, 144,		
	158, 346–348, 400, 402, 413, 414, 423, 424		
<code>\ifst@rred</code>	12		
<code>\ifstrempy</code>	148, 155, 163, 344, 346–349		
<code>\ifstrequal</code>	19, 24, 73, 344		
<code>\ifthenelse</code>	141, 142, 200		
<code>\IfTrackedDialectHasMapping</code>	42		
<code>\IfTrackedLanguageFileExists</code>	340		

<code>\listcsxadd</code>	34, 116	<code>\newabbreviationstyle</code> 227, 229–232, 234,	
<code>\listxadd</code>	107	236–241, 243–249, 251, 253, 255, 256,	
<code>\loadglsentries</code>	60, 139	259–263, 265–267, 269, 270, 273–283,	
<code>\long</code>	45	285–288, 290, 292, 295, 296, 298, 300,	
		301, 303–306, 308, 310–313, 315, 317–319	
M			
<code>\m@ne</code>	341	<code>\newacronym</code>	121, 123
<code>\MakeAcronymsAbbreviations</code>	123	<code>\newacronymhook</code>	122
<code>\makeatletter</code>	58, 138, 143, 194	<code>\newacronymstyle</code>	123
<code>\makeatother</code>	194	<code>\newcommand</code> ... 5–18, 20–26, 28–41, 43–54,	
<code>\makebox</code>	386, 413, 414	56–58, 61–64, 66–69, 71–73, 76–78, 89,	
<code>\makefirstuc</code>	198	90, 92–96, 98, 101, 103–110, 112, 114–	
<code>makeglossaries</code>	131	118, 120, 121, 123, 124, 128–134, 136–	
<code>\makeglossaries</code>	124, 139–142, 146	144, 146–172, 174–183, 189–206, 208–	
<code>\makeglossary</code>	125	221, 223–227, 229–232, 236, 238, 241,	
<code>makeindex</code>	451	243, 244, 258, 259, 272, 273, 294, 295,	
<code>makeindex</code>	16, 124, 451	297, 300, 304, 306, 309, 312, 314, 317–	
<code>\makenoidxglossaries</code>	140	319, 321–346, 348–351, 354–383, 385,	
<code>\MakeTextUppercase</code>	322	387, 397–401, 403–405, 412, 413, 421–	
<code>\MakeUppercase</code>	322, 323	423, 426–433, 435–439, 441, 442, 444–449	
<code>\marginpar</code>	27	<code>\newcount</code>	144, 154
<code>\markboth</code>	321	<code>\newcounter</code>	26, 162
<code>\markright</code>	321	<code>\newentry</code>	22
<code>\mathit</code>	353	<code>\newglossary</code>	20, 125
<code>\mathrm</code>	352–354	<code>\newglossaryentry</code>	
<code>\maxdimen</code>	65, 66	... 22, 59–61, 110, 117, 122, 181, 182, 208	
<code>\mbox</code>	388, 389, 413	<code>\newglossaryentry options</code>	
<code>\medskip</code>	142, 152	access	173
<code>\MessageBreak</code>	60,	alias	18, 50, 55, 57
64, 111, 112, 120, 124, 127, 128, 143, 224		desc	168, 176
<code>mfirstuc package</code>	197	descplural	168, 169, 176
<code>\mfirstucMakeUppercase</code>		first 95, 165, 166, 175, 227, 328, 329, 336, 451	
... 38, 78–90, 99–101, 103, 105, 106,		firstaccess	174
114, 122, 162, 164–171, 174–177, 186,		firstplural	166, 175, 227, 329, 336, 451
187, 191, 212, 213, 215, 217, 219, 221–223		group	152, 153
<code>\mfu@checkword@arg</code>	197, 198	loclist	34
<code>\mfu@checkword@do</code>	198	long	170, 177, 337
<code>\midrule</code>	430, 433, 435,	longplural	171, 177, 338
436, 438, 439, 441, 442, 444, 445, 447, 449		name	51, 164, 174, 192, 325, 326, 334
		plural	165, 174, 175, 227, 327, 328, 335
		see	17, 18, 28, 50, 51, 55, 57, 61, 125
		seealso	18, 50, 53, 55, 57, 462
N			
<code>\NeedsTeXFormat</code>	5, 340, 385, 421, 428	short	169, 176, 204
<code>\new@atom@glossaryentry</code>	59	shortaccess	172–174
<code>\new@command</code>	341	shortaccessplural	172
<code>\new@glossaryentry</code>	61, 127	shortplural	170, 177, 204
<code>\new@ifnextchar</code>	33, 89, 90,	symbol	167, 175
114, 155, 159–161, 202, 210–221, 350, 351		symbolplural	167, 168, 175, 176
<code>\newabbr</code>	21	text 95, 164, 165, 174, 227, 229, 326, 327, 335	
<code>\newabbreviation</code>	21	textaccess	173
<code>\newabbreviationhook</code>	208		

user2	43	\o	368
\newglossarystyle ...	423, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446, 447, 449	\openout	59
\newif	71, 191, 227, 432	\or	7, 16, 22, 23, 27, 60, 72, 398, 424
\newlength	403, 404	\org@glossaryentrynumbers	66, 129
\newnum	22	\org@glossarytitle	128
\newrobustcmd	32, 34, 35, 38–41, 54, 55, 59, 73, 77, 90, 101–103, 114, 129, 134, 147, 148, 151, 152, 155, 156, 159–161, 190, 197, 198, 210–221, 303, 321, 322, 324–333, 350, 351, 405–412	\org@ifKV@glslink@hyper	74, 76
\newsym	22	P	
\newterm	181	\p@glsl@hyp@opt	94
\newtoks	204	package options:	
\newwrite	59, 125	abbreviations	20
\nobreak	388, 389, 399, 401–403, 415–418	accsupp	23, 164
\NoCaseChange	103–106, 148, 324–332	acronym	20
\noexpand	11, 13, 24, 54, 56–58, 73, 122, 138, 139, 143, 150, 151, 153, 162, 193, 194, 208, 341, 350, 385, 423–425, 432, 434–441, 443–449	automake	127, 140, 146
\nofiles	141	immediate	146
\noindent	142, 401–403, 416–418, 420	true	146
\nopagebreak	399, 401–403, 414–420, 422, 426	autoseeindex	28
\nopostdesc	46, 62, 63, 129, 181	false	28
\ns@GLSxtrfull	212	counter	
\ns@GLsxtrfull	211	wrglossary	25
\ns@glsxtrfull	210	debug	
\ns@GLSxtrfullpl	213	showtargets	96
\ns@GLsxtrfullpl	213	docdef	16, 59, 60, 110, 117
\ns@glsxtrfullpl	212	atom	59
\ns@GLSxtrlong	217	false	59, 60
\ns@GLsxtrlong	216	restricted	17
\ns@glsxtrlong	216	true	58, 61
\ns@GLSxtrlongpl	221	docdefs	
\ns@GLsxtrlongpl	220	restricted	60
\ns@glsxtrlongpl	219	equations	9, 74
\ns@GLSxtrshort	215	indexcounter	341
\ns@GLsxtrshort	214	nonumberlist	66
\ns@glsxtrshort	214	nopostdot	18
\ns@GLSxtrshortpl	219	false	19
\ns@GLsxtrshortpl	218	numbers	22
\ns@glsxtrshortpl	217	postdot	18
\null	24	record	7, 14, 15, 59, 69, 124, 142, 143, 211–221, 460
\number	58, 116–119, 143, 155	alsoindex	8, 10
\numexpr	116, 117, 119	nameref	12, 29, 30, 143, 146
O		only	8, 141
\O	365, 368	shortcuts	22
		ac	22
		all	22
		false	22
		none	22
		true	22
		sort	
		use	77

<code>\rGlsformat</code>	160	<code>\string</code>	6, 11–13, 30, 54, 55, 59– 61, 64, 68, 75, 89, 90, 93, 95, 102, 109– 111, 117–121, 123–128, 138–143, 145– 147, 150–152, 154, 155, 185, 187, 188, 190, 192, 200, 340, 341, 350, 354–383, 426
<code>\rglsformat</code>	159, 162	<code>\strut</code>	386–397, 402, 429
<code>\rGLSpl</code>	157	<code>\subglossentry</code>	129, 153, 386–396, 398, 400, 402, 414, 424, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447, 448, 450
<code>\rGlspl</code>	157	<code>\subitem</code>	398
<code>\rglspl</code>	157	<code>\subsubitem</code>	398
<code>\rGLSplformat</code>	161	<code>\symbolname</code>	431, 438, 439, 441, 442, 444, 445, 447, 449
<code>\rGlsplformat</code>	160		
<code>\rglsplformat</code>	159, 162		
<code>\romannumeral</code>	404, 405, 413		
S			
<code>\s@glsxtrfmt</code>	33	T	
<code>\s@gls@hyp@opt</code>	94	<code>\tabcolsep</code>	431, 432
<code>\s@glsxtr@enabletagging</code>	196, 197	<code>\tablehead</code>	393–396
<code>\s@glsxtrfmt</code>	32	<code>\tabletail</code>	393–396
<code>\s@GlsXtrIfFieldCmpNum</code>	36	<code>\tabularnewline</code>	389–397, 430, 433–450
<code>\s@GlsXtrIfFieldEqNum</code>	36	<code>\TeX</code>	140
<code>\s@GlsXtrIfFieldEqStr</code>	39	<code>\texorpdfstring</code>	34, 37, 38, 103–105, 322, 333–339
<code>\s@GlsXtrIfFieldEqXpStr</code>	40	<code>\textbf</code>	397, 430
<code>\s@GlsXtrIfFieldNonZero</code>	36	<code>textcase</code> package	320
<code>\s@glsxtrifhasfield</code>	35, 40, 41	<code>\textsc</code>	244
<code>\s@GlsXtrIfXpFieldEqXpStr</code>	40	<code>\textsmaller</code>	258
<code>\s@GlsXtrStartUnsetBuffering</code>	107	<code>\texttt</code>	27, 139–142
<code>\s@GlsXtrStopUnsetBuffering</code>	107	<code>\the</code>	122, 137, 144, 195, 196, 208, 227–234, 236, 238–241, 243–249, 251, 252, 255–257, 259–262, 264, 266, 269, 271, 273–281, 283, 285, 286, 288, 290–292, 295, 296, 298–303, 305–308, 310–315, 317–320, 345
<code>\s@printunsrtglossary</code>	149, 151	<code>\theglsentrycounter</code> ..	8, 10–12, 75, 76, 158
<code>\seealsoname</code>	54, 55	<code>\theH</code>	30
<code>\seename</code>	51	<code>\theHglentrycounter</code> .	8, 10–12, 75, 76, 158
<code>\setabbreviationstyle</code>	123, 229, 237	<code>\theindex</code>	191
<code>\setacronymstyle</code>	123, 124	<code>\thewrglossary</code>	26
<code>\setentrycounter</code>	136, 343, 345	<code>\this@dialect</code>	340, 384
<code>\SetGenericNewAcronym</code>	123	<code>\thisgrptitle</code>	426
<code>\setglossarystyle</code>	25, 128, 386, 388, 389, 399, 401, 402, 414–420, 423	<code>\toks@</code>	137, 195, 196, 345
<code>\setkeys</code>	9, 25, 29, 33, 74, 76, 93, 122, 128, 158, 206, 207	<code>\toprule</code>	430, 433, 435, 436, 438, 439, 441, 442, 444, 445, 447, 448
<code>\setlength</code>	65, 66, 398, 400, 402, 403, 414, 416–418, 423, 431, 432	<code>\TrackedDialectClosestSubMatch</code> ..	41, 42
<code>\settowidth</code>	124, 404–413, 430, 431	<code>tracklang</code> package	41, 145, 354
<code>\setupglossaries</code>	5, 29	<code>\TrackLangGetDefaultScript</code>	383
<code>\sfcode</code>	19, 200, 397	<code>\TrackLangIfHasDefaultScript</code>	383
<code>\small</code>	27, 53	<code>\TrackLangRequireDialectPrefix</code>	383, 384
<code>soul</code> package	107	<code>\triangleright</code>	53
<code>\space</code> ...	6, 13, 54, 60, 61, 64, 93, 109–111, 117–121, 123–128, 139, 141, 143, 146, 150, 152, 155, 200, 201, 205, 209, 229, 386, 387, 397, 400, 401, 403, 404, 413, 421		
<code>\spacefactor</code>	19, 200, 207, 397		
<code>\stepcounter</code>	163		

U		X	
\u	340	\x	162, 345, 350
\undef	16, 58, 197	\xcapitalisewords	183
\underline	198	\xdef	129, 151, 341
\unskip	46, 58, 386	\xifinlist	116
upgreek package	353	\xifinlistcs	35
\usepackage	141, 142	xindy	451
W		xindy	16, 124, 451
\warn@nomakeglossaries	126	xkeyval package	5
\warn@noprintglossary	126, 129	\XKV@checkchoice	68
wrglossary (counter)	25, 26	\XKV@plfalse	68
\write	54, 111, 120, 125, 138, 139, 146	\XKV@resa	68
		\XKV@sttrue	68