

glossaries-extra.sty v1.37: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-11-30

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)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/11/30 1.37 (NLCT)]
```

Requires xkeyval to define package options.

```
3 \RequirePackage{xkeyval}
```

Requires etoolbox package.

```
4 \RequirePackage{etoolbox}
```

Has glossaries already been loaded?

```
5 \@ifpackageloaded{glossaries}
```

```
6 {%
```

Already loaded so pass any options to \setupglossaries. This means that the options that can only be set when glossaries is loaded can't be used.

```
7 \newcommand{\glstr@dooption}[1]{\setupglossaries{#1}}%
```

```
8 \let\@glstr@declareoption\@gls@declareoption
```

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

```
11 \newcommand{\glstr@dooption}[1]{%
```

```
12 \PassOptionsToPackage{#1}{glossaries}%
```

```
13 }%
```

Set the defaults.

```
14 \PassOptionsToPackage{toc}{glossaries}
```

```
15 \PassOptionsToPackage{nopostdot}{glossaries}
```

```
16 \PassOptionsToPackage{noredefwarn}{glossaries}
```

```
17 \@ifpackageloaded{polyglossia}%
```

```
18 {}%
```

```
19 {%
```

```
20 \@ifpackageloaded{babel}%
```

```
21 {\PassOptionsToPackage{translate=babel}{glossaries}}%
```

```
22 {}%
```

```
23 }%
```

```
24 \newcommand*{\@glstr@declareoption}[2]{%
```

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

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

```
27 }
```

```
28 }
```

Declare package options.

`\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@sentries\glstr@do@redef@for\glstr@sentries
71    \or
72      \let\glstrundefaction\glstr@err@undefaction
73      \let\glstr@warnonexistssordo\glstr@gobble
74      \let\glstr@redef@for\glstr@sentries\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   \begin{group}
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 \end{group}

```

`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@@do@@wrglossary{#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}

```

glxtr@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
179 \edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
180 {\thegl@sentrycounter}{\theHgl@sentrycounter}%
181 }%
182 \@do@gl@getcounterprefix
183 \fi

```

Don't protect the `\@gl@recordloc` 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.

```

184 \ifx\@gl@xtr@record@setting\@gl@xtr@record@setting@nameref
185 \@gl@xtr@do@nameref@record
186 {\@gl@label}{\@gl@counterprefix}{\@gl@counter}{\@gl@numberformat}%
187 {\@gl@recordloc}%
188 \else
189 \protected@write\@auxout{}{\string\@gl@xtr@record
190 {\@gl@label}{\@gl@counterprefix}{\@gl@counter}{\@gl@numberformat}%
191 {\@gl@recordloc}}%
192 \fi
193 \@gl@xtr@counterrecordhook
194 \let\@gl@counterprefix\@gl@xtr@orgprefix
195 }

```

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

```

196 \newcommand*\@gl@xtr@dorecordnodefer{%
197 \ifx\thegl@sentrycounter\theHgl@sentrycounter
198 \ifx\@gl@xtr@record@setting\@gl@xtr@record@setting@nameref
199 \@gl@xtr@do@nameref@record
200 {\@gl@label}{\@gl@counter}{\@gl@numberformat}%
201 {\thegl@sentrycounter}%
202 \else
203 \protected@write\@auxout{}{\string\@gl@xtr@record
204 {\@gl@label}{\@gl@counter}{\@gl@numberformat}%
205 {\thegl@sentrycounter}}%
206 \fi
207 \else
208 \edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
209 {\thegl@sentrycounter}{\theHgl@sentrycounter}%
210 }%
211 \@do@gl@getcounterprefix
212 \ifx\@gl@xtr@record@setting\@gl@xtr@record@setting@nameref
213 \@gl@xtr@do@nameref@record
214 {\@gl@label}{\@gl@counterprefix}{\@gl@counter}%
215 {\@gl@numberformat}{\thegl@sentrycounter}%
216 \else
217 \protected@write\@auxout{}{\string\@gl@xtr@record
218 {\@gl@label}{\@gl@counterprefix}{\@gl@counter}{\@gl@numberformat}%

```

```

219         {\theglsentrycounter}}}%
220     \fi
221 \fi
222 \@glsxtr@counterrecordhook
223 }

```

`xtr@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.

```

224 \newcommand{\@glsxtr@ifnum@mmode}[2]{%
225   \ifmmode
226     \ifst@rred
227     #2%
228   \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`.

```

229     \if@display #1\else #2\fi
230   \fi
231 \else
232   #2%
233 \fi
234 }

```

`@nameref@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`.

```

235 \newcommand*{\@glsxtr@do@nameref@record}[5]{%
236   \gls@ifnotmeasuring
237   {%
238     \protected@write\@auxout{}\string\glsxtr@record@nameref
239     {#1}{#2}{#3}{#4}{#5}%
240     {\csuse{@currentlabelname}}{\csuse{@currentHref}}}%
241     {\theHglentrycounter}}}%
242   }%
243 }

```

`r@recordcounter`

```

244 \newcommand*{\@glsxtr@recordcounter}{%
245   \@glsxtr@noop@recordcounter
246 }

```

`p@recordcounter`

```

247 \newcommand*{\@glxtr@noop@recordcounter}[1]{%
248   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
249     requires record=only or record=alsoindex package option}{}%
250 }

p@recordcounter

251 \newcommand*{\@glxtr@op@recordcounter}[1]{%
252   \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
253 }

lsxtr@recordsee Deal with \glssee in record mode. (This doesn't increment the associated counter.)
254 \newcommand*{\@glxtr@recordsee}[2]{%
255   \@@glxtrwrglossmark
256   \def\@gls@xref{#2}%
257   \@onelevel@sanitize\@gls@xref
258   \protected@write\@auxout{}\string\glxtr@recordsee{#1}{\@gls@xref}}%
259 }

srtglossaryunit

260 \newcommand{\printunsrtglossaryunit}{%
261   \print@noop@unsrtglossaryunit
262 }

tr@setup@record Initialise.
263 \newcommand*{\glxtr@setup@record}{\let\@do@wrglossary\glxtr@do@wrglossary}

aveentrycounter Only store the entry counter information if the indexing is on.
264 \newcommand*{\glxtr@indexonly@saveentrycounter}{%
265   \ifKV@glslink@noindex
266   \else
267     \glxtr@saveentrycounter
268   \fi
269 }

addloclistfield

270 \newcommand*{\glxtr@addloclistfield}{%
271   \key@ifundefined{glossentry}{loclist}%
272   {%
273     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
274     \appto\@gls@keymap{,{loclist}{loclist}}%
275     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
276     \appto\@newglossaryentryposthook{%
277       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
278     }%
279     \glssetnoexpandfield{loclist}%
280   }%
281   {}}%

```

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

```

282 \key@ifundefined{glossentry}{location}%
283 {%
284   \define@key{glossentry}{location}{\def\@glo@location{##1}}%
285   \appto\@gls@keymap{,{location}{location}}%
286   \appto\@newglossaryentryprehook{\def\@glo@location{}}%
287   \appto\@newglossaryentryposthook{%
288     \gls@assign@field}{\@glo@label}{location}{\@glo@location}%
289   }%
290   \glssetnoexpandfield{location}%
291 }%
292 {}%
```

Add a key to store the group heading.

```

293 \key@ifundefined{glossentry}{group}%
294 {%
295   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
296   \appto\@gls@keymap{,{group}{group}}%
297   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
298   \appto\@newglossaryentryposthook{%
299     \gls@assign@field}{\@glo@label}{group}{\@glo@group}%
300   }%
301   \glssetnoexpandfield{group}%
302 }%
303 {}%
304 }
```

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

```

305 \newcommand*{\@glsxtr@record@setting}{off}
```

alsoindex

```

306 \newcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}
```

only

```

307 \newcommand*{\@glsxtr@record@setting@only}{only}
```

nameref

```

308 \newcommand*{\@glsxtr@record@setting@nameref}{nameref}
```

ifrecordonly

```

309 \newcommand*{\@glsxtr@if@record@only}[2]{%
310   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
311     #1%
312   \else
313     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
314       #1%
315     \else
316       #2%
317     \fi
}
```

```

318 \fi
319 }

cord@setting@off
320 \newcommand*{\@glxtr@record@setting@off}{off}

cord@only@setup Initialisation code for record=only and record=nameref
321 \newcommand*{\@glxtr@record@only@setup}{%
322 \def\glxtr@setup@record{%
323 \glxtr@autoseeindexfalse
324 \let\do@seeglossary\glxtr@recordsee
325 \let\glxtr@record\@glxtr@record
326 \let\do@wrglossary\glxtr@do@record@wrglossary
327 \let\glxtr@saveentrycounter\relax
328 \let\glxtrundefaction\glxtr@warn@undefaction
329 \let\glxtr@warnonexistsordo\glxtr@warn@onexistsordo
330 \glxtr@addloclistfield
331 \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
332 \let\@glxtr@recordcounter\glxtr@op@recordcounter
333 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

Switch off the index suppression for aliased entries. (bib2gls will deal with them.)
334 \def\glxtrsetaliasnoindex{}%

\@glxtr@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.
335 \ifdefined\@glxtr@setupsort@none{\@glxtr@setupsort@none}{}%

Warn about using \printglossary:
336 \def\glxtrNoGlossaryWarning{\@glxtr@record@noglossarywarning}%

Load glossaries-extra-bib2gls:
337 \RequirePackage{glossaries-extra-bib2gls}%
338 }%
339 }

record Now define the record package option.
340 \define@choicekey{glossaries-extra.sty}{record}
341 [ \glxtr@record@setting\glxtr@record@nr ]%
342 {off,only,alsoindex,nameref}%
343 [only]%
344 {%
345 \ifcase\glxtr@record@nr\relax

Don't record.
346 \def\glxtr@setup@record{%
347 \renewcommand*{\do@seeglossary}{\glxtr@doseeglossary}%
348 \renewcommand*{\glxtr@record}[3]{}%
349 \let\do@wrglossary\glxtr@do@wrglossary
350 \let\glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter

```

```

351 \let\glxstrundefaction\@glxstr@err@undefaction
352 \let\glxstr@warnonexistssordo\@gobble
353 \let\@glxstr@recordcounter\@glxstr@noop@recordcounter
354 \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
355 \undef\glxstrsetaliasnoindex
356 }%
357 \or

```

Only record (don't index).

```

358 \@glxstr@record@only@setup
359 \or

```

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

```

360 \def\glxstr@setup@record{%
361 \renewcommand*{\@do@seeglossary}{\@glxstr@dosee@alsoindex@glossary}%
362 \let\@glxstr@record\@glxstr@record
363 \let\@do@wrglossary\glxstr@do@alsoindex@wrglossary
364 \let\@glxstr@saveentrycounter\glxstr@indexonly@saveentrycounter
365 \let\glxstrundefaction\@glxstr@warn@undefaction
366 \let\glxstr@warnonexistssordo\@glxstr@warn@onexistssordo
367 \glxstr@addloclistfield
368 \let\@glxstr@recordcounter\@glxstr@op@recordcounter
369 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
370 \undef\glxstrsetaliasnoindex
371 }%
372 \or

```

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

```

373 \@glxstr@record@only@setup
374 \ifundef\hyperlink
375 {\GlossariesExtraWarning{You have requested record=nameref but
376 the document doesn't support hyperlinks}}%
377 }%
378 \fi
379 }

```

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.

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

```

380 \newcommand*{\@glxstr@docdefval}{0}

```

Need to provide conditional commands that are backward compatible:

`\if@glxstrdocdef`

```

381 \newcommand*{\if@glxstrdocdef}{\ifnum\@glxstr@docdefval>0 }

```

`\glxstrdocdeftrue`

```

382 \newcommand*{\@glxstrdocdeftrue}{\def\@glxstr@docdefval{1}}

```

sxtrdocdeffalse

```
383 \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.

```
384 \define@choicekey{glossaries-extra.sty}{docdef}{%
385   [\@glxtr@docdefsetting\@glxtr@docdefval]%
386   {false,true,restricted,atom}[true]%
387   {%
388     \ifnum\@glxtr@docdefval>1\relax
389       \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
390     \else
391       \renewcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}%
392     \fi
393 }
```

ocdefrestricted

```
394 \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).

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

indexcrossrefs Automatically index cross references at the end of the document

```
396 \define@boolkey{glossaries-extra.sty}[@glxtr]{indexcrossrefs}[true]{%
397   \if@glxtrindexcrossrefs
398   \else
399     \renewcommand*{\@glxtr@autoindexcrossrefs}{}}%
400 \fi
401 }
```

Switch off since this can increase the build time.

```
402 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
403 \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.

```
404 \define@boolkey{glossaries-extra.sty}[@glxtr@]{autoseeindex}[true]{%
405 }
406 \@glxtr@autoseeindextrue
```


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

```

407 \define@boolkey{glossaries-extra.sty}[@glsxtr@]{equations}[true]{%
408 }
409 \@glsxtr@equationsfalse

\glsxtr@float
410 \let\glsxtr@float\@float

glsxtr@dblfloat
411 \let\glsxtr@dblfloat\@dblfloat

floats Provide a boolean option to automatically switch to the the corresponding counter when in a float.
412 \define@boolkey{glossaries-extra.sty}[@glsxtr@]{floats}[true]{%
413   \if@glsxtr@floats
414     \renewcommand*{\@float}[1]{\renewcommand{\glscounter}{##1}\glsxtr@float{##1}}%
415     \renewcommand*{\@dblfloat}[1]{\renewcommand{\glscounter}{##1}\glsxtr@dblfloat{##1}}%
416   \else
417     \let\@float\glsxtr@float
418     \let\@dblfloat\glsxtr@dblfloat
419   \fi
420 }
421 \@glsxtr@floatsfalse

iesExtraWarning Allow users to suppress warnings.
422 \newcommand*{\GlossariesExtraWarning}[1]{\PackageWarning{glossaries-extra}{#1}}

raWarningNoLine Allow users to suppress warnings.
423 \newcommand*{\GlossariesExtraWarningNoLine}[1]{%
424   \PackageWarningNoLine{glossaries-extra}{#1}}

425 \@glsxtr@declareoption{nowarn}{%
426   \let\GlossariesExtraWarning\@gobble
427   \let\GlossariesExtraWarningNoLine\@gobble
428   \glsxtr@doption{nowarn}%
429 }

xtr@defpostpunc Redefines \glspostdescription. The postdot and nopostdot options will have to redefine this.
430 \newcommand*{\@glsxtr@defpostpunc}{%

postdot Shortcut for nopostdot=false
431 \@glsxtr@declareoption{postdot}{%
432   \glsxtr@doption{nopostdot=false}%
433   \renewcommand*{\@glsxtr@defpostpunc}{%
434     \renewcommand*{\glspostdescription}{%
435       \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%

```

```

436 }%
437 }

```

nopostdot Needs to redefine \@glsxtr@defpostpunc

```

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

```

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

```

445 \define@key{glossaries-extra.sty}{postpunc}{%
446   \glsxtr@doption{nopostdot=false}%
447   \ifstrequal{#1}{dot}%
448   {%
449     \renewcommand*{\@glsxtr@defpostpunc}{%
450       \renewcommand*{\glspostdescription}{.\spacefactor\sfcode'\. }%
451     }%
452   }%
453   {%
454     \ifstrequal{#1}{comma}%
455     {%
456       \renewcommand*{\@glsxtr@defpostpunc}{%
457         \renewcommand*{\glspostdescription}{,}%
458       }%
459     }%
460     {%
461       \ifstrequal{#1}{none}%
462       {%
463         \glsxtr@doption{nopostdot=true}%
464         \renewcommand*{\@glsxtr@defpostpunc}{%
465           \renewcommand*{\glspostdescription}{}%
466         }%
467       }%
468       {%
469         \renewcommand*{\@glsxtr@defpostpunc}{%
470           \renewcommand*{\glspostdescription}{#1}%
471         }%
472       }%
473     }%
474   }%
475 }

```

glsxtrabbrvtype Glossary type for abbreviations.

```

476 \newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}

```

bbreviationsdef Set by abbreviations option.

```
477 \newcommand*{\@glxtr@abbreviationsdef}{}%
```

bbreviationsdef

```
478 \newcommand*{\@glxtr@doabbreviationsdef}{%
479   \ifpackageloaded{babel}%
480   {\providecommand{\abbreviationsname}{\acronymname}}%
481   {\providecommand{\abbreviationsname}{Abbreviations}}%
482   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
483   \renewcommand*{\glxtrabbrvtype}{abbreviations}%
484   \newcommand*{\printabbreviations}[1][1]{%
485     \printglossary[type=\glxtrabbrvtype,##1]%
486   }%
487   \disable@keys{glossaries-extra.sty}{abbreviations}%
```

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

```
488   \ifglsacronym
489   \else
490     \renewcommand*{\acronymtype}{\glxtrabbrvtype}%
491   \fi
492 }%
```

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

```
493 \@glxtr@declareoption{abbreviations}{%
494   \let\@glxtr@abbreviationsdef\@glxtr@doabbreviationsdef
495 }
```

iationShortcuts 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).

```
496 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
497   \newcommand*{\ab}{\cglsl}%
498   \newcommand*{\abp}{\cglspl}%
499   \newcommand*{\as}{\glxtrshort}%
500   \newcommand*{\asp}{\glxtrshortpl}%
501   \newcommand*{\al}{\glxtrlong}%
502   \newcommand*{\alp}{\glxtrlongpl}%
503   \newcommand*{\af}{\glxtrfull}%
504   \newcommand*{\afp}{\glxtrfullpl}%
505   \newcommand*{\Ab}{\cGls}%
506   \newcommand*{\Abp}{\cGlspl}%
507   \newcommand*{\As}{\Glsxtrshort}%
508   \newcommand*{\Asp}{\Glsxtrshortpl}%
509   \newcommand*{\Al}{\Glsxtrlong}%
510   \newcommand*{\Alp}{\Glsxtrlongpl}%
511   \newcommand*{\Af}{\Glsxtrfull}%
512   \newcommand*{\Afp}{\Glsxtrfullpl}%
513   \newcommand*{\AB}{\cGLS}%
514   \newcommand*{\ABP}{\cGLSpl}%
```

```

515 \newcommand*{\AS}{\GLSxtrshort}%
516 \newcommand*{\ASP}{\GLSxtrshortpl}%
517 \newcommand*{\AL}{\GLSxtrlong}%
518 \newcommand*{\ALP}{\GLSxtrlongpl}%
519 \newcommand*{\AF}{\GLSxtrfull}%
520 \newcommand*{\AFP}{\GLSxtrfullpl}%

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

  Disable this command after it's been used.
522 \let\GlsXtrDefineAbbreviationShortcuts\relax
523 }

```

`\defineAcShortcuts` Enable shortcut commands for the abbreviations, but uses the analogous commands provided by glossaries.

```

524 \newcommand*{\GlsXtrDefineAcShortcuts}{%
525   \newcommand*{\ac}{\cgl}%
526   \newcommand*{\acp}{\cglsp}%
527   \newcommand*{\acs}{\glxtrshort}%
528   \newcommand*{\acsp}{\glxtrshortpl}%
529   \newcommand*{\acl}{\glxtrlong}%
530   \newcommand*{\aclp}{\glxtrlongpl}%
531   \newcommand*{\acf}{\glxtrfull}%
532   \newcommand*{\acfp}{\glxtrfullpl}%
533   \newcommand*{\Ac}{\cGls}%
534   \newcommand*{\Acp}{\cGlspl}%
535   \newcommand*{\Acs}{\Glsxtrshort}%
536   \newcommand*{\Acsp}{\Glsxtrshortpl}%
537   \newcommand*{\Acl}{\Glsxtrlong}%
538   \newcommand*{\Ac lp}{\Glsxtrlongpl}%
539   \newcommand*{\Acf}{\Glsxtrfull}%
540   \newcommand*{\Acfp}{\Glsxtrfullpl}%
541   \newcommand*{\AC}{\cGLS}%
542   \newcommand*{\ACP}{\cGLSpl}%
543   \newcommand*{\ACS}{\GLSxtrshort}%
544   \newcommand*{\ACSP}{\GLSxtrshortpl}%
545   \newcommand*{\ACL}{\GLSxtrlong}%
546   \newcommand*{\ACLP}{\GLSxtrlongpl}%
547   \newcommand*{\ACF}{\GLSxtrfull}%
548   \newcommand*{\ACFP}{\GLSxtrfullpl}%

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

  Disable this command after it's been used.
550   \let\GlsXtrDefineAcShortcuts\relax
551 }

```

`\defineOtherShortcuts` Similarly provide shortcut versions for the commands provided by the symbols and numbers options.

```

552 \newcommand*{\GlsXtrDefineOtherShortcuts}{%

```

```

553 \newcommand*{\newentry}{\newglossaryentry}%
554 \ifdef\printsymbols
555 {%
556   \newcommand*{\newsym}{\glstrnewsymbol}%
557 }{}%
558 \ifdef\printnumbers
559 {%
560   \newcommand*{\newnum}{\glstrnewnumber}%
561 }{}%
562 \let\GlsXtrDefineOtherShortcuts\relax
563 }

```

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.

```
564 \newcommand*{\@glstr@setupshortcuts}{}%
```

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

```
565 \newcommand*{\@glstr@shortcutsval}{\ifglscrshortcuts 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).

```

566 \define@choicekey{glossaries-extra.sty}{shortcuts}%
567 [\@glstr@shortcutsval\@glstr@shortcutsnr]%
568 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
569   \ifcase\@glstr@shortcutsnr\relax % acronyms
570     \renewcommand*{\@glstr@setupshortcuts}{%
571       \glscrshortcuttrue
572       \DefineAcronymSynonyms
573     }%
574   \or % acro
575     \renewcommand*{\@glstr@setupshortcuts}{%
576       \glscrshortcuttrue
577       \DefineAcronymSynonyms
578     }%
579   \or % abbreviations
580     \renewcommand*{\@glstr@setupshortcuts}{%
581       \GlsXtrDefineAbbreviationShortcuts
582     }%
583   \or % abbr
584     \renewcommand*{\@glstr@setupshortcuts}{%
585       \GlsXtrDefineAbbreviationShortcuts
586     }%
587   \or % other

```

```

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

```

Leave none and false as last option.

```

610 \else % none, false
611 \renewcommand*{\@glxtr@setupshortcuts}{}%
612 \fi
613 }

```

lsxtr@doaccsupp

```

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

```

accsupp If accsupp, load glossaries-accsupp package.

```

615 \@glxtr@declareoption{accsupp}{%
616 \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

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

```

617 \newcommand{\@glxtrNoGlossaryWarning}[1]{%
618 \GlossariesExtraWarning{Glossary '#1' is missing}%
619 \@glxtr@defaultnoglossarywarning{#1}%
620 }

```

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

```

621 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
622 [ \@glxtr@nomissingglstextval \@glxtr@nomissingglstextnr ]%

```

```

623 {true,false}[true]{%
624   \ifcase\@glxtr@nomissingglstextnr\relax % true
625     \renewcommand{\glxtrNoGlossaryWarning}[1]{\null}%
626   \else % false
627     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
628       \glxtr@defaultnoglossarywarning{#1}%
629     }%
630   \fi
631 }

```

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

xtr@redefstyles

```

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

```

stylemods

```

633 \define@key{glossaries-extra.sty}{stylemods}[default]{%
634   \ifstrequal{#1}{default}%
635   {%
636     \renewcommand*{\@glxtr@redefstyles}{%
637       \RequirePackage{glossaries-extra-stylemods}}%
638   }%
639   {%
640     \ifstrequal{#1}{all}%
641     {%
642       \renewcommand*{\@glxtr@redefstyles}{%
643         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
644         \RequirePackage{glossaries-extra-stylemods}%
645       }%
646     }%
647     {%
648       \renewcommand*{\@glxtr@redefstyles}{}%
649       \@for\@glxtr@tmp:=#1\do{%
650         \IfFileExists{glossary-\@glxtr@tmp.sty}%
651         {%
652           \eappto\@glxtr@redefstyles{%
653             \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
654         }%
655         {%
656           \PackageError{glossaries-extra}%
657             {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
658              doesn’t exist (did you mean to use the ‘style’ key?)}%
659             {The list of values (#1) in the ‘stylemods’ key should
660              match the glossary-xxx.sty files provided with
661              glossaries.sty}%
662         }%
663       }%
664       \appto\@glxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
665     }

```

```

666 }%
667 }

```

glsxtr@do@style

```

668 \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.

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

672 \setglossarystyle{#1}%

```

```

673 }%

```

```

674 }

```

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.

```

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

```

ocationHyperlink

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

The first two arguments are always control sequences.

```

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

```

```

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

```

```

678 }

```

cationhyperlink

```

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

```

```

680 \pageref{wrglossary.#3}%

```

```

681 }

```

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.

```

682 \@glsxtr@declareoption{indexcounter}{%

```



```

683 \glsxtr@doption{counter=wr glossary}%
684 \ifundef\c@wr glossary
685 {%
686   \newcounter{wr glossary}%
687   \renewcommand{\thewr glossary}{\arabic{wr glossary}}%
688 }%
689 {}%
690 \renewcommand*{\glsxtr@inc@wr glossaryctr}[1]{%
  Only increment if the current counter is wr glossary.
691   \ifdefstring\@gls@counter{wr glossary}%
692   {%
693     \refstepcounter{wr glossary}%
694     \label{wr glossary.\thewr glossary}%
695   }%
696   {}%
697 }%
698 \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
699   \ifdefstring\glsentrycounter{wr glossary}%
700   {%
701     \@glsxtr@wr glossary@locationhyperlink{##1}{##2}{##3}%
702   }%
703   {\glsxtrhyperlink{##1##2##3}{##3}}%
704 }%
705 }

```

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

```
706 \newcommand*{\@glsxtrwrglossmark}{}

```

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

```

707 \newcommand*{\@glsxtrwrglossmark}{}
708 \AtBeginDocument{\renewcommand*{\@glsxtrwrglossmark}{\@glsxtrwrglossmark}}

```

`\sxtwrwrglossmark` Does nothing by default.

```
709 \newcommand*{\glsxtrwrglossmark}{\ensuremath{\cdotp}}

```

`debug` Provide extra debug options.

```

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

```

```

721 \or % showwrgloss
722 \glxstr@doption{debug=true}%
723 \renewcommand*{\@glxtrwrglossmark}{\glxtrwrglossmark}%
724 \or % all
725 \glxstr@doption{debug=showtargets}%
726 \renewcommand*{\@glxtrwrglossmark}{\glxtrwrglossmark}%
727 \fi
728 }

```

Pass all other options to glossaries.

```

729 \DeclareOptionX*{%
730 \expandafter\glxstr@doption\expandafter{\CurrentOption}}

```

Process options.

```
731 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
732 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
733 \@glxstr@doaccsupp
```

Redefine \glspostdescription if required.

```
734 \@glxstr@defpostpunc
```

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

```

735 \def\glsshowtarget#1{%
736 \glxstrtitleorpdforheading
737 {%
738 \ifmmode
739 \texttt{\small [#1]}%
740 \else
741 \ifinner
742 \texttt{\small [#1]}%
743 \else
744 \marginpar{\texttt{\small #1}}%
745 \fi
746 \fi
747 }%
748 {[#1]}%
749 {\texttt{\small [#1]}}%
750 }

```

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

```
751 \let\@glxstr@org@doseeglossary\@do@seeglossary
```

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

```

752 \newcommand*{\@glxstr@doseeglossary}[2]{%
753 \glsdofexists{#1}%

```

```

754  {%
755    \@glsxtrwrglossmark
756    \@glsxtr@org@doseeglossary{#1}{#2}%
757  }%
758 }

oindex@glossary
759 \newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
760   \@glsxtr@recordsee{#1}{#2}%
761   \@glsxtr@doseeglossary{#1}{#2}%
762 }

@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.)
763 \let\@glsxtr@org@gloautosee\@glo@autosee

                  Check if user tried autoseeindex=false when it can't be supported.
764 \if@glsxtr@autoseeindex
765 \else
766   \ifdef\@glsxtr@org@gloautosee
767   {%
768     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
769       option requires at least v4.30 of glossaries.sty}%
770     {You need to update the glossaries.sty package}%
771   }
772 \fi

\@glo@autosee  If \@glo@autosee has been defined (glossaries v4.30 onwards), redefine it to test the au-
               toseeindex option.
773 \ifdef\@glo@autosee
774 {%
775   \renewcommand*{\@glo@autosee}{%
776     \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
777 }%
778 {}

checkseeallowed  Don't prohibit the use of the see key before the indexing files have been opened if the auto-
                  matic see indexing has been disabled, since it's no longer an issue.
779 \renewcommand*{\gls@checkseeallowed}{%
780   \if@glsxtr@autoseeindex\@gls@see@noindex\fi
781 }

                  Define abbreviations glossaries if required.
782 \@glsxtr@abbreviationsdef
783 \let\@glsxtr@abbreviationsdef\relax

                  Setup shortcuts if required.
784 \@glsxtr@setupshortcuts

```

Redefine `\@glxtr@redef@for@gl@sentries` if required.

```

785 \@glxtr@redef@for@gl@sentries

```

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

```

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

```

Now define the user command:

```

787 \newcommand*{\glossariesextrasetup}[1]{%
788   \let\glxtr@setup@record\relax
789   \let\glxtr@setupshortcuts\relax
790   \let\glxtr@redef@for@gl@sentries\relax
791   \setkeys{glossaries-extra.sty}{#1}%
792   \@glxtr@abbreviationsdef
793   \let\glxtr@abbreviationsdef\relax
794   \@glxtr@setupshortcuts
795   \glxtr@setup@record
796   \@glxtr@redef@for@gl@sentries
797 }

```

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

```

798 \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.

```

799 \newcommand*{\glxtr@@do@wrglossary}[1]{%
800   \@@glxtrwrglossmark
801   \glxtr@inc@wrglossaryctr{#1}%
802   \glxtr@org@@do@wrglossary{#1}%
803 }

```

`saveentrycounter` Save original definition of `\@gl@s@saveentrycounter`.

```

804 \let\glxtr@saveentrycounter\@gl@s@saveentrycounter

```

`saveentrycounter` Change `\@gl@s@saveentrycounter` so that it only stores the entry counter information if the indexing is on.

```

805 \let\@gl@s@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.

```

806 \renewcommand*\@gl@s@getcounterprefix[2]{%
807   \protected@edef\@gl@s@thisloc{#1}\protected@edef\@gl@s@thisHloc{#2}%
808   \ifx\@gl@s@thisloc\@gl@s@thisHloc
809     \def\@glo@counterprefix{}%
810   \else

```

```

811 \def\@gls@get@counterprefix#1.#1##2\end@getprefix{%
812 \def\@glo@tmp{##2}%
813 \ifx\@glo@tmp\@empty
814 \def\@glo@counterprefix{}%
815 \else
816 \def\@glo@counterprefix{##1}%
817 \fi
818 }%
819 \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed, unless record=nameref.

820 \ifx\@glo@counterprefix\@empty
821 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
822 \else
823 \GlossariesExtraWarning{Hyper target ‘#2’ can’t be formed by
824 prefixing^^Jlocation ‘#1’. You need to modify the
825 definition of \string\theH\@gls@counter^^Jotherwise you
826 will get the warning: “name{\@gls@counter.#1}’ has been^^J
827 referenced but does not exist”%
828 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
829 . You may want to consider using record=nameref instead%
830 \fi}%
831 \fi
832 \fi
833 \fi
834 }

```

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

sxtrdialecthook

```
835 \newcommand*{\@glsxtrdialecthook}{}

```

Set up record option if required.

```
836 \glsxtr@setup@record

```

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

```

837 \AtBeginDocument{%
838 \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
839 \def\@glsxtrundeftag{\glsxtrundeftag}%
840 }

```

1.2 Extra Utilities

nusedOrUndefined

```
\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).

```

841 \newcommand*{\GlsXtrIfUnusedOrUndefined}[3]{%
842   \ifglssentryexists{#1}%
843   {\ifbool{glo@\glsdetoklabel{#1}@flag}{#3}{#2}}%
844   {#2}}%
845 }

```

`\glstrifemptyglossary`

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

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

```

846 \newcommand{\glstrifemptyglossary}[3]{%
847   \ifcsdef{glolist@#1}%
848   {%
849     \ifcsstring{glolist@#1}{,}{#2}{#3}%
850   }%
851   {%
852     \glstrundefaction{Glossary type '#1' doesn't exist}{}%
853     #2%
854   }%
855 }

```

`\glstrifkeydefined`

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

```

856 \newcommand*{\glstrifkeydefined}[3]{%
857   \key@ifundefined{glossentry}{#1}{#3}{#2}%
858 }

```

`\glstrprovidestoragekey`

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

```

859 \newcommand*{\glstrprovidestoragekey}{%
860   \@ifstar\@sglsxtr@provide@storagekey\@glstr@provide@storagekey
861 }

```

`\glstrprovide@storagekey`

Unstarred version.

```

862 \newcommand*{\@glstr@provide@storagekey}[3]{%
863   \key@ifundefined{glossentry}{#1}%
864   {%
865     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
866     \appto\@gls@keymap{,{#1}{#1}}%
867     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
868     \appto\@newglossaryentryposthook{%
869       \letcs{\@glo@tmp}{@glo@#1}%

```

```

870     \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
871 }%

```

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

```

872     \ifblank{#3}
873     {}%
874     {%
875         \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
876     }%
877 }%
878 {%

```

Provide the no-link command if not already defined.

```

879     \ifblank{#3}
880     {}%
881     {%
882         \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
883     }%
884 }%
885 }

```

`\provide@storagekey` Starred version.

```

886 \newcommand*\s@glstr@provide@storagekey}[1]{%
887   \key@ifundefined{glossentry}{#1}%
888   {%
889     \expandafter\newcommand\expandafter*\expandafter
890     {\csname gls@assign@#1@field\endcsname}[2]{%
891       \@gls@expand@field{##1}{#1}{##2}%
892     }%
893   }%
894   {}%
895   \@glstr@provide@addstoragekey{#1}%
896 }

```

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`

```

897 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

898 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```

`\@glsxtrfmt` Unstarred form.

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

`\s@glsxtrfmt` Starred form.

```
901 \newcommand*{\s@glsxtrfmt}[3] [] {%
902   \new@ifnextchar[{\s@@glsxtrfmt{#1}{#2}{#3}}}%
903   {\@glsxtrfmt{#1}{#2}{#3}{}}}%
904 }
```

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

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

`\@@glsxtrfmt` Actual inner working.

```
906 \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).

```
907   \begingroup
908     \def\glslabel{#2}%
909     \glsdoifexistsordo{#2}%
910     {%
911       \ifglshasfield{\GlsXtrFmtField}{#2}%
912       {%
913         \let\do@glsl@link@checkfirsthyper\relax
914         \expandafter\@glsl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
915         {\glsxtrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}}%
916       }%
917       {\glsxtrfmtdisplay{@firstofone}{#3}{#4}}}%
918     }%
919     {%
```

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

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

`\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).

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


`\glxxtrentryfmt` No link or indexing.

```

930 \ifdef\teorpdfstring
931 {
932   \newcommand*\glxxtrentryfmt}[2]{%
933     \teorpdfstring{\@glxxtrentryfmt{#1}{#2}}{#2}%
934   }
935 }
936 {
937   \newcommand*\glxxtrentryfmt{\@glxxtrentryfmt}
938 }

```

`@glxxtrentryfmt`

```

939 \newrobustcmd*\@glxxtrentryfmt}[2]{%
940   \glsoifexistsordo{#1}%
941   {%
942     \ifglshasfield{\GlsXtrFmtField}{#1}%
943     {%
944       \csuse{\glscurrentfieldvalue}{#2}%
945     }%
946     {#2}%
947   }%
948   {#2}%
949 }

```

`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.

```

950 \newcommand*\glxxttrfieldlistadd}[3]{%
951   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
952 }

```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```

953 \newcommand*\glxxttrfieldlistgadd}[3]{%
954   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
955 }

```

`trfieldlistead` Similarly but uses `\listcseadd`.

```

956 \newcommand*\glxxttrfieldlistead}[3]{%
957   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
958 }

```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```

959 \newcommand*\glxxttrfieldlistxadd}[3]{%
960   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
961 }

```

Now provide commands to iterate over these lists.

fielddolistloop

```
962 \newcommand*\glxtrfielddolistloop}[2]{%
963   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
964 }
```

fieldforlistloop

```
965 \newcommand*\glxtrfieldforlistloop}[3]{%
966   \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
967 }
```

List element tests:

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

```
968 \newcommand*\glxtrfieldifinlist}[5]{%
969   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
970 }
```

rfieldxifinlist Expands item.

```
971 \newcommand*\glxtrfieldxifinlist}[5]{%
972   \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
973 }
```

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

```
974 \newcommand*\glxtrforcsvfield}[3]{%
975   \@glxtrifhasfield{#2}{#1}%
976   {%
977     \let\glxxtrendfor\endfortrue
978     \@for\@glxtr@label:=\glscurrentfieldvalue\do
979     {\expandafter#3\expandafter{\@glxtr@label}}}%
980   }%
981 }
```

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.

```
982 \newrobustcmd{\glxtrifhasfield}{%
983   \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
984 }
```

lsxtrifhasfield Unstarred version adds grouping.

```
985 \newcommand{\@glxtrifhasfield}[4]{%
986   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
987 }
```

lsxtrifhasfield Starred version omits grouping.

```
988 \newcommand{\s@glxtrifhasfield}[4]{%
989   \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
990   \ifundef\glscurrentfieldvalue
991     {#4}%
992     {%
993       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
994     }%
995 }
```

rIfFieldNonZero Designed for numeric fields.

```
996 \newcommand{\GlsXtrIfFieldNonZero}[4]{%
997   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
998 }
```

sXtrIfFieldEqNum $\backslash\text{GlsXtrIfFieldEqNum}\{\langle field \rangle\}\{\langle label \rangle\}\{\langle value \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Designed for numeric fields.

```
999 \newcommand{\GlsXtrIfFieldEqNum}[5]{%
1000   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1001 }
```

XtrIfFieldCmpNum $\backslash\text{GlsXtrIfFieldCmpNum}\{\langle field \rangle\}\{\langle label \rangle\}\{\langle comparison \rangle\}\{\langle value \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Designed for numeric fields.

```
1002 \newcommand{\GlsXtrIfFieldCmpNum}[6]{%
1003   {%
1004     \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
1005     \ifundef\glscurrentfieldvalue
1006       {\def\glscurrentfieldvalue{0}}%
1007     {%
1008       \ifdefempty\glscurrentfieldvalue
1009         {\def\glscurrentfieldvalue{0}}%
1010       }%
1011     }%
1012     \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1013   }%
1014 }
```

sXtrIfFieldUndef $\backslash\text{GlsXtrIfFieldUndef}\{\langle field \rangle\}\{\langle label \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Just uses \ifcsundef.

```
1015 \newcommand{\GlsXtrIfFieldUndef}[2]{%
1016   \ifcsundef{glo@\glsdetoklabel{#2}@#1}%
1017 }
```

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

```
1018 \newcommand*\glstrusefield[2]{%
1019   \@gls@entry@field{#1}{#2}%
1020 }
```

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

```
1021 \ifdef\texorpdfstring
1022 {
1023   \newcommand*\Glsxtrusefield[2]{%
1024     \texorpdfstring
1025       {\@Gls@entry@field{#1}{#2}}
1026       {\@gls@entry@field{#1}{#2}}%
1027   }
1028 }
1029 {
1030   \newcommand*\Glsxtrusefield[2]{%
1031     \@Gls@entry@field{#1}{#2}%
1032   }
1033 }
```

\GLSxtrusefield As above but convert to all caps.

```
1034 \ifdef\texorpdfstring
1035 {
1036   \newcommand*\GLSxtrusefield[2]{%
1037     \texorpdfstring
1038       {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}}%
1039       {\@gls@entry@field{#1}{#2}}%
1040   }
1041 }
1042 {
1043   \newcommand*\GLSxtrusefield[2]{%
1044     \glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}%
1045   }
1046 }
```

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

```
1047 \newcommand*\glstrdeffield[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}
```

glxtredefield Just use \csedef to provide a field value for the given entry.

```
1048 \newcommand*\glxtredefield[2]{\protected@csedef{glo@\glsdetoklabel{#1}@#2}}
```

etfieldifexists

```
1049 \newcommand*\glstrsetfieldifexists[3]{\glsdoifexists{#1}{#3}}
```

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

```
1050 \newrobustcmd*{\GlsXtrSetField}[3]{%
1051   \glstrsetfieldifexists{#1}{#2}%
1052   {\csdef{glo@glstoklabel{#1}@#2}{#3}}%
1053 }
```

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

```
1054 \newrobustcmd*{\GlsXtrLetField}[3]{%
1055   \glstrsetfieldifexists{#1}{#2}%
1056   {\cslet{glo@glstoklabel{#1}@#2}{#3}}%
1057 }
```

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

```
1058 \newrobustcmd*{\csGlsXtrLetField}[3]{%
1059   \glstrsetfieldifexists{#1}{#2}%
1060   {\csletcs{glo@glstoklabel{#1}@#2}{#3}}%
1061 }
```

`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.

```
1062 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
1063   \glstrsetfieldifexists{#1}{#2}%
1064   {\csletcs{glo@glstoklabel{#1}@#2}{glo@glstoklabel{#3}@#4}}%
1065 }
```

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

```
1066 \newrobustcmd*{\gGlsXtrSetField}[3]{%
1067   \glstrsetfieldifexists{#1}{#2}%
1068   {\csgdef{glo@glstoklabel{#1}@#2}{#3}}%
1069 }
```

`xGlsXtrSetField`

```
1070 \newrobustcmd*{\xGlsXtrSetField}[3]{%
1071   \glstrsetfieldifexists{#1}{#2}%
1072   {\protected@csxdef{glo@glstoklabel{#1}@#2}{#3}}%
1073 }
```

`eGlsXtrSetField`

```
1074 \newrobustcmd*{\eGlsXtrSetField}[3]{%
1075   \glstrsetfieldifexists{#1}{#2}%
1076   {\protected@csedef{glo@glstoklabel{#1}@#2}{#3}}%
1077 }
```

`XtrIfFieldEqStr`

```
1078 \newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
```

```

1079 \glstrifhasfield{#1}{#2}%
1080 {%
1081   \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1082   }%
1083   {#5}%
1084 }

```

rIfFieldEqXpStr Like the above but first expands the string.

```

1085 \newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
1086   \glstrifhasfield{#1}{#2}%
1087   {%
1088     \protected@edef\@gls@tmp{#3}%
1089     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1090     }%
1091     {#5}%
1092 }

```

fXpFieldEqXpStr Like the above but also expands the field value.

```

1093 \newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
1094   \glstrifhasfield{#1}{#2}%
1095   {%
1096     \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
1097     \let\glscurrentfieldvalue\@gls@tmp
1098     \protected@edef\@gls@tmp{#3}%
1099     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1100     }%
1101     {#5}%
1102 }

```

lsXtrForeignText `\GlsXtrForeignText{<entry label>}{<text>}`

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`.

```

1103 \ifdef\foreignlanguage
1104 {
1105   \ifdef\GetTrackedDialectFromLanguageTag
1106   {
1107     \newcommand{\GlsXtrForeignText}[2]{%

```

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

```

1108     \let\@glstr@org@currentfieldvalue\glscurrentfieldvalue
1109     \glstrifhasfield{\GlsXtrForeignTextField}{#1}%
1110     {%
1111       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
1112       {\glscurrentfieldvalue}{\@glstr@dialect}%

```

```

1113      \let\@glxtr@locale\glscurrentfieldvalue
1114      \let\glscurrentfieldvalue\@glxtr@org@currentfieldvalue
1115      \ifdefempty\@glxtr@dialect
1116      {%

```

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

```

1117      \ifundef\TrackedDialectClosestSubMatch
1118      {%
1119      \GlossariesExtraWarning{Can't obtain dialect label
1120      (tracklang v1.3.6+ required)}%
1121      }%
1122      {\let\@glxtr@dialect\TrackedDialectClosestSubMatch}%
1123      }%
1124      }%
1125      \ifdefempty\@glxtr@dialect
1126      {%

```

No tracked dialect found for the root language.

```

1127      }%
1128      {%

```

Check if there's a caption hook for the given dialect label.

```

1129      \ifcsundef{captions\@glxtr@dialect}{}%
1130      {%

```

Dialect label not recognised. Check if there's a known mapping.

```

1131      \IfTrackedDialectHasMapping{\@glxtr@dialect}%
1132      {%
1133      \edef\@glxtr@dialect{%
1134      \GetTrackedDialectToMapping{\@glxtr@dialect}}%

```

Does a caption hook exist for this?

```

1135      \ifcsundef{captions\@glxtr@dialect}{}%
1136      {%

```

No mapping. Try root language label instead.

```

1137      \ifcsundef{captions\@tracklang@lang}{}%
1138      {%
1139      \let\@glxtr@dialect\@tracklang@lang
1140      }%
1141      }%
1142      }%
1143      {%

```

No mapping. Try root language label instead.

```

1144      \ifcsundef{captions\@tracklang@lang}{}%
1145      {%
1146      \let\@glxtr@dialect\@tracklang@lang
1147      }%
1148      }%
1149      }%

```

```

1150     }%
1151     \ifdefempty\@glstr@dialect
1152     {%
1153         \GlsXtrUnknownDialectWarning{\@glstr@locale}{\@tracklang@lang}%
1154         #2%
1155     }%
1156     {\foreignlanguage{\@glstr@dialect}{#2}}%
1157 }%
1158 {#2}% key not set
1159 }
1160 }
1161 {
1162     \newcommand{\GlsXtrForeignText}[2]{%
1163         \GlossariesExtraWarning{Can't encapsulate foreign text:
1164             tracklang v1.3.6+ required}%
1165         #2%
1166     }
1167 }
1168 }
1169 {
    \foreignlanguage isn't defined so just do <text>.
1170     \newcommand{\GlsXtrForeignText}[2]{#2}
1171 }

```

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

```

1172 \newcommand*{\GlsXtrForeignTextField}{userii}

```

`\nDialectWarning`

```

1173 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
1174     \GlossariesExtraWarning{Can't determine valid dialect label
1175         for locale '#1' (root language: #2)}%
1176 }

```

`\glstrpageref` Like `\glssrefentry` 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.

```

1177 \ifdef\GlsEntryCounterLabelPrefix
1178 {%
1179     \newcommand*{\glstrpageref}[1]{%
1180         \ifglssentrycounter
1181             \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1182         \else
1183             \ifglssubentrycounter
1184                 \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1185             \else
1186                 \gls{#1}%
1187             \fi
1188         \fi

```



```

1189 }
1190 }%
1191 {%
1192   \newcommand*{\glxtrpageref}[1]{%
1193     \ifglentrycounter
1194       \pageref{glentry-\glstoklabel{#1}}%
1195     \else
1196       \ifglssubentrycounter
1197         \pageref{glentry-\glstoklabel{#1}}%
1198       \else
1199         \gls{#1}%
1200       \fi
1201     \fi
1202   }
1203 }%

```

lossary preamble

```

1204 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1205   \ifcsdef{glolist@#1}%
1206   {%
1207     \ifcsundef{@glossarypreamble@#1}%
1208     {\csdef{@glossarypreamble@#1}{}}%
1209   }%
1210   \csappto{@glossarypreamble@#1}{#2}%
1211 }%
1212 {%
1213   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1214 }%
1215 }

```

lossary preamble

```

1216 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1217   \ifcsdef{glolist@#1}%
1218   {%
1219     \ifcsundef{@glossarypreamble@#1}%
1220     {\csdef{@glossarypreamble@#1}{}}%
1221   }%
1222   \cspreto{@glossarypreamble@#1}{#2}%
1223 }%
1224 {%
1225   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1226 }%
1227 }

```

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.

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

In the event that undefined entries should trigger a warning rather than an error, `\ifglused` 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.

```
1228 \renewcommand*{\ifglused}[3]{%
1229   \glstoifexists{#1}{\ifbool{glo@\glstetoklabel{#1}@flag}{#2}{#3}}%
1230 }
```

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`

```
1231 \renewcommand*{\longnewglossaryentry}{%
1232   \@ifstar{\glstr@s@longnewglossaryentry\@glstr@longnewglossaryentry
1233 }
```

`ewglossaryentry` Starred version.

```
1234 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1235   \glstoifnoexists{#1}%
1236   {%
1237     \bgroup
1238     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1239     \long\def\@newglossaryentryprehook{%
1240       \long\def\@glo@desc{#3}%
1241       \@org@newglossaryentryprehook
1242     }%
1243     \renewcommand*{\gls@assign@desc}[1]{%
1244       \global\cslet{glo@\glstetoklabel{#1}@desc}{\@glo@desc}%
1245       \global\cslet{glo@\glstetoklabel{#1}@descplural}{\@glo@descplural}%
1246     }
1247     \gls@defglossaryentry{#1}{#2}%
1248   \egroup
1249   }%
1250 }
```

`ewglossaryentry` Unstarred version.

```
1251 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1252   \glstoifnoexists{#1}%
1253   {%
1254     \bgroup
1255     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1256     \long\def\@newglossaryentryprehook{%
1257       \long\def\@glo@desc{#3\glstrpostlongdescription}%

```

```

1258      \@org@newglossaryentryprehook
1259    }%
1260    \renewcommand*{\gls@assign@desc}[1]{%
1261      \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1262      \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1263    }
1264    \gls@defglossaryentry{#1}{#2}%
1265  \egroup
1266 }%
1267 }

```

The following is different from the base glossaries.sty:

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

```

1268 \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.

```

1269 \renewcommand{\newignoredglossary}{%
1270   \@ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
1271 }

```

ignoredglossary The original definition is patched to check for existence.

```

1272 \newcommand*{\glsxtr@org@newignoredglossary}[1]{%
1273   \ifcsdef{glolist@#1}
1274   {%
1275     \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
1276   }%
1277   {%
1278     \ifdefempty\@ignored@glossaries
1279     {%
1280       \edef\@ignored@glossaries{#1}%
1281     }%
1282     {%
1283       \eappto\@ignored@glossaries{,#1}%
1284     }%
1285     \csgdef{glolist@#1}{,}%
1286     \ifcsundef{gls@#1@entryfmt}%
1287     {%
1288       \defglsentryfmt[#1]{\glsentryfmt}%
1289     }%
1290     {}%
1291     \ifdefempty\@gls@nohyperlist
1292     {%
1293       \renewcommand*{\@gls@nohyperlist}{#1}%
1294     }%
1295     {%

```

```

1296      \eappto\@gls@nohyperlist{,#1}%
1297  }%
1298  }%
1299 }

```

ignoredglossary Starred form.

```

1300 \newcommand*{\glstr@s@newignoredglossary}[1]{%
1301   \ifcsdef{glolist@#1}
1302   {%
1303     \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1304   }%
1305   {%
1306     \ifdefempty\@ignored@glossaries
1307     {%
1308       \edef\@ignored@glossaries{#1}%
1309     }%
1310     {%
1311       \eappto\@ignored@glossaries{,#1}%
1312     }%
1313     \csgdef{glolist@#1}{,}%
1314     \ifcsundef{gls@#1@entryfmt}%
1315     {%
1316       \defglsentryfmt[#1]{\glsentryfmt}%
1317     }%
1318     {}%
1319   }%
1320 }

```

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

```

1321 \glsifusetranslator
1322 {%
1323   \renewcommand*{\glstoctitle}[1]{%
1324     \ifcsdef{gls@tr@set@#1@toctitle}%
1325     {%
1326       \csuse{gls@tr@set@#1@toctitle}%
1327     }%
1328     {%
1329       \ifcsdef{@glotype@#1@title}%
1330       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1331       {\def\glossarytoctitle{\glossarytitle}}%
1332     }%
1333   }%
1334 }
1335 {
1336   \renewcommand*{\glstoctitle}[1]{%
1337     \ifcsdef{@glotype@#1@title}%
1338     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1339     {\def\glossarytoctitle{\glossarytitle}}%

```

```

1340 }
1341 }

```

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

```

1342 \newcommand{\provideignoredglossary}{%
1343 \ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1344 }

```

ignoredglossary Unstarred version.

```

1345 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1346 \ifcsdef{glolist@#1}
1347 {}%
1348 {%
1349 \ifdefempty\@ignored@glossaries
1350 {%
1351 \edef\@ignored@glossaries{#1}%
1352 }%
1353 {%
1354 \eappto\@ignored@glossaries{,#1}%
1355 }%
1356 \csgdef{glolist@#1}{,}%
1357 \ifcsundef{gls@#1@entryfmt}%
1358 {%
1359 \defglsentryfmt[#1]{\glsentryfmt}%
1360 }%
1361 {}%
1362 \ifdefempty\@gls@nohyperlist
1363 {%
1364 \renewcommand*{\@gls@nohyperlist}{#1}%
1365 }%
1366 {%
1367 \eappto\@gls@nohyperlist{,#1}%
1368 }%
1369 }%
1370 }

```

ignoredglossary Starred form.

```

1371 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1372 \ifcsdef{glolist@#1}
1373 {}%
1374 {%
1375 \ifdefempty\@ignored@glossaries
1376 {%
1377 \edef\@ignored@glossaries{#1}%
1378 }%
1379 {%
1380 \eappto\@ignored@glossaries{,#1}%
1381 }%
1382 \csgdef{glolist@#1}{,}%

```

```

1383 \ifcsundef{gls@#1@entryfmt}%
1384 {%
1385 \defglsentryfmt[#1]{\glsentryfmt}%
1386 }%
1387 {}%
1388 }%
1389 }

```

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

```

1390 \newcommand*\glstrcopytoglossary[2]{%
1391 \glsdoifexists{#1}%
1392 {%
1393 \ifcsdef{glolist@#2}
1394 {%
1395 \cseappto{glolist@#2}{#1,}%
1396 }%
1397 {%
1398 \glstrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1399 }%
1400 }%
1401 }

```

1.3.1 Existence Checks

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

```

1402 \renewcommand{\glsdoifexists}[2]{%
1403 \ifglsentryexists{#1}{#2}%
1404 {%

```

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

```

1405 \edef\glslabel{\glsdetoklabel{#1}}%
1406 \glstrundefaction{Glossary entry ‘\glslabel’
1407 has not been defined}{You need to define a glossary entry before
1408 you can reference it.}%
1409 }%
1410 }

```

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

```

1411 \renewcommand{\glsdoifnoexists}[2]{%
1412 \ifglsentryexists{#1}{%
1413 \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1414 has already been defined}{}}{#2}%
1415 }

```

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

```

1416 \ifdef\glsdoifexistsordo
1417 {%
1418   \renewcommand{\glsdoifexistsordo}[3]{%
1419     \ifglentryexists{#1}{#2}%
1420     {%
1421       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1422         has not been defined}{You need to define a glossary entry
1423         before you can use it.}%
1424       #3%
1425     }%
1426   }%
1427 }
1428 {%
1429   \glstr@warnonexistsordo\glsdoifexistsordo
1430   \newcommand{\glsdoifexistsordo}[3]{%
1431     \ifglentryexists{#1}{#2}%
1432     {%
1433       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1434         has not been defined}{You need to define a glossary entry
1435         before you can use it.}%
1436       #3%
1437     }%
1438   }%
1439 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

1440 \ifdef\doifglossarynoexistsordo
1441 {%
1442   \renewcommand{\doifglossarynoexistsordo}[3]{%
1443     \ifglossaryexists{#1}%
1444     {%
1445       \glstrundefaction{Glossary type '#1' already exists}{}%
1446       #3%
1447     }%
1448     {#2}%
1449   }%
1450 }
1451 {%
1452   \glstr@warnonexistsordo\doifglossarynoexistsordo
1453   \newcommand{\doifglossarynoexistsordo}[3]{%
1454     \ifglossaryexists{#1}%
1455     {%
1456       \glstrundefaction{Glossary type '#1' already exists}{}%
1457       #3%
1458     }%
1459     {#2}%
1460   }%
1461 }
1462

```

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.

```
1463 \appto\@newglossaryentryposthook{%
1464   \ifdefvoid\@glo@see
1465     {\csxdef{glo@\@glo@label @see}{}}%
1466     {%
1467       \csxdef{glo@\@glo@label @see}{\@glo@see}%
1468       \if@glxtr@autoseeindex
1469         \@glxtr@autoindexcrossrefs
1470       \fi
1471     }%
1472 }
1473 \appto\@gls@keymap{,{see}{see}}
```

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

```
1474 \newcommand*{\glxtrusesee}[1]{%
1475   \glsdoifexists{#1}%
1476   {%
1477     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1478     \ifdefempty\@glo@see
1479       {}%
1480       {%
1481         \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1482       }%
1483     }%
1484 }
```

`\glxtr@usesee`

```
1485 \newcommand*{\glxtr@usesee}[1][\seename]{%
1486   \@glxtr@usesee[#1]%
1487 }
```

`\@glxtr@usesee`

```
1488 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1489   \glxtruseseeformat{#1}{#2}%
1490 }
```

`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.

```
1491 \newcommand*{\glxtruseseeformat}[2]{%
1492   \glsseeformat[#1]{#2}{}%
1493 }
```

`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.

```
1494 \renewcommand*{\glsseeitemformat}[1]{%
1495   \ifglshasshort{#1}{\glsaccesstext{#1}}{\glsaccessname{#1}}}%
1496 }
```

`\glsxtrhiername` `\glsxtrhiername{<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.

```
1497 \newcommand*{\glsxtrhiername}[1]{%
1498   \glsdoifexists{#1}%
1499   {%
1500     \glsxtrifhasfield{parent}{#1}%
1501     {\glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
1502     }%
1503   \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}}%
1504   }%
1505 }
```

`\Glsxtrhiername` `\Glsxtrhiername{<label>}`

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

```
1506 \newcommand*{\Glsxtrhiername}[1]{%
1507   \glsdoifexists{#1}%
1508   {%
1509     \glsxtrifhasfield{parent}{#1}%
1510     {%
1511       \Glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
1512       \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}}%
1513     }%
1514     {\ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}}%
1515   }%
1516 }
```

`\GlsXtrhiername` `\GlsXtrhiername{<label>}`

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

```
1517 \newcommand*{\GlsXtrhiername}[1]{%
1518   \glsdoifexists{#1}%
1519   {%
1520     \glsxtrifhasfield{parent}{#1}%
1521     {%
1522       \GlsXtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
1523       \ifglshasshort{#1}{\GlsXaccessshort{#1}}{\GlsXaccessname{#1}}}%
1524     }%
1525     {\ifglshasshort{#1}{\GlsXaccessshort{#1}}{\GlsXaccessname{#1}}}%
1526   }%
1527 }
```

```

1519  {%
1520    \glxtrifhasfield{parent}{#1}%
1521    {\GLsXtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1522    }%
1523    \ifglshasshort{#1}{\GLsaccesssshort{#1}}{\GLsaccessname{#1}}}%
1524  }%
1525 }

```

`\GLSxtrhiername` `\GLSxtrhiername{<label>}`

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

```

1526 \newcommand*{\GLSxtrhiername}[1]{%
1527   \glsdoidexists{#1}%
1528   {%
1529     \glxtrifhasfield{parent}{#1}%
1530     {%
1531       \GLSxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1532       \ifglshasshort{#1}{\GLsaccesssshort{#1}}{\GLsaccessname{#1}}}%
1533     }%
1534     {\ifglshasshort{#1}{\GLSaccesssshort{#1}}{\GLSaccessname{#1}}}%
1535   }%
1536 }

```

`\GLSXTRhiername` `\GLSXTRhiername{<label>}`

As above but displays all names in all-caps.

```

1537 \newcommand*{\GLSXTRhiername}[1]{%
1538   \glsdoidexists{#1}%
1539   {%
1540     \glxtrifhasfield{parent}{#1}%
1541     {\GLSXTRhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1542     {}
1543     \ifglshasshort{#1}{\GLSaccesssshort{#1}}{\GLSaccessname{#1}}}%
1544   }%
1545 }

```

`sxtrhiernamesep` Separator used in `\glxtrhiername` and variants.

```

1546 \newcommand*{\glxtrhiernamesep}{\,\small$\triangleright$}\,}

```

`lsxtruseseealso` Apply `\glseeformat` to the `seealso` key if not empty. There's no optional tag to worry about here.

```

1547 \newcommand*{\lsxtruseseealso}[1]{%
1548   \glsdoidexists{#1}%

```

```

1549 {%
1550   \letcs{\@glo@see}{glo\glsdetoklabel{#1}@seealso}%
1551   \ifdefempty\@glo@see
1552     {}%
1553     {%
1554       \expandafter\glsxtruseealsoformat\expandafter{\@glo@see}%
1555     }%
1556   }%
1557 }

```

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

```

1558 \newcommand*{\glsxtruseealsoformat}[1]{%
1559   \glsseeformat[\seealso]{#1}%
1560 }

```

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

```

1561 \newrobustcmd{\glsxtrseelist}[1]{%
1562   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1563 }

```

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

```

1564 \providecommand{\seealso}{see also}

```

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

```

1565 \ifdef\@xdycrossrefhook
1566 {
  Add the cross-reference class definition to the hook.
1567   \appto\@xdycrossrefhook{%
1568     \write\glswrite{(define-crossref-class \string"seealso\string"
1569       :unverified )}%
1570     \write\glswrite{(markup-crossref-list
1571       :class \string"seealso\string"^^J\space\space\space
1572       :open \string"\string\glsxtruseealsoformat\glsopenbrace\string"
1573       :close \string"\glsclosebrace\string")}%
1574   }

```

Append to class list.

```

1575   \appto\@xdylocationclassorder{\space\string"seealso\string"}

```

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

```

1576 \newrobustcmd*{\glsxtrindexseealso}[2]{%
1577   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
1578     \@glsxtr@recordsee{#1}{#2}%

```

```

1579 \fi
1580 \glsdoifexists{#1}%
1581 {%
1582   \@@glsxtrwrglossmark
1583   \def\@gls@xref{#2}%
1584   \@onelevel@sanitize\@gls@xref
1585   \@gls@checkmkidxchars\@gls@xref
1586   \gls@glossary{\csname glo@#1@type\endcsname}{%
1587     (indexentry
1588       :tkey (\csname glo@#1@index\endcsname)
1589       :xref (\string"\@gls@xref\string")
1590       :attr \string"seealso\string"
1591     )
1592   }%
1593 }%
1594 }
1595 }
1596 {
  xindy not in use or glossaries version too old to support this.
1597 \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
1598 }

```

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`.

```

1599 \ifdef\gls@set@xr@key
1600 {

```

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

```

1601 \define@key{glossentry}{alias}{%
1602   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1603 }
1604 \define@key{glossentry}{seealso}{%
1605   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1606 }

```

Add to the key mappings.

```

1607 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1608 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1609 \appto\@newglossaryentryposthook{%
1610   \ifdefvoid\@glo@seealso
1611     {\csxdef{glo@\@glo@label @seealso}{}}%

```

```

1612    {%
1613        \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1614        \if@glstr@autoseeindex
1615            \@glstr@autoindexcrossrefs
1616        \fi
1617    }%

```

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

```

1618    \ifdefvoid\@glo@alias
1619        {\csxdef{glo@\@glo@label @alias}{}}%
1620    {%
1621        \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1622    }%
1623 }

```

Provide user-level commands to access the values.

`\glstralias`

```

1624 \newcommand*\glstralias[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1625 \newcommand*\glstrseealsolabels[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1626 \appto\@glo@autoseehook{%
1627     \ifdefvoid\@glo@alias
1628     {%
1629         \ifdefvoid\@glo@seealso
1630         {}%
1631         {%
1632             \edef\@do@glsee{\noexpand\glstrindexseealso
1633                 {\@glo@label}{\@glo@seealso}}%
1634             \@do@glsee
1635         }%
1636     }%
1637     {%

```

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

```

1638     \ifdefvoid\@glo@see
1639     {%
1640         \edef\@do@glsee{\noexpand\glsee{\@glo@label}{\@glo@alias}}%
1641         \@do@glsee
1642     }%
1643     {}%
1644 }%
1645 }%
1646 }
1647 {

```

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

```

\glxtralias
1648 \glsaddstoragekey*{alias}{\glxtralias}

trseealsolabels
1649 \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.
1650 \appto\@newglossaryentryposthook{%
1651 \ifcsvoid{glo@\@glo@label @alias}%
1652 {%
1653 \ifcsvoid{glo@\@glo@label @seealso}%
1654 }%
1655 {%
1656 \edef\@do@glsssee{\noexpand\glxtrindexseealso
1657 {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1658 \@do@glsssee
1659 }%
1660 }%
1661 {%

Add cross-reference if see key hasn't been used.
1662 \ifdefvoid\@glo@see
1663 {%
1664 \edef\@do@glsssee{\noexpand\glsssee
1665 {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1666 \@do@glsssee
1667 }%
1668 }%
1669 }%
1670 }

1671 }

Add all unused cross-references at the end of the document.
1672 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}

addallcrossrefs Iterate through all used entries and if they have a cross-reference, make sure the cross-
reference has been added.
1673 \newcommand*\glxtraddallcrossrefs{%
1674 \forallglossaries{\@glo@type}%
1675 {%
1676 \forglsentries[\@glo@type]{\@glo@label}%
1677 {%
1678 \ifglused{\@glo@label}%
1679 {\expandafter\@glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1680 }%

```

```

1681 }%
1682 }

```

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

```

1683 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
1684   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@see}%
1685   \ifdefvoid\@glo@see
1686   {}%
1687   {%
1688     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1689   }%
1690   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@seealso}%
1691   \ifdefvoid\@glo@see
1692   {}%
1693   {%
1694     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1695   }%
1696 }

```

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

```

1697 \newcommand*{\glxtr@addunused}[1][ ]{%
1698   \@glxtr@addunused
1699 }

```

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

```

1700 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
1701   \@for\@glxtr@label:=#1\do
1702   {%
1703     \ifglused{\@glxtr@label}{}%
1704     {%
1705       \gladd[format=glxtrunusedformat]{\@glxtr@label}%
1706       \glunset{\@glxtr@label}%
1707       \expandafter\glxtr@addunusedxrefs\expandafter{\@glxtr@label}%
1708     }%
1709   }%
1710 }

```

xtrunusedformat

```

1711 \newcommand*{\glxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

ls@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 \@glxtr@docdefval so that it only inputs the .glsdefs file if docdef=true.

```

1712 \ifdef\gls@begindocdefs
1713 {%

```

```

1714 \renewcommand*{\gls@begindocdefs}{%
1715   \ifnum\@glsxtr@docdefval=1\relax
1716     \@gls@enablesavenonumberlist
1717     \edef\@gls@restoreat{%
1718       \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1719     \makeatletter
1720     \InputIfFileExists{\jobname.glsdefs}{\relax}%
1721     \@gls@restoreat
1722     \undef\@gls@restoreat
1723     \gls@defdocnewglossaryentry
1724   \else
1725     \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.

```

1726     \@gls@enablesavenonumberlist
1727     \let\gls@checkseeallowed\relax
1728     \let\newglossaryentry\new@atom@glossaryentry
1729     \global\newwrite\@gls@deffile
1730     \immediate\openout\@gls@deffile=\jobname.glsdefs

```

Write all currently defined entries.

```

1731     \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
1732   \fi
1733 \fi
1734 }
1735 }
1736 {%
1737   \ifnum\@glsxtr@docdefval=3\relax
1738     \PackageError{glossaries-extra}{Package option
1739       'docdef=\@glsxtr@docdefsetting' requires at least version 4.37
1740       of the base glossaries.sty package}{%
1741   \fi
1742 }

```

m@glossaryentry

```

1743 \newrobustcmd{\new@atom@glossaryentry}[2]{%
1744   \gls@defglossaryentry{#1}{#2}%
1745   \@gls@writedef{#1}%
1746 }

```

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.

```

1747 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1748 \renewcommand{\makenoidxglossaries}{%
1749   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1750   {%
1751     \glsxtr@orgmakenoidxglossaries

```


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

```

1752 \renewcommand{\do@seeglossary}[2]{%
1753   \@glsxtrwrglossmark
1754   \edef\@gls@label{\glsdetoklabel{##1}}%
1755   \protected@write\@auxout{}{%
1756     \string\@gls@reference
1757     {\csname glo@\@gls@label @type\endcsname}%
1758     {\@gls@label}%
1759     {%
1760       \string\glsseeformat##2}%
1761     }%
1762   }%
1763 }%

```

Check for docdefs=restricted:

```

1764 \ifglsxtrdocdefrestricted

```

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

```

1765 \renewcommand*\@gls@reference[3]{%
1766   \ifcsundef{\@glsref@##1}{\csgdef{\@glsref@##1}{}}{}%
1767   \ifinlistcs{##2}{\@glsref@##1}%
1768   {}%
1769   {\listcsgadd{\@glsref@##1}{##2}}%
1770   \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1771   {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1772   {}%
1773   \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1774   }%
1775 \else

```

Disable document definitions.

```

1776 \@glsxtrdocdeffalse
1777 \fi
1778 \disable@keys{glossaries-extra}{docdef}%
1779 }%
1780 {%
1781 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1782 not permitted\MessageBreak
1783 with record=\@glsxtr@record@setting\space package option}%
1784 {You may only use \string\makenoidxglossaries\space with the
1785 record=off option}%
1786 }%
1787 }

```

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

```

1788 \renewcommand*\@gls@defdocnewglossaryentry{%
1789 \ifcase\@glsxtr@docdefval
1790 docdef=false:

```

```

1790 \renewcommand*{\newglossaryentry}[2]{%
1791 \PackageError{glossaries-extra}{Glossary entries must
1792 be \MessageBreak defined in the preamble with \MessageBreak
1793 package option 'docdef=false'\MessageBreak(consider using
1794 'docdef=restricted')}{Move your glossary definitions to
1795 the preamble. You can also put them in a \MessageBreak separate file
1796 and load them with \string\loadglsentries.}%
1797 }%
1798 \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.

```

1799 \let\gls@checkseeallowed\relax
1800 \let\newglossaryentry\newglossaryentry
1801 \else

```

Restricted mode just needs to allow the see value.

```

1802 \let\gls@checkseeallowed\relax
1803 \fi
1804 }%

```

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`

```

1805 \newcommand*{\GlsXtrEnableOnTheFly}{%
1806 \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1807 }

```

`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.)

```

1808 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1809 \renewcommand*{\glsdetoklabel}[1]{%
1810 \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1811 {%
1812 \expandafter\detokenize\expandafter{##1}%
1813 }%
1814 {\detokenize{##1}}%
1815 }%
1816 \@GlsXtrEnableOnTheFly
1817 }
1818 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end#3#4{%
1819 \expandafter\if\glsbackslash#1%
1820 #3%
1821 \else
1822 #4%

```

```

1823 \fi
1824 }

```

sxtrstarflywarn

```

1825 \newcommand*{\glxtrstarflywarn}{%
1826   \GlossariesExtraWarning{Experimental starred version of
1827   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1828   read the warnings in the glossaries-extra user manual)}}%
1829 }

```

rEnableOnTheFly

```

1830 \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:

\glxtrcat

```

1831 \newcommand*{\glxtrcat}{general}

```

\glxtr

```

1832 \newcommand*{\glxtr}[1] [] {%
1833   \def\glxtr@keylist{##1}%
1834   \@glxtr
1835 }

```

\@glxtr

```

1836 \newcommand*{\@glxtr}[2] [] {%
1837   \ifglentryexists{##2}%
1838   {%
1839     \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
1840   }%
1841   {%
1842     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
1843     description={\nopostdesc},##1}%
1844   }%
1845   \expandafter\gls\expandafter[\glxtr@keylist]{##2}%
1846 }

```

\Glsxtr

```

1847 \newcommand*{\Glsxtr}[1] [] {%
1848   \def\glxtr@keylist{##1}%
1849   \@Glsxtr
1850 }

```

\@Glsxtr

```

1851 \newcommand*{\@Glsxtr}[2] [] {%
1852   \ifglentryexists{##2}%

```

```

1853   {%
1854     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1855   }%
1856   {%
1857     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1858       description={\nopostdesc},##1}%
1859   }%
1860   \expandafter\Gls\expandafter[\glstr@keylist]{##2}%
1861 }

```

\glstrpl

```

1862 \newcommand*{\glstrpl}[1][]{%
1863   \def\glstr@keylist{##1}%
1864   \@glstrpl
1865 }

```

\@glstrpl

```

1866 \newcommand*{\@glstrpl}[2][]{%
1867   \ifglentryexists{##2}%
1868   {%
1869     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1870   }%
1871   {%
1872     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1873       description={\nopostdesc},##1}%
1874   }%
1875   \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1876 }

```

\Glsxtrpl

```

1877 \newcommand*{\Glsxtrpl}[1][]{%
1878   \def\glstr@keylist{##1}%
1879   \@Glsxtrpl
1880 }

```

\@Glsxtrpl

```

1881 \newcommand*{\@Glsxtrpl}[2][]{%
1882   \ifglentryexists{##2}
1883   {%
1884     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1885   }%
1886   {%
1887     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1888       description={\nopostdesc},##1}%
1889   }%
1890   \expandafter\Glspl\expandafter[\glstr@keylist]{##2}%
1891 }

```

\GlsXtrWarning

```

1892 \newcommand*{\GlsXtrWarning}[2]{%
1893   \def\@glxtr@optlist{##1}%
1894   \@onelevel@sanitize\@glxtr@optlist
1895   \GlossariesExtraWarning{The options ‘\@glxtr@optlist’ have
1896     been ignored for entry ‘##2’ as it has already been defined}%
1897 }

```

Disable commands after the glossary:

```

1898 \renewcommand\@printglossary[2]{%
1899   \def\@glxtr@printglossopts{##1}%
1900   \@glxtr@orgprintglossary{##1}{##2}%
1901   \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
1902   \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1903   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1904   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1905 }

```

abledflycommand

```

1906 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1907   \PackageError{glossaries-extra}%
1908   {\string##1\space can’t be used after any of the \MessageBreak
1909     glossaries have been displayed}%
1910   {The on-the-fly commands enabled by
1911     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1912     before the glossaries. If you want to use any entries \MessageBreak
1913     after any of the glossaries, you must use the standard \MessageBreak
1914     method of first defining the entry and then using the \MessageBreak
1915     entry with commands like \string\glx}%
1916   \@glxtr@disabledflycommand
1917 }%
1918 \newcommand*{\@glxtr@disabledflycommand}[2][{}]{##2}

```

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

```

1919 \let\GlsXtrEnableOnTheFly\relax
1920 }
1921 \@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.

```

1922 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

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

etglossarystyle

```

1923 \renewcommand*{\setglossarystyle}[1]{%

```

```

1924 \ifcsundef{@glsstyle@#1}%
1925 {%
1926   \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1927 }%
1928 {%
1929   \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

1930   \protected@edef\@glsxtr@current@style{#1}%
1931 }%
1932 \ifx\@glossary@default@style\relax
1933   \protected@edef\@glossary@default@style{#1}%
1934 \fi
1935 }

```

In case we have an old version of glossaries:

```

1936 \ifdef\@glossary@default@style
1937 {}
1938 {%
1939   \let\@glossary@default@style\relax
1940 }

```

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

```

1941 \ifdef\glslistdottedwidth
1942 {%
1943   \ifdim\glslistdottedwidth=.5\hsize
1944     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1945     \AtBeginDocument{%
1946       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1947         \setlength{\glslistdottedwidth}{.5\columnwidth}%
1948       \fi
1949     }%
1950 \fi
1951 }
1952 {}%

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

1953 \ifdef\glsdescwidth
1954 {%
1955   \ifdim\glsdescwidth=.6\hsize
1956     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1957     \AtBeginDocument{%
1958       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1959         \setlength{\glsdescwidth}{.6\columnwidth}%
1960       \fi
1961     }%
1962 \fi

```

```

1963 }
1964 {}%

```

and for \glspagelistwidth:

lspagelistwidth

```

1965 \ifdef\glspagelistwidth
1966 {%
1967   \ifdim\glspagelistwidth=.1\hsize
1968     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1969     \AtBeginDocument{%
1970       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1971         \setlength{\glspagelistwidth}{.1\columnwidth}%
1972       \fi
1973     }%
1974   \fi
1975 }
1976 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```

1977 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1978 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1979   \glsnonumberlistfalse
1980   \renewcommand*{\glossaryentrynumbers}[1]{%
1981     \ifglentryexists{\glscurrententrylabel}%
1982     {%
1983       \@glstrpreloctag
1984       \GlsXtrFormatLocationList{#1}%
1985       \@glstrpostloctag
1986       \gls@save@numberlist{#1}%
1987     }{}%
1988   }%
1989 \else
1990   \glsnonumberlisttrue
1991   \renewcommand*{\glossaryentrynumbers}[1]{%
1992     \ifglentryexists{\glscurrententrylabel}%
1993     {%
1994       \gls@save@numberlist{#1}%
1995     }{}%
1996   }%
1997 \fi

```

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

```

1998 \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`

```

1999 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
2000   \let\@glstrpreloctag\@glstrpreloctag
2001   \let\@glstrpostloctag\@glstrpostloctag
2002   \renewcommand*{\@glstr@pagetag}{#1}%
2003   \renewcommand*{\@glstr@pagetag}{#2}%
2004   \renewcommand*{\@glstr@savepreloctag}[2]{%
2005     \csgdef{\@glstr@preloctag@##1}{##2}%
2006   }%
2007   \renewcommand*{\@glstr@doloctag}{%
2008     \ifcsundef{\@glstr@preloctag\@glscurrententrylabel}%
2009     {%
2010       \GlossariesWarning{Missing pre-location tag for ‘\@glscurrententrylabel’.
2011       Rerun required}%
2012     }%
2013     {%
2014       \csuse{\@glstr@preloctag\@glscurrententrylabel}%
2015     }%
2016   }%
2017 }
2018 \@onlypreamble\GlsXtrEnablePreLocationTag

```

`glstrpreloctag`

```

2019 \newcommand*{\@glstrpreloctag}{%
2020   \let\@glstr@org@delimN\delimN
2021   \let\@glstr@org@delimR\delimR
2022   \let\@glstr@org@glsglignore\glsglignore
2023   \gdef is required as the delimiters may occur inside a scope.
2024   \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2025   \renewcommand*{\delimN}{%
2026     \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2027     \@glstr@org@delimN}%
2028   \renewcommand*{\delimR}{%
2029     \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2030     \@glstr@org@delimR}%
2031   \renewcommand*{\glsglignore}[1]{%
2032     \gdef\@glstr@thisloctag{\relax}%
2033     \@glstr@org@glsglignore{##1}}%
2034   \@glstr@doloctag

```

`glstrpreloctag`

```

2035 \newcommand*{\@glstrpreloctag}{}

```


@glsxtr@pagetag

```
2036 \newcommand*{\@glsxtr@pagetag}{}%
```

glsxtr@pagetag

```
2037 \newcommand*{\@glsxtr@pagetag}{}%
```

lsxtrpostloctag

```
2038 \newcommand*{\@glsxtrpostloctag}{%
2039   \let\delimN\@glsxtr@org@delimN
2040   \let\delimR\@glsxtr@org@delimR
2041   \let\glsignore\@glsxtr@org@glsignore
2042   \protected@write\@auxout{}%
2043     {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
2044 }
```

lsxtrpostloctag

```
2045 \newcommand*{\@glsxtrpostloctag}{}
```

lsxtr@preloctag

```
2046 \newcommand*{\@glsxtr@savepreloctag}[2]{%
2047 \protected@write\@auxout{}{%
2048   \string\providecommand\string\@glsxtr@savepreloctag[2]{}}
```

glsxtr@doloctag

```
2049 \newcommand*{\@glsxtr@doloctag}{}
```

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

```
2050 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
2051   \XKV@plfalse
2052   \XKV@sttrue
2053   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
2054   {%
2055     \csname glsnonumberlist\XKV@resa\endcsname
2056     \ifglsnonumberlist
2057       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
2058     \else
2059       \def\glossaryentrynumbers##1{%
2060         \@glsxtrpreloctag
2061         \GlsXtrFormatLocationList{##1}%
2062         \@glsxtrpostloctag
2063         \gls@save@numberlist{##1}}%
2064       \fi
2065   }%
2066 }
```

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.

```

2067 \renewcommand*{\glsentryfmt}{%
2068   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
2069   \glsifregular{\glslabel}%
2070   {\glsxtrregularfont{\glsentryfmt}}%
2071   {%
2072     \ifglshasshort{\glslabel}%
2073     {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
2074     {\glsxtrregularfont{\glsentryfmt}}%
2075   }%
2076 }

```

`\glsxtrregularfont` Font used for regular entries.

```

2077 \newcommand*{\glsxtrregularfont}[1]{#1}

```

`\glsxtrabbreviationfont` Font used for abbreviation entries.

```

2078 \newcommand*{\glsxtrabbreviationfont}[1]{#1}

```

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

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

```

2079 \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).

```

2080   \@glsxtr@record{#2}{#3}{\glslink}%
2081   \glsdoifexists{#3}%
2082   {%

```

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).

```

2083   \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
2084   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2085   \def\glscustomtext{#4}%
2086   \@glsxtr@field@linkdefs
2087   #1%

```

```

2088 \gls@link[#2]{#3}{#4}%
2089 \let\ifKV@glslink@hyper\glstrorg@ifKV@glslink@hyper
2090 }%
2091 \glspostlinkhook
2092 }

```

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

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

```

2093 \let\glstr@org@gls@\@gls@
2094 \def\@gls@#1#2{%
2095 \glstr@record{#1}{#2}{glslink}%
2096 \glstr@org@gls@{#1}{#2}%
2097 }%

```

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

```

2098 \let\glstr@org@glsp@\@glsp@
2099 \def\@glsp@#1#2{%
2100 \glstr@record{#1}{#2}{glslink}%
2101 \glstr@org@glsp@{#1}{#2}%
2102 }%

```

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

```

2103 \let\glstr@org@Gls@\@Gls@
2104 \def\@Gls@#1#2{%
2105 \glstr@record{#1}{#2}{glslink}%
2106 \glstr@org@Gls@{#1}{#2}%
2107 }%

```

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

```

2108 \let\glstr@org@Glspl@\@Glspl@
2109 \def\@Glspl@#1#2{%
2110 \glstr@record{#1}{#2}{glslink}%
2111 \glstr@org@Glspl@{#1}{#2}%
2112 }%

```

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

```

2113 \let\glstr@org@GLS@\@GLS@
2114 \def\@GLS@#1#2{%
2115 \glstr@record{#1}{#2}{glslink}%
2116 \glstr@org@GLS@{#1}{#2}%
2117 }%

```

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

```

2118 \let\glstr@org@GLspl@\@GLspl@
2119 \def\@GLspl@#1#2{%
2120 \glstr@record{#1}{#2}{glslink}%
2121 \glstr@org@GLspl@{#1}{#2}%
2122 }%

```

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

```

2123 \renewcommand*{\@glsdisp}[3] [] {%
2124   \@glsxtr@record{#1}{#2}{glslink}%
2125   \glsdoifexists{#2}{%
2126     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
2127     \let\glsifplural\@secondoftwo
2128     \let\glscapscase\@firstofthree
2129     \def\glscustomtext{#3}%
2130     \def\glsinsert{}%
2131     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
2132     \@gls@link[#1]{#2}{\@glo@text}%
2133     \ifKV@glslink@local
2134       \glslocalunset{#2}%
2135     \else
2136       \glsunset{#2}%
2137     \fi
2138   }%
2139   \glspostlinkhook
2140 }

```

\@gls@@link@ Redefine to include \@glsxtr@record

```

2141 \renewcommand*{\@gls@@link}[3] [] {%
2142   \@glsxtr@record{#1}{#2}{glslink}%
2143   \glsdoifexistsordo{#2}%
2144   {%
2145     \let\do@gls@link@checkfirsthyper\relax

```

Post-link hook commands need initialising.

```

2146   \def\glscustomtext{#3}%
2147   \@glsxtr@field@linkdefs
2148   \@gls@link[#1]{#2}{#3}%
2149   }%
2150   {%
2151     \glstextformat{#3}%
2152   }%
2153   \glspostlinkhook
2154 }

```

sxtrinitwrgloss Set the default if the wrgloss is omitted.

```

2155 \newcommand*{\glsxtrinitwrgloss}{%
2156   \glsifattribute{\glslabel}{wrgloss}{after}%
2157   {%
2158     \glsxtrinitwrglossbeforefalse
2159   }%
2160   {%
2161     \glsxtrinitwrglossbeforetrue
2162   }%
2163 }

```

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

```
2164 \newif\ifglxtrinitwrglossbefore
2165 \glxtrinitwrglossbeforetrue
```

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

```
2166 \define@choicekey{glslink}{wrgloss}%
2167 [\@glxtr@wrglossval\@glxtr@wrglossnr]%
2168 {before,after}%
2169 {%
2170   \ifcase\@glxtr@wrglossnr\relax
2171   \glxtrinitwrglossbeforetrue
2172   \or
2173   \glxtrinitwrglossbeforefalse
2174   \fi
2175 }

2176 \define@key{glslink}{thevalue}{\def\@glxtr@thevalue{#1}}

2177 \define@key{glslink}{theHvalue}{\def\@glxtr@theHvalue{#1}}
```

tr@hyperoutside Define a hyperoutside key to determine whether \hyperlink should be outside \glstextformat.

```
2178 \define@boolkey{glslink}[glxtr@]{hyperoutside}[true]{}
2179 \glxtr@hyperoutsidettrue
```

ocal@textformat Provide a key to locally change the text format.

```
2180 \define@key{glslink}{textformat}{%
2181   \ifcsdef{#1}
2182   {%
2183     \letcs{\@glxtr@local@textformat}{#1}%
2184   }%
2185   {%
2186     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2187   }%
2188 }

2189 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}
```

nithyperoutside Set the default if the hyperoutside is omitted.

```
2190 \newcommand*{\glxtrinithyperoutside}{%
2191   \glusifattribute{\glslabel}{hyperoutside}{false}%
2192   {%
2193     \glxtr@hyperoutsidefalse
2194   }%
2195   {%
2196     \glxtr@hyperoutsidettrue
2197   }%
2198 }
```

`r@inc@linkcount` Does nothing by default.
2199 `\newcommand*{\glstr@inc@linkcount}{}`

`slinkpresetkeys` User hook performed immediately before options are set. Does nothing by default.
2200 `\newcommand*{\glslinkpresetkeys}{}`

`sXtrExpandedFmt` 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.
2201 `\newrobustcmd*{\GlsXtrExpandedFmt}[2]{%`
2202 `\protected@edef\@glstr@tmp{#2}%`
2203 `\expandafter#1\expandafter{\@glstr@tmp}%`
2204 `}`

`tion@counter@or` 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`.
2205 `\newcommand*{\@glstr@use@equation@counter}{%`
2206 `\@glstr@ifnum@mmode{\def\@gls@counter{equation}}{}}%`
2207 `}`

`sxtr@do@autoadd` If `\GlsXtrAutoAddOnFormat` is used, this will automatically use `\glsadd`. It's therefore only used with `\@gls@link` not with `\glsadd` otherwise it could trigger an infinite loop. The argument indicates the key family (`glslink` or `glossadd`).
2208 `\newcommand*{\glstr@do@autoadd}[1]{}`

`rAutoAddOnFormat` `\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.

```

2209 \newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
2210   \renewcommand*{\glstr@do@autoadd}[1]{%
2211     \begingroup
2212       \protected@edef\@glstr@do@autoadd{%
2213         \noexpand\ifstrequal{##1}{glslink}%
2214         {%
2215           \noexpand\DTLifinlist{\@glsnumberformat}{#2}{\noexpand\glsadd[format={\@glsnumberfor
2216           }%
2217           }%
2218         }%
2219       \@glstr@do@autoadd
2220     \endgroup
2221   }%
2222 }
```

`\@gls@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`.

```

2223 \def\@gls@link[#1]#2#3{%
2224   \leavevmode
2225   \edef\glslabel{\glsdetoklabel{#2}}}%
2226   \def\@gls@link@opts{#1}%
2227   \let\@gls@link@label\glslabel
2228   \let\@glsnumberformat\@glsxtr@defaultnumberformat
2229   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2230   \edef\gls@type{\csname glo@\glslabel @type\endcsname}%
2231   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

```

Save current value of `\glolinkprefix`:

```

2232 \let\@glsxtr@org@glolinkprefix\glolinkprefix

```

Initialise `\@glsxtr@local@textformat`

```

2233 \let\@glsxtr@local@textformat\relax

```

Initialise `thevalue` and `theHvalue` (v1.19).

```

2234 \def\@glsxtr@thevalue{%
2235   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%

```

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

```

2236 \glsxtrinitwrgloss

```

Initialise whether `\hyperlink` should be outside `\glstextformat` (new to v1.21).

```

2237 \glsxtrinithyperoutside

```

Note that the default link options may override `\glsxtrinitwrgloss`.

```

2238 \@gls@setdefault@glslink@opts

```

Increment link counter if enabled (new to v1.26).

```

2239 \glsxtr@inc@linkcount

```

Check if the equations option has been set (new to v1.37).

```

2240 \if@glsxtr@equations
2241   \@glsxtr@use@equation@counter
2242 \fi

```

As the original definition.

```

2243 \do@gl:disablehyperinlist
2244 \do@gls@link@checkfirsthyper

```

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

```

2245 \glslinkpresetkeys

```

Set options.

```

2246 \setkeys{glslink}{#1}%

```

Perform auto add if set (new to v1.37)

```

2247 \glsxtr@do@autoadd{glslink}%

```

User hook after options are set:

```
2248 \glslinkpostsetkeys
```

Check thevalue and theHvalue before saving (v1.19).

```
2249 \ifdefempty{\@glsxtr@thevalue}%
2250 {%
2251   \@gls@saveentrycounter
2252 }%
2253 {%
2254   \let\theglsentrycounter\@glsxtr@thevalue
2255   \def\theHglentrycounter{\@glsxtr@theHvalue}%
2256 }%
2257 \@gls@setsort{\glslabel}%
```

Check if the textformat key has been used.

```
2258 \ifx\@glsxtr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
2259   \glsasattribute{\glslabel}{textformat}%
2260   {%
2261     \edef\@glsxtr@attrval{\glsgetattribute{\glslabel}{textformat}}%
2262     \ifcsdef{\@glsxtr@attrval}%
2263     {%
2264       \letcs{\@glsxtr@textformat}{\@glsxtr@attrval}%
2265     }%
2266     {%
2267       \GlossariesExtraWarning{Unknown control sequence name
2268         '\@glsxtr@attrval' supplied in textformat attribute
2269         for entry '\glslabel'. Reverting to default \string\glstextformat}%
2270       \let\@glsxtr@textformat\glstextformat
2271     }%
2272   }%
2273   {%
2274     \let\@glsxtr@textformat\glstextformat
2275   }%
2276 \else
2277   \let\@glsxtr@textformat\@glsxtr@local@textformat
2278 \fi
```

Do write if it should occur before the link text:

```
2279 \ifglsxtrinitwrglossbefore
2280   \do@wrglossary{#2}%
2281 \fi
```

Do the link text:

```
2282 \ifKV@glslink@hyper
2283   \ifglsxtr@hyperoutside
2284     \@glslink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
2285   \else
2286     \@glsxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2287   \fi
```



```

2288 \else
2289   \ifglxtr@hyperoutside
2290     \glsdonohyperlink{\glolinkprefix\glslabel}{\@glxtr@textformat{#3}}%
2291   \else
2292     \@glxtr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
2293   \fi
2294 \fi

```

Do write if it should occur after the link text:

```

2295 \ifglxtrinitwrglossbefore
2296 \else
2297   \@do@wrglossary{#2}%
2298 \fi

```

Restore original value of \glolinkprefix:

```

2299 \let\glolinkprefix\@glxtr@org@glolinkprefix

```

As the original definition:

```

2300 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2301 }

```

```

2302 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}

```

```

2303 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}

```

\saddpresetkeys

```

2304 \newcommand*{\glsaddpresetkeys}{ }

```

\saddpostsetkeys

```

2305 \newcommand*{\glsaddpostsetkeys}{ }

```

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

```

2306 \renewrobustcmd*{\glsadd}[2][ ]{%
2307   \glxtrifinmark
2308   }{%
2309   {%
2310     \@gls@adjustmode
2311     \begingroup
2312     \@glxtr@record{#1}{#2}{glossadd}%
2313     \glsdoifexists{#2}%
2314     {%
2315       \let\@glsnumberformat\@glxtr@defaultnumberformat
2316       \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
2317       \def\@glxtr@thevalue{}%
2318       \def\@glxtr@theHvalue{\@glxtr@thevalue}%

```

Implement any default settings (before options are set)

```

2319   \glsaddpresetkeys
2320   \setkeys{glossadd}{#1}%

```

Implement any default settings (after options are set)

```

2321 \glsaddpostsetkeys
2322 \ifdefempty{\@glstr@thevalue}%
2323 {%
2324 \@gls@saveentrycounter
2325 }%
2326 {%
2327 \let\theglentrycounter\@glstr@thevalue
2328 \def\theHglentrycounter{\@glstr@theHvalue}%
2329 }%
```

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

```

2330 \@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.

```

2331 \KV@glslink@noindexfalse
2332 \@do@wrglossary{#2}%
2333 }%
2334 \endgroup
2335 }%
2336 }
```

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

```

2337 \newrobustcmd{\glsaddeach}[2][]{%
2338 \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2339 }
```

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

```

2340 \newcommand*{\@glstr@field@linkdefs}{%
2341 \let\glstrifwasfirstuse\@secondoftwo
2342 \let\glsifplural\@secondoftwo
2343 \let\glscapscase\@firstofthree
2344 \let\glsinsert\@empty
2345 }
```

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

assignfieldfont

```

2346 \newcommand*{\glstrassignfieldfont}[1]{%
2347 \ifglentryexists{#1}%
2348 {%
2349 \ifglshasshort{#1}%
2350 {%
2351 \glssetabbrvfmt{\glscategory{#1}}%
2352 \glsifregular{#1}%
2353 {\let\@gls@field@font\glstrregularfont}%
```

```

2354     {\let\@gls@field@font\@firstofone}%
2355 }%
2356 {%
2357     \glsifnotregular{#1}%
2358     {\let\@gls@field@font\@firstofone}%
2359     {\let\@gls@field@font\glsxtrregularfont}%
2360 }%
2361 }%
2362 {%
2363     \let\@gls@field@font\@gobble
2364 }%
2365 }

```

`\@gls@text@` The abbreviation format may also need setting.

```

2366 \def\@gls@text@#1#2[#3]{%
2367     \glsxtrassignfieldfont{#2}%
2368     \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2369 }

```

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

```

2370 \def\@GLStext@#1#2[#3]{%
2371     \glsxtrassignfieldfont{#2}%
2372     \@gls@field@link[\let\gls@scapscase\@thirdofthree]{#1}{#2}%
2373     {\@gls@field@font{\GLS@accesstext{#2}\mfirstucMakeUppercase{#3}}}%
2374 }

```

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

```

2375 \def\@Gls@text@#1#2[#3]{%
2376     \glsxtrassignfieldfont{#2}%
2377     \@gls@field@link[\let\gls@scapscase\@secondofthree]{#1}{#2}%
2378     {\@gls@field@font{\Gls@accesstext{#2}#3}}%
2379 }

```

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`

```

2380 \newcommand*\glsxtrchecknohyperfirst[1]{%
2381     \glsifattribute{#1}{nohyperfirst}{true}{\KV@gls@link@hyperfalse}{}%
2382 }

```

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

```

2383 \def\@glsfirst@#1#2[#3]{%
2384     \glsxtrassignfieldfont{#2}%
2385     \@gls@field@link
2386     [\let\glsxtrifwasfirstuse\@firstoftwo
2387     \glsxtrchecknohyperfirst{#2}%

```

```

2388 ]{#1}{#2}%
2389 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2390 }

```

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

```

2391 \def\@Glsfirst@#1#2[#3]{%
2392   \glsxtrassignfieldfont{#2}%
      Ensure that \@Glsfirst honours the nohyperfirst attribute.
2393   \@gls@field@link
2394   [\let\glsxtrifwasfirstuse\@firstoftwo
2395     \let\glscapscase\@secondofthree
2396     \glsxtrchecknohyperfirst{#2}%
2397   ]%
2398   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2399 }

```

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

```

2400 \def\@GLSfirst@#1#2[#3]{%
2401   \glsxtrassignfieldfont{#2}%
      Ensure that \@GLSfirst honours the nohyperfirst attribute.
2402   \@gls@field@link
2403   [\let\glsxtrifwasfirstuse\@firstoftwo
2404     \let\glscapscase\@thirdofthree
2405     \glsxtrchecknohyperfirst{#2}%
2406   ]%
2407   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2408 }

```

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

```

2409 \def\@glsplural@#1#2[#3]{%
2410   \glsxtrassignfieldfont{#2}%
2411   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2412   {\@gls@field@font{\glsaccessplural{#2}#3}}%
2413 }

```

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

```

2414 \def\@Glsplural@#1#2[#3]{%
2415   \glsxtrassignfieldfont{#2}%
2416   \@gls@field@link
2417   [\let\glsifplural\@firstoftwo
2418     \let\glscapscase\@secondofthree
2419   ]%
2420   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2421 }

```

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

```

2422 \def\@GLSplural@#1#2[#3]{%

```

```

2423 \glstrassignfieldfont{#2}%
2424 \@gls@field@link
2425 [\let\glsifplural\@firstoftwo
2426 \let\glscapscase\@thirdofthree
2427 ]%
2428 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2429 }

```

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

```

2430 \def\@glsfirstplural@#1#2[#3]{%
2431 \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
2432 \@gls@field@link
2433 [\let\glstrifwasfirstuse\@firstoftwo
2434 \let\glsifplural\@firstoftwo
2435 \glstrchecknohyperfirst{#2}%
2436 ]%
2437 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2438 }

```

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

```

2439 \def\@Glsfirstplural@#1#2[#3]{%
2440 \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
2441 \@gls@field@link
2442 [\let\glstrifwasfirstuse\@firstoftwo
2443 \let\glsifplural\@firstoftwo
2444 \let\glscapscase\@secondofthree
2445 \glstrchecknohyperfirst{#2}%
2446 ]%
2447 {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2448 }

```

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

```

2449 \def\@GLSfirstplural@#1#2[#3]{%
2450 \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
2451 \@gls@field@link
2452 [\let\glstrifwasfirstuse\@firstoftwo
2453 \let\glsifplural\@firstoftwo
2454 \let\glscapscase\@thirdofthree
2455 \glstrchecknohyperfirst{#2}%
2456 ]%
2457 {#1}{#2}%
2458 {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2459 }

```

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

```
2460 \def\@glsname@#1#2[#3]{%
2461   \glstrassignfieldfont{#2}%
2462   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessname{#2}#3}}%
2463 }
```

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

```
2464 \def\@Glsname@#1#2[#3]{%
2465   \glstrassignfieldfont{#2}%
2466   \@gls@field@link
2467   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2468   {\@gls@field@font{\@Glsaccessname{#2}#3}}%
2469 }
```

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

```
2470 \def\@GLSname@#1#2[#3]{%
2471   \glstrassignfieldfont{#2}%
2472   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2473   {#1}{#2}%
2474   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2475 }
```

\@glsdesc@

```
2476 \def\@glsdesc@#1#2[#3]{%
2477   \glstrassignfieldfont{#2}%
2478   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2479 }
```

\@Glsdesc@ First letter uppercase version.

```
2480 \def\@Glsdesc@#1#2[#3]{%
2481   \glstrassignfieldfont{#2}%
2482   \@gls@field@link
2483   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2484   {\@gls@field@font{\@Glsaccessdesc{#2}#3}}%
2485 }
```

\@GLSdesc@ All uppercase version.

```
2486 \def\@GLSdesc@#1#2[#3]{%
2487   \glstrassignfieldfont{#2}%
2488   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2489   {#1}{#2}{\@gls@field@font{\@GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2490 }
```

@glsdescplural@ No case-changing version.

```
2491 \def\@glsdescplural@#1#2[#3]{%
2492   \glstrassignfieldfont{#2}%
2493   \@gls@field@link
2494   [\let\glscapscase\@secondoftwo
```

```

2495 \let\glsifplural\@firstoftwo
2496 ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
2497 }

```

@Glsdescplural@ First letter uppercase version.

```

2498 \def\@Glsdescplural@#1#2[#3]{%
2499 \glsxtrassignfieldfont{#2}%
2500 \@gls@field@link
2501 [\let\glscapscase\@secondoftwo
2502 \let\glsifplural\@firstoftwo
2503 ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2504 }

```

@GLSdescplural@ All uppercase version.

```

2505 \def\@GLSdesc@#1#2[#3]{%
2506 \glsxtrassignfieldfont{#2}%
2507 \@gls@field@link
2508 [\let\glscapscase\@thirdoftwo
2509 \let\glsifplural\@firstoftwo
2510 ]%
2511 {#1}{#2}%
2512 {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2513 }

```

\@glssymbol@

```

2514 \def\@glssymbol@#1#2[#3]{%
2515 \glsxtrassignfieldfont{#2}%
2516 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2517 }

```

\@Glsymbol@ First letter uppercase version.

```

2518 \def\@Glsymbol@#1#2[#3]{%
2519 \glsxtrassignfieldfont{#2}%
2520 \@gls@field@link
2521 [\let\glscapscase\@secondoftwo]%
2522 {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2523 }

```

\@GLSsymbol@ All uppercase version.

```

2524 \def\@GLSsymbol@#1#2[#3]{%
2525 \glsxtrassignfieldfont{#2}%
2526 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2527 {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2528 }

```

lssymbolplural@ No case-changing version.

```

2529 \def\@glssymbolplural@#1#2[#3]{%
2530 \glsxtrassignfieldfont{#2}%

```

```

2531 \@gls@field@link
2532 [\let\glscapscase\@secondoftwo
2533 \let\glsifplural\@firstoftwo
2534 ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}{#3}}}%
2535 }

```

lssymbolplural@ First letter uppercase version.

```

2536 \def\@GLssymbolplural@#1#2[#3]{%
2537 \glstrassignfieldfont{#2}%
2538 \@gls@field@link
2539 [\let\glscapscase\@secondoftwo
2540 \let\glsifplural\@firstoftwo
2541 ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}{#3}}}%
2542 }

```

LSsymbolplural@ All uppercase version.

```

2543 \def\@GLSsymbol@#1#2[#3]{%
2544 \glstrassignfieldfont{#2}%
2545 \@gls@field@link
2546 [\let\glscapscase\@thirdoftwo
2547 \let\glsifplural\@firstoftwo
2548 ]%
2549 {#1}{#2}%
2550 {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2551 }

```

\@Glsuseri@ First letter uppercase version.

```

2552 \def\@Glsuseri@#1#2[#3]{%
2553 \glstrassignfieldfont{#2}%
2554 \@gls@field@link
2555 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2556 {\@gls@field@font{\Glsentryuseri{#2}{#3}}}%
2557 }

```

\@GLSuseri@ All uppercase version.

```

2558 \def\@GLSuseri@#1#2[#3]{%
2559 \glstrassignfieldfont{#2}%
2560 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2561 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}{#3}}}}%
2562 }

```

\@Glsuserii@ First letter uppercase version.

```

2563 \def\@Glsuserii@#1#2[#3]{%
2564 \glstrassignfieldfont{#2}%
2565 \@gls@field@link
2566 [\let\glscapscase\@secondoftwo]%
2567 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}{#3}}}%
2568 }

```


\@GLSuserii@ All uppercase version.

```
2569 \def\@GLSuserii@#1#2[#3]{%
2570   \glstrassignfieldfont{#2}%
2571   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2572   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2573 }
```

\@Glsuseriii@ First letter uppercase version.

```
2574 \def\@Glsuseriii@#1#2[#3]{%
2575   \glstrassignfieldfont{#2}%
2576   \@gls@field@link
2577   [\let\glscapscase\@secondoftwo]%
2578   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}}%
2579 }
```

\@GLSuseriii@ All uppercase version.

```
2580 \def\@GLSuseriii@#1#2[#3]{%
2581   \glstrassignfieldfont{#2}%
2582   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2583   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2584 }
```

\@Glsuseriv@ First letter uppercase version.

```
2585 \def\@Glsuseriv@#1#2[#3]{%
2586   \glstrassignfieldfont{#2}%
2587   \@gls@field@link
2588   [\let\glscapscase\@secondoftwo]%
2589   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}}%
2590 }
```

\@GLSuseriv@ All uppercase version.

```
2591 \def\@GLSuseriv@#1#2[#3]{%
2592   \glstrassignfieldfont{#2}%
2593   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2594   {#1}{#2}%
2595   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2596 }
```

\@Glsuserv@ First letter uppercase version.

```
2597 \def\@Glsuserv@#1#2[#3]{%
2598   \glstrassignfieldfont{#2}%
2599   \@gls@field@link
2600   [\let\glscapscase\@secondoftwo]%
2601   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}}%
2602 }
```

\@GLSuserv@ All uppercase version.

```
2603 \def\@GLSuserv@#1#2[#3]{%
```

```

2604 \glstrassignfieldfont{#2}%
2605 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2606 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2607 }

```

\@Glsuservi@ First letter uppercase version.

```

2608 \def\@Glsuservi@#1#2[#3]{%
2609 \glstrassignfieldfont{#2}%
2610 \@gls@field@link
2611 [\let\glscapscase\@secondoftwo]%
2612 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
2613 }

```

\@GLSuservi@ All uppercase version.

```

2614 \def\@GLSuservi@#1#2[#3]{%
2615 \glstrassignfieldfont{#2}%
2616 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2617 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2618 }

```

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

\@acrshort No case change.

```

2619 \def\@acrshort#1#2[#3]{%
2620 \glsdoifexists{#2}%
2621 {%
2622 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2623 \let\glxtrifwasfirstuse\@secondoftwo
2624 \let\glsifplural\@secondoftwo
2625 \let\glscapscase\@firstofthree
2626 \let\glsinsert\@empty
2627 \def\glscustomtext{%
2628 \acronymfont{\glsaccesssshort{#2}}#3%
2629 }%
2630 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2631 }%
2632 \glspostlinkhook
2633 }

```

\@Acrshort First letter uppercase.

```

2634 \def\@Acrshort#1#2[#3]{%
2635 \glsdoifexists{#2}%
2636 {%
2637 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2638 \let\glxtrifwasfirstuse\@secondoftwo
2639 \let\glsifplural\@secondoftwo
2640 \let\glscapscase\@secondofthree
2641 \let\glsinsert\@empty

```

```

2642 \def\glscustomtext{%
2643 \acronymfont{\Glsaccesssshort{#2}}#3%
2644 }%
2645 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2646 }%
2647 \glspostlinkhook
2648 }

```

\@ACRshort All uppercase.

```

2649 \def\@ACRshort#1#2[#3]{%
2650 \glsdoifexists{#2}%
2651 {%
2652 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2653 \let\glstrifwasfirstuse\@secondoftwo
2654 \let\glsifplural\@secondoftwo
2655 \let\glscapscase\@thirdofthree
2656 \let\glsinsert\@empty
2657 \def\glscustomtext{%
2658 \mfirstucMakeUppercase{\acronymfont{\Glsaccesssshort{#2}}#3}%
2659 }%
2660 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2661 }%
2662 \glspostlinkhook
2663 }

```

\@acrshortpl No case change.

```

2664 \def\@acrshortpl#1#2[#3]{%
2665 \glsdoifexists{#2}%
2666 {%
2667 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2668 \let\glstrifwasfirstuse\@secondoftwo
2669 \let\glsifplural\@firstoftwo
2670 \let\glscapscase\@firstofthree
2671 \let\glsinsert\@empty
2672 \def\glscustomtext{%
2673 \acronymfont{\Glsaccesssshortpl{#2}}#3%
2674 }%
2675 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2676 }%
2677 \glspostlinkhook
2678 }

```

\@Acrshortpl First letter uppercase.

```

2679 \def\@Acrshortpl#1#2[#3]{%
2680 \glsdoifexists{#2}%
2681 {%
2682 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2683 \let\glstrifwasfirstuse\@secondoftwo
2684 \let\glsifplural\@firstoftwo

```

```

2685 \let\glscapscase\@secondofthree
2686 \let\glsinsert\@empty
2687 \def\glscustomtext{%
2688     \acronymfont{\Glsaccessshortpl{#2}}#3%
2689 }%
2690 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2691 }%
2692 \glspostlinkhook
2693 }

```

\@ACRshortpl All uppercase.

```

2694 \def\@ACRshortpl#1#2[#3]{%
2695     \glsdoifexists{#2}%
2696     {%
2697         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2698         \let\glxtrifwasfirstuse\@secondoftwo
2699         \let\glsifplural\@firstoftwo
2700         \let\glscapscase\@thirdofthree
2701         \let\glsinsert\@empty
2702         \def\glscustomtext{%
2703             \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
2704         }%
2705         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2706     }%
2707     \glspostlinkhook
2708 }

```

\@acrlong No case change.

```

2709 \def\@acrlong#1#2[#3]{%
2710     \glsdoifexists{#2}%
2711     {%
2712         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2713         \let\glxtrifwasfirstuse\@secondoftwo
2714         \let\glsifplural\@secondoftwo
2715         \let\glscapscase\@firstofthree
2716         \let\glsinsert\@empty
2717         \def\glscustomtext{%
2718             \acronymfont{\Glsaccesslong{#2}}#3%
2719         }%
2720         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2721     }%
2722     \glspostlinkhook
2723 }

```

\@Acrlong First letter uppercase.

```

2724 \def\@Acrlong#1#2[#3]{%
2725     \glsdoifexists{#2}%
2726     {%
2727         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2728 \let\glxtrifwasfirstuse\@secondoftwo
2729 \let\glsifplural\@secondoftwo
2730 \let\glscapscase\@secondofthree
2731 \let\glsinsert\@empty
2732 \def\glscustomtext{%
2733   \acronymfont{\Glsaccesslong{#2}}#3%
2734 }%
2735 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2736 }%
2737 \glspostlinkhook
2738 }

```

\@ACRlong All uppercase.

```

2739 \def\@ACRlong#1#2[#3]{%
2740   \glsdoifexists{#2}%
2741   {%
2742     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2743     \let\glxtrifwasfirstuse\@secondoftwo
2744     \let\glsifplural\@secondoftwo
2745     \let\glscapscase\@thirdofthree
2746     \let\glsinsert\@empty
2747     \def\glscustomtext{%
2748       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
2749     }%
2750     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2751   }%
2752   \glspostlinkhook
2753 }

```

\@acrlongpl No case change.

```

2754 \def\@acrlongpl#1#2[#3]{%
2755   \glsdoifexists{#2}%
2756   {%
2757     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2758     \let\glxtrifwasfirstuse\@secondoftwo
2759     \let\glsifplural\@firstoftwo
2760     \let\glscapscase\@firstofthree
2761     \let\glsinsert\@empty
2762     \def\glscustomtext{%
2763       \acronymfont{\Glsaccesslongpl{#2}}#3%
2764     }%
2765     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2766   }%
2767   \glspostlinkhook
2768 }

```

\@Acrlongpl First letter uppercase.

```

2769 \def\@Acrlongpl#1#2[#3]{%
2770   \glsdoifexists{#2}%

```

```

2771 {%
2772   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2773   \let\glxtrifwasfirstuse\@secondoftwo
2774   \let\gl@sifplural\@firstoftwo
2775   \let\glscapscase\@secondofthree
2776   \let\gl$insert\@empty
2777   \def\glscustomtext{%
2778     \acronymfont{\Glsaccesslongpl{#2}}#3%
2779   }%
2780   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2781 }%
2782 \glspostlinkhook
2783 }

```

\@ACRlongpl All uppercase.

```

2784 \def\@ACRlongpl#1#2[#3]{%
2785   \glsdofexists{#2}%
2786   {%
2787     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2788     \let\glxtrifwasfirstuse\@secondoftwo
2789     \let\gl@sifplural\@firstoftwo
2790     \let\glscapscase\@thirdofthree
2791     \let\gl$insert\@empty
2792     \def\glscustomtext{%
2793       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
2794     }%
2795     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2796   }%
2797   \glspostlinkhook
2798 }

```

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

\@glsaddkey

```

2799 \renewcommand*{\@glsaddkey}[7]{%
2800   \key@ifundefined{glossentry}{#1}%
2801   {%
2802     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2803     \appto\@gl@keymap{, {#1}{#1}}%
2804     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2805     \appto\@newglossaryentryposthook{%
2806       \letcs{\@glo@tmp}{@glo@#1}%
2807       \gl@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2808     }%
2809     \newcommand*{#3}[1]{\@gl@entry@field{##1}{#1}}%
2810     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2811   \ifcsdef{@gl@user@#1@}%

```

```

2812 {%
2813   \PackageError{glossaries}%
2814   {Can't define '\string#5' as helper command
2815     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2816     exists}%
2817   }%
2818 }%
2819 {%
2820   \expandafter\newcommand\expandafter*\expandafter
2821   {\csname @gls@user@#1\endcsname}[2][\%
2822     \new@ifnextchar[%
2823       {\csuse{@gls@user@#1@}{##1}{##2}}%
2824       {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2825   \csdef{@gls@user@#1@}##1##2[##3]{%
2826     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2827   }%
2828   \newrobustcmd*{#5}{%
2829     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2830   }%

```

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

```

2831   \ifcsdef{@Gls@user@#1@}%
2832   {%
2833     \PackageError{glossaries}%
2834     {Can't define '\string#6' as helper command
2835       '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2836       exists}%
2837     }%
2838   }%
2839   {%
2840     \expandafter\newcommand\expandafter*\expandafter
2841     {\csname @Gls@user@#1\endcsname}[2][\%
2842       \new@ifnextchar[%
2843         {\csuse{@Gls@user@#1@}{##1}{##2}}%
2844         {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2845     \csdef{@Gls@user@#1@}##1##2[##3]{%
2846       \@gls@field@link[\let\glscaps@case\@secondofthree]%
2847       {##1}{##2}{#4{##2}##3}%
2848     }%
2849     \newrobustcmd*{#6}{%
2850       \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2851     }%

```

Finally the all caps version (modified):

```

2852   \ifcsdef{@GLS@user@#1@}%
2853   {%
2854     \PackageError{glossaries}%
2855     {Can't define '\string#7' as helper command
2856       '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2857       exists}%

```

```

2858     {}%
2859 }%
2860 {%
2861   \expandafter\newcommand\expandafter*\expandafter
2862     {\csname @GLS@user@#1\endcsname}[2][{}]{%
2863     \new@ifnextchar[%
2864       {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2865     {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
2866   \csdef{@GLS@user@#1@}##1##2[##3]{%
2867     \@gls@field@link[\let\gls@scaps@case\@thirdofthree]%
2868     {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2869 }%
2870 \newrobustcmd*{#7}{%
2871   \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2872 }%
2873 }%
2874 {%
2875   \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
2876 }%
2877 }

```

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

```

2878 \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.

```

2879 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2880 \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).

```

2881 \ifglsused{\glslabel}%
2882 {\let\glsxtrifwasfirstuse\@secondoftwo}
2883 {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2884 \edef\gls@categorylabel{\gls@category{\glslabel}}%
2885 \ifglsused{\glslabel}%
2886 {%
2887   \glsifcategoryattribute{\gls@categorylabel}{nohypernext}{true}%
2888   {\KV@glslink@hyperfalse}}}%
2889 }%
2890 {%
2891   \glsifcategoryattribute{\gls@categorylabel}{nohyperfirst}{true}%
2892   {\KV@glslink@hyperfalse}}}%

```



```

2893 }%
2894 \glslinkcheckfirsthyperhook
2895 }

```

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.

```

2896 \ifdef\do@gl:disablehyperinlist
2897 {%
2898   \let\@gl:extr@do@gl:disablehyperinlist\do@gl:disablehyperinlist
2899   \renewcommand*{\do@gl:disablehyperinlist}{%
2900     \@gl:extr@do@gl:disablehyperinlist
2901     \glsifattribute{\glslabel}{nohyper}{true}{\KV@gl:link@hyperfalse}}}%
2902   }
2903 }
2904 {}

```

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

```

2905 \define@boolkey{glslink}{noindex}[true]{}
2906 \KV@gl:link@noindexfalse

```

If \@gl:issetdefault@gl:link@opts has been defined (glossaries v4.20) use it to set the default keys in \@gl:link.

lt@gl:link@opts

```

2907 \ifdef\@gl:issetdefault@gl:link@opts
2908 {
2909   \renewcommand*{\@gl:issetdefault@gl:link@opts}{%
2910     \KV@gl:link@noindexfalse
2911     \@gl:extr:setaliasnoindex
2912   }
2913 }
2914 {

```

Not defined so prepend it to \do@gl:disablehyperinlist to achieve the same effect.

```

2915   \newcommand*{\@gl:issetdefault@gl:link@opts}{%
2916     \KV@gl:link@noindexfalse
2917     \@gl:extr:setaliasnoindex
2918   }
2919   \preto\do@gl:disablehyperinlist{\@gl:issetdefault@gl:link@opts}
2920 }

```

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.)

```

2921 \providecommand*\gl:extr:setaliasnoindex{}%
2922 \KV@gl:link@noindextrue
2923 }

```

setaliasnoindex

```

2924 \newcommand*{\@glxtrsetaliasnoindex}{%
2925   \glxtrifhasfield{alias}{\glslabel}%
2926   {%
2927     \let\glxtrindexaliased\@glxtrindexaliased
2928     \glxtrsetaliasnoindex
2929     \let\glxtrindexaliased\@no@glxtrindexaliased
2930   }%
2931   {}}%
2932 }

```

xtrindexaliased

```

2933 \newcommand{\@glxtrindexaliased}{%
2934   \ifKV@glslink@noindex
2935   \else
2936     \begingroup
2937     \let\@glsnnumberformat\@glxtr@defaultnumberformat
2938     \edef\@gls@counter{\csname glo@\glsdetoklabel{\glslabel}@counter\endcsname}%
2939     \glxtr@saveentrycounter
2940     \@do@wrglossary{\glxtralias{\glslabel}}%
2941   \endgroup
2942   \fi
2943 }

```

xtrindexaliased

```

2944 \newcommand{\@no@glxtrindexaliased}{%
2945   \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2946     not permitted outside definition of \string\glxtrsetaliasnoindex}%
2947   {}}%
2948 }

```

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

```

2949 \let\glxtrindexaliased\@no@glxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

2950 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2951   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2952     \setkeys{glslink}{#1}%
2953     \@glxtrsetaliasnoindex
2954   }%
2955 }

```

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

```

2956 \newcommand*{\glxtrifindexing}[2]{%
2957   \ifKV@glslink@noindex #2\else #1\fi
2958 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

2959 \renewcommand*{\glswriteentry}[2]{%

```

```

2960 \glxtrifindexing
2961 {%
2962 \ifglindexonlyfirst
2963 \ifglused{#1}
2964 {\glxtrdoautoindexname{#1}{dualindex}}}%
2965 {#2}%
2966 \else
2967 \glusifattribute{#1}{indexonlyfirst}{true}%
2968 {\ifglused{#1}
2969 {\glxtrdoautoindexname{#1}{dualindex}}}%
2970 {#2}}}%
2971 {#2}%
2972 \fi
2973 }%
2974 {}%
2975 }

```

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

```

2976 \appto@@do@@wrglossary{\@glxtr@do@@wrindex
2977 \glxtrdowrglossaryhook{\@gls@label}}%
2978 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2979 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2980 \glxtrdowrglossaryhook{\@gls@label}}%
2981 }

```

`xtr@do@@wrindex`

```

2982 \newcommand*{\@glxtr@do@@wrindex}{%
2983 \glxtrdoautoindexname{\@gls@label}{dualindex}}%
2984 }

```

`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.)

```

2985 \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.

```

2986 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2987 \let\glslinkvar\@firstofthree
2988 \let\@gls@hyp@opt@cs#1\relax
2989 \@ifstar{\s@gls@hyp@opt}%
2990 {\@ifnextchar+%
2991 {\@firstoftwo{\p@gls@hyp@opt}}}%

```

```

2992   {%
2993     \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2994     {\@firstoftwo{\@alt@gls@hyp@opt}}}%
2995     {#1}%
2996   }%
2997 }%
2998 }

alt@gls@hyp@opt  User version
2999 \newcommand*{\@alt@gls@hyp@opt}[1][]{%
3000   \let\glslinkvar\@firstofthree
3001   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}

lt@hyp@opt@char  Contains the character used as the command modifier.
3002 \newcommand*{\@gls@alt@hyp@opt@char}{}

lt@hyp@opt@keys  Contains the option list used as the command modifier.
3003 \newcommand*{\@gls@alt@hyp@opt@keys}{}

rSetAltModifier
3004 \newcommand*{\GlsXtrSetAltModifier}[2]{%
3005   \let\@gls@hyp@opt\@gls@alt@hyp@opt
3006   \def\@gls@alt@hyp@opt@char{#1}%
3007   \def\@gls@alt@hyp@opt@keys{#2}%
3008   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
3009   }{%
3010   {%
      Let bib2gls know the modifier.
3011     \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@altmodifier}[1]{}}%
3012     \protected@write\@auxout{}{\string\@glsxtr@altmodifier{#1}}%
3013   }%
3014 }

org@dohyperlink
3015 \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.
3016 \ifdef\glsnavhyperlink
3017 {
3018   \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
3019     \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
      Scope:
3020     {%
3021       \let\glsdohyperlink\glsxtr@org@dohyperlink

```

```

3022     \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
3023   }%
3024 }%
3025 }
3026 {}

```

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

```

3027 \renewcommand*{\glsdohyperlink}[2]{%
3028   \glsattribute{\glslabel}{targeturl}%
3029   {%
3030     \glsattribute{\glslabel}{targetname}%
3031     {%
3032       \glsattribute{\glslabel}{targetcategory}%
3033       {%
3034         \hyperref{\glsattribute{\glslabel}{targeturl}}{%
3035           {\glsattribute{\glslabel}{targetcategory}}%
3036           {\glsattribute{\glslabel}{targetname}}%
3037           {\glxtrprotectlinks#2}}%
3038       }%
3039     }%
3040     \hyperref{\glsattribute{\glslabel}{targeturl}}{%
3041       }%
3042       {\glsattribute{\glslabel}{targetname}}%
3043       {\glxtrprotectlinks#2}}%
3044   }%
3045 }%
3046 {%
3047   \href{\glsattribute{\glslabel}{targeturl}}{%
3048     {\glxtrprotectlinks#2}}%
3049 }%
3050 }%
3051 {}

```

Check for alias.

```

3052   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
3053   \ifvoid\gloaliaslabel
3054   {%
3055     \glxtrhyperlink{#1}{\glxtrprotectlinks#2}%
3056   }%
3057   {}

```

Redirect link to the alias target.

```

3058   \glxtrhyperlink

```

```

3059     {\glolinkprefix\glsetoklabel{\gloaliaslabel}}%
3060     {\glxtrprotectlinks#2}}%
3061   }%
3062 }%
3063 }

```

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

```

3064 \ifdef\glsshowtarget
3065 {
3066   \newcommand{\glxtrhyperlink}[2]{%
3067     \@glsshowtarget{#1}%
3068     \hyperlink{#1}{#2}%
3069   }%
3070 }
3071 {
3072   \newcommand{\glxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
3073 }

```

`gl:disablehyper` 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.

```

3074 \renewrobustcmd*{\glshyperlink}[2][\glsetentrytext{\@glo@label}]{%
3075   \glsdoidexists{#2}%
3076   {%
3077     \def\@glo@label{#2}%
3078     {\edef\glslabel{#2}%
3079     \@glslink{\glolinkprefix\glslabel}{#1}}%
3080   }%
3081 }

```

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

```

3082 \renewcommand{\gl:disablehyper}{%
3083   \KV@glslink@hyperfalse
3084   \def\@glslink{\glsdohyperlink}%
3085   \let\@glstarget\@secondoftwo
3086 }

```

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

```

3087 \renewcommand{\gl:enablehyper}{%
3088   \KV@glslink@hypertrue
3089   \def\@glslink{\glsdohyperlink}%
3090   \def\@glstarget{\glsdohypertarget}%
3091 }

```

`gl:sdonohyperlink` 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).

```
3092 \def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}
```

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

```
3093 \ifcsundef{hyperlink}%
3094 {%
3095   \def\@glslink{\glsdonohyperlink}
3096 }%
3097 {%
3098   \def\@glslink{\glsdohyperlink}
3099 }
```

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

```
3100 \newcommand*{\glsxtrprotectlinks}{%
3101   \KV@glslink@hyperfalse
3102   \KV@glslink@noindextrue
3103   \let\@gls@\@glsxtr@p@text@
3104   \let\@Gls@\@Glsxtr@p@text@
3105   \let\@GLS@\@GLSxtr@p@text@
3106   \let\@glspl@\@glsxtr@p@plural@
3107   \let\@Glspl@\@Glsxtr@p@plural@
3108   \let\@GLSpl@\@GLSxtr@p@plural@
3109   \let\@glsxtrshort@\@glsxtr@p@short@
3110   \let\@Glsxtrshort@\@Glsxtr@p@short@
3111   \let\@GLSxtrshort@\@GLSxtr@p@short@
3112   \let\@glsxtrlong@\@glsxtr@p@long@
3113   \let\@Glsxtrlong@\@Glsxtr@p@long@
3114   \let\@GLSxtrlong@\@GLSxtr@p@long@
3115   \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
3116   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
3117   \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
3118   \let\@glsxtrlongpl@\@glsxtr@p@longpl@
3119   \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
3120   \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
3121   \let\@acrshort@\@glsxtr@p@acrshort@
3122   \let\@Acrshort@\@Glsxtr@p@acrshort@
3123   \let\@ACRshort@\@GLSxtr@p@acrshort@
3124   \let\@acrshortpl@\@glsxtr@p@acrshortpl@
3125   \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
3126   \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
3127   \let\@acrlong@\@glsxtr@p@acrlong@
3128   \let\@Acrlong@\@Glsxtr@p@acrlong@
3129   \let\@ACRlong@\@GLSxtr@p@acrlong@
3130   \let\@acrlongpl@\@glsxtr@p@acrlongpl@
3131   \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
3132   \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
3133 }
```

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

```
@glxtr@p@text@
3134 \def\@glxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}}

@Glsxtr@p@text@
3135 \def\@Glsxtr@p@text@#1#2[#3]{\@GLstext@{#1}{#2}[#3]}}

@GLSxtr@p@text@
3136 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}}

lsxtr@p@plural@
3137 \def\@glxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}}

lsxtr@p@plural@
3138 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}}

LSxtr@p@plural@
3139 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}}

glxtr@p@short@
3140 \def\@glxtr@p@short@#1#2[#3]{%
3141  {%
3142   \glssetabbrvfmt{\glscategory{#2}}%
3143   \glsabbrvfont{\glsentryshort{#2}}#3%
3144  }%
3145 }

Glsxtr@p@short@
3146 \def\@Glsxtr@p@short@#1#2[#3]{%
3147  {%
3148   \glssetabbrvfmt{\glscategory{#2}}%
3149   \glsabbrvfont{\Glsentryshort{#2}}#3%
3150  }%
3151 }

GLSxtr@p@short@
3152 \def\@GLSxtr@p@short@#1#2[#3]{%
3153  {%
3154   \glssetabbrvfmt{\glscategory{#2}}%
3155   \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
3156  }%
3157 }

sxtr@p@shortpl@
3158 \def\@glxtr@p@shortpl@#1#2[#3]{%
3159  {%
3160   \glssetabbrvfmt{\glscategory{#2}}%
```



```

3161 \glsabbrvfont{\glsentryshortpl{#2}}#3%
3162 }%
3163 }

sctr@p@shortpl@
3164 \def\@Glsctr@p@shortpl@#1#2[#3]{%
3165 {%
3166 \glssetabbrvfmt{\glsentryshortpl{#2}}#3%
3167 \glsabbrvfont{\glsentryshortpl{#2}}#3%
3168 }%
3169 }

Sctr@p@shortpl@
3170 \def\@GLSctr@p@shortpl@#1#2[#3]{%
3171 {%
3172 \glssetabbrvfmt{\glsentryshortpl{#2}}#3%
3173 \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
3174 }%
3175 }

@glscxtr@p@long@
3176 \def\@glscxtr@p@long@#1#2[#3]{\glsentrylong{#2}#3}

@Glsxtr@p@long@
3177 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

@GLSxtr@p@long@
3178 \def\@GLSxtr@p@long@#1#2[#3]{%
3179 {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}

lsctr@p@longpl@
3180 \def\@glscxtr@p@longpl@#1#2[#3]{\glsentrylongpl{#2}#3}

lsxtr@p@longpl@
3181 \def\@Glsxtr@p@longpl@#1#2[#3]{\glsentrylongpl{#2}#3}

LSxtr@p@longpl@
3182 \def\@GLSxtr@p@longpl@#1#2[#3]{%
3183 {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}

xctr@p@acrshort@
3184 \def\@glscxtr@p@acrshort@#1#2[#3]{\acronymfont{\glsentryshort{#2}}#3}

xtr@p@acrshort@
3185 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

xtr@p@acrshort@
3186 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
3187 {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

```

```

r@p@acrshortpl@
3188 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
3189 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
3190 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
3191   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}}

sxtr@p@acrlong@
3192 \def\@glsxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
3193 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
3194 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
3195   {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}}

tr@p@acrlongpl@
3196 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
3197 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
3198 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
3199   {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}}

Commands to minimise conflict.

\@glsxtrp@opt
3200 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}

\glsxtrsetpopts  Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3201 \newcommand*{\glsxtrsetpopts}[1]{%
3202   \renewcommand*{\@glsxtrp@opt}{#1}%
3203 }

lossxtrsetpopts  Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3204 \newcommand*{\glossxtrsetpopts}{%
3205   \glsxtrsetpopts{noindex}%
3206 }

\@@glsxtrp
3207 \newrobustcmd*{\@@glsxtrp}[2]{%

```

Add scope.

```

3208  {%
3209    \let\glspostlinkhook\relax
3210    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
3211  }%
3212 }

```

\@glsxtrp

```

3213 \newrobustcmd*{\@glsxtrp}[2]{%
3214   \ifcsdef{gls#1}%
3215   {%
3216     \@glsxtrp{gls#1}{#2}%
3217   }%
3218   {%
3219     \ifcsdef{glsxtr#1}%
3220     {%
3221       \@glsxtrp{glsxtr#1}{#2}%
3222     }%
3223     {%
3224       \PackageError{glossaries-extra}{‘#1’ not recognised by
3225         \string\glsxtrp}{}%
3226     }%
3227   }%
3228 }

```

\@Glsxtrp

```

3229 \newrobustcmd*{\@Glsxtrp}[2]{%
3230   \ifcsdef{Gls#1}%
3231   {%
3232     \@glsxtrp{Gls#1}{#2}%
3233   }%
3234   {%
3235     \ifcsdef{Glsxtr#1}%
3236     {%
3237       \@glsxtrp{Glsxtr#1}{#2}%
3238     }%
3239     {%
3240       \PackageError{glossaries-extra}{‘#1’ not recognised by
3241         \string\Glsxtrp}{}%
3242     }%
3243   }%
3244 }

```

\@GLSxtrp

```

3245 \newrobustcmd*{\@GLSxtrp}[2]{%
3246   \ifcsdef{GLS#1}%
3247   {%
3248     \@glsxtrp{GLS#1}{#2}%
3249   }%

```

```

3250 {%
3251   \ifcsdef{GLSxtr#1}%
3252   {%
3253     \@@glsxtrp{GLSxtr#1}{#2}%
3254   }%
3255   {%
3256     \PackageError{glossaries-extra}{‘#1’ not recognised by
3257       \string\GLSxtrp}{}%
3258   }%
3259 }%
3260 }

```

\glsxtr@entry@p

```

3261 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3262   \glsifattribute{#1}{headuc}{true}%
3263   {%
3264     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3265   }%
3266   {%
3267     \@gls@entry@field{#1}{#2}%
3268   }%
3269 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3270 \ifdef\texorpdfstring
3271 {
3272   \newcommand{\glsxtrp}[2]{%
3273     \protect\NoCaseChange
3274     {%
3275       \protect\texorpdfstring
3276       {%
3277         \protect\glsxtrifinmark
3278         {%
3279           \ifcsdef{glsxtrhead#1}%
3280           {%
3281             {\protect\csuse{glsxtrhead#1}{#2}}%
3282           }%
3283           {%
3284             \glsxtr@headentry@p{#2}{#1}%
3285           }%
3286         }%
3287         {%
3288           \@glsxtrp{#1}{#2}%
3289         }%
3290       }%
3291       {%
3292         \protect\@gls@entry@field{#2}{#1}%
3293       }%
3294     }%

```

```

3295 }
3296 }
3297 {
3298   \newcommand{\glxtrp}[2]{%
3299     \protect\NoCaseChange
3300     {%
3301       \protect\glxtrifinmark
3302       {%
3303         \ifcsdef{glxtrhead#1}%
3304         {%
3305           {\protect\csuse{glxtrhead#1}}%
3306         }%
3307         {%
3308           \glxtr@headentry@p{#2}{#1}%
3309         }%
3310       }%
3311       {%
3312         \@glxtrp{#1}{#2}%
3313       }%
3314     }%
3315   }
3316 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3317 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

3318 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

3319 \ifdef\textorpdfstring
3320 {
3321   \newcommand{\Glsxtrp}[2]{%
3322     \protect\NoCaseChange
3323     {%
3324       \protect\textorpdfstring
3325       {%
3326         \protect\glxtrifinmark
3327         {%
3328           \ifcsdef{Glsxtrhead#1}%
3329           {%
3330             {\protect\csuse{Glsxtrhead#1}{#2}}%
3331           }%
3332           {%
3333             \protect\@Gls@entry@field{#2}{#1}%
3334           }%

```

```

3335     }%
3336     {%
3337         \@Glsxtrp{#1}{#2}%
3338     }%
3339 }%
3340 {%
3341     \protect\@gls@entry@field{#2}{#1}%
3342 }%
3343 }%
3344 }
3345 }
3346 {
3347     \newcommand{\Glsxtrp}[2]{%
3348         \protect\NoCaseChange
3349         {%
3350             \protect\glsxtrifinmark
3351             {%
3352                 \ifcsdef{Glsxtrhead#1}%
3353                 {%
3354                     {\protect\csuse{Glsxtrhead#1}}%
3355                 }%
3356                 {%
3357                     \protect\@Gls@entry@field{#2}{#1}%
3358                 }%
3359             }%
3360             {%
3361                 \@Glsxtrp{#1}{#2}%
3362             }%
3363         }%
3364     }
3365 }

```

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

```

3366 \ifdef\texorpdfstring
3367 {
3368     \newcommand{\Glsxtrp}[2]{%
3369         \protect\NoCaseChange
3370         {%
3371             \protect\texorpdfstring
3372             {%
3373                 \protect\glsxtrifinmark
3374                 {%
3375                     \ifcsdef{Glsxtr#1}%
3376                     {%
3377                         {\protect\Glsxtrshort[noindex,hyper=false]{#1}[]}%
3378                     }%
3379                     {%
3380                         \protect\mfirstucMakeUppercase
3381                         {%

```

```

3382         \protect\@gls@entry@field{#2}{#1}%
3383     }%
3384 }%
3385 }%
3386 {%
3387     \@GLSxtrp{#1}{#2}%
3388 }%
3389 }%
3390 {%
3391     \protect\@gls@entry@field{#2}{#1}%
3392 }%
3393 }%
3394 }
3395 }
3396 {
3397     \newcommand{\GLSxtrp}[2]{%
3398         \protect\NoCaseChange
3399         {%
3400             \protect\glsxtrifinmark
3401             {%
3402                 \ifcsdef{GLSxtr#1}%
3403                 {%
3404                     {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3405                 }%
3406                 {%
3407                     \protect\mfirstucMakeUppercase
3408                     {%
3409                         \protect\@gls@entry@field{#2}{#1}%
3410                     }%
3411                 }%
3412             }%
3413             {%
3414                 \@GLSxtrp{#1}{#2}%
3415             }%
3416         }%
3417     }
3418 }

```

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 `\@glsxtr@unset`, which performs the actual unsetting through `\@@glsunset` and then does the hook. This means that the unsetting (and the hook) can be switched off by redefining `\@glsunset` and then switched back on again by changing the definition back to `\@glsxtr@unset`.

`\@glsxtr@unset` Global unset.

```
3419 \newcommand*{\@glsxtr@unset}[1]{%
3420   \@@glsunset{#1}%
3421   \glsxtrpostunset{#1}%
3422 }
```

`\@glsunset` Global unset.

```
3423 \let\@glsunset\@glsxtr@unset
```

`glsxtrpostunset`

```
3424 \newcommand*{\glsxtrpostunset}[1]{}
```

Provide a command to store a list of labels that will need unsetting.

`tUnsetBuffering`

```
3425 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3426   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3427 }
```

`tUnsetBuffering` Unstarred version doesn't check for duplicates.

```
3428 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3429   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3430   \def\@glsxtr@unset@buffer{}%
3431   \let\@glsunset\@glsxtrbuffer@unset
3432 }
```

`tUnsetBuffering` Starred version checks for duplicates.

```
3433 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3434   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3435   \def\@glsxtr@unset@buffer{}%
3436   \let\@glsunset\@glsxtrbuffer@nodup@unset
3437 }
```

`xtrbuffer@unset` This must use a global change since `\gls` may have to be placed inside `\mbox` (for example, with soul commands).

```
3438 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3439   \listxadd\@glsxtr@unset@buffer{#1}%
3440 }
```

`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.)


```

3441 \newcommand*{\@glxtrbuffer@nodup@unset}[1]{%
3442   \expandafter\ifinlist\expandafter{#1}{\@glxtr@unset@buffer}{}}%
3443   {\listxadd\@glxtr@unset@buffer{#1}}%
3444 }

```

pUnsetBuffering

```

3445 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3446   \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3447 }

```

pUnsetBuffering Unstarred form (global unset).

```

3448 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3449   \let\@glsunset\@glxtr@unset
3450   \forlistloop\@glsunset\@glxtr@unset@buffer
3451   \let\@glxtr@unset@buffer\@glxtr@org@unset@buffer
3452 }

```

pUnsetBuffering Starred form (local unset).

```

3453 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3454   \forlistloop\@glsllocalunset\@glxtr@unset@buffer
3455   \let\@glsunset\@glxtr@unset
3456 }

```

setBufferedList Iterate over labels stored in the current buffer. The argument is the handler macro.

```

3457 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3458   \forlistloop#1\@glxtr@unset@buffer
3459 }

```

\@glsllocalunset Local unset.

```

3460 \renewcommand*{\@glsllocalunset}[1]{%
3461   \@@glsllocalunset{#1}%
3462   \glxtrpostlocalunset{#1}%
3463 }%

```

rpostlocalunset

```

3464 \newcommand*{\glxtrpostlocalunset}[1]{}

```

\@glreset Global reset.

```

3465 \renewcommand*{\@glreset}[1]{%
3466   \@@glreset{#1}%
3467   \glxtrpostreset{#1}%
3468 }%

```

glxtrpostreset

```

3469 \newcommand*{\glxtrpostreset}[1]{}

```

`\@glslocalreset` Local reset.

```
3470 \renewcommand*{\@glslocalreset}[1]{%
3471   \@glslocalreset{#1}%
3472   \glstrpostlocalreset{#1}%
3473 }%
```

`rpostlocalreset`

```
3474 \newcommand*{\glstrpostlocalreset}[1]{}
```

`slocalreseteach` Locally reset a list of entries.

```
3475 \newcommand*{\glslocalreseteach}[1]{%
3476   \gls@ifnotmeasuring
3477   {%
3478     \@for\@gls@thislabel:=#1\do{%
3479       \glsdoifexists{\@gls@thislabel}%
3480       {%
3481         \@glslocalreset{\@gls@thislabel}%
3482       }%
3483     }%
3484   }%
3485 }
```

`slocalunseteach` Locally unset a list of entries.

```
3486 \newcommand*{\glslocalunseteach}[1]{%
3487   \gls@ifnotmeasuring
3488   {%
3489     \@for\@gls@thislabel:=#1\do{%
3490       \glsdoifexists{\@gls@thislabel}%
3491       {%
3492         \@glslocalunset{\@gls@thislabel}%
3493       }%
3494     }%
3495   }%
3496 }
```

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

```
3497 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
3498   \glsenableentrycount
```

Redefine `\gls` etc:

```
3499   \renewcommand*{\gls}{\cgl}%
3500   \renewcommand*{\Gls}{\cGls}%
3501   \renewcommand*{\glspl}{\cglspl}%
3502   \renewcommand*{\Glspl}{\cGlspl}%
3503   \renewcommand*{\GLS}{\cGLS}%
3504   \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
3505 \@glstr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
3506 \let\GlsXtrEnableEntryCounting\@glstr@setentrycountunsetattr
3507 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3508 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3509 can't be used with \string\GlsXtrEnableEntryCounting}%
3510 {Use one or other but not both commands}}%
3511 }
```

ycountunsetattr

```
3512 \newcommand*{\@glstr@setentrycountunsetattr}[2]{%
3513 \@for\@glstr@cat:=#1\do
3514 {%
3515 \ifdefempty{\@glstr@cat}{}%
3516 {%
3517 \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
3518 }%
3519 }%
3520 }
```

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

nableentrycount

```
3521 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
3522 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
3523 \renewcommand*{\gls@defdocnewglossaryentry}{%
3524 \renewcommand*{\newglossaryentry}[2]{%
3525 \PackageError{glossaries}{\string\newglossaryentry\space
3526 may only be used in the preamble when entry counting has
3527 been activated}{If you use \string\glsenableentrycount\space
3528 you must place all entry definitions in the preamble not in
3529 the document environment}%
3530 }%
3531 }%
```

New commands to access new fields:

```
3532 \newcommand*{\glscurrentcount}[1]{%
3533 \ifcsundef{glo@\glscdetoklabel{##1}@currentcount}%
3534 {0}{\@gls@entry@field{##1}{currentcount}}%
3535 }%
3536 \newcommand*{\glsprevcount}[1]{%
3537 \ifcsundef{glo@\glscdetoklabel{##1}@prevcount}%
3538 {0}{\@gls@entry@field{##1}{prevcount}}%
3539 }%
```

Adjust post unset and reset:

```

3540 \let\@glxtr@entrycount@org@unset\glxtrpostunset
3541 \renewcommand*{\glxtrpostunset}[1]{%
3542   \@glxtr@entrycount@org@unset{##1}%
3543   \@glx@increment@currcount{##1}%
3544 }%
3545 \let\@glxtr@entrycount@org@localunset\glxtrpostlocalunset
3546 \renewcommand*{\glxtrpostlocalunset}[1]{%
3547   \@glxtr@entrycount@org@localunset{##1}%
3548   \@glx@local@increment@currcount{##1}%
3549 }%
3550 \let\@glxtr@entrycount@org@reset\glxtrpostreset
3551 \renewcommand*{\glxtrpostreset}[1]{%
3552   \@glxtr@entrycount@org@reset{##1}%
3553   \csgdef{glo\glx@detoklabel{##1}@currcount}{0}%
3554 }%
3555 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
3556 \renewcommand*{\glxtrpostlocalreset}[1]{%
3557   \@glxtr@entrycount@org@localreset{##1}%
3558   \csdef{glo\glx@detoklabel{##1}@currcount}{0}%
3559 }%

```

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

```

3560 \let\@cglx@ \@cglx@
3561 \let\@cglxpl@ \@cglxpl@

3562 \let\@cGlx@ \@cGlx@
3563 \let\@cGlxpl@ \@cGlxpl@
3564 \let\@cGLS@ \@cGLS@
3565 \let\@cGLSpl@ \@cGLSpl@

```

The rest is as the original definition.

```

3566 \AtEndDocument{\@glx@write@entrycounts}%
3567 \renewcommand*{\@glx@entry@count}[2]{%
3568   \csgdef{glo\glx@detoklabel{##1}@prevcount}{##2}%
3569 }%
3570 \let\glxenableentrycount\relax
3571 \renewcommand*{\glxenableentryunitcount}{%
3572   \PackageError{glossaries-extra}{\string\glxenableentryunitcount\space
3573     can't be used with \string\glxenableentrycount}%
3574   {Use one or other but not both commands}%
3575 }%
3576 }

```

`\@glx@write@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.

```

3577 \renewcommand*{\@glx@write@entrycounts}{%
3578   \immediate\write\@auxout
3579   {\string\providecommand*{\string\@glx@entry@count}[2]{}}%

```

```

3580 \count@=0\relax
3581 \forallglentries{\@glentry}{%
3582   \glshasattribute{\@glentry}{entrycount}%
3583   {%
3584     \ifglused{\@glentry}%
3585     {%
3586       \immediate\write\@auxout
3587       {\string\@gl@entry@count{\@glentry}{\glentrycurrcount{\@glentry}}}%
3588     }%
3589   }%
3590   \advance\count@ by \@ne
3591 }%
3592 {}%
3593 }%
3594 \ifnum\count@=0
3595   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3596   \MessageBreak with \string\glsenableentrycount\space but the
3597   \MessageBreak attribute 'entrycount' hasn't
3598   \MessageBreak been assigned to any of the defined
3599   \MessageBreak entries}%
3600 \fi
3601 }

```

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

```

3602 \newcommand*\glxtrifcounttrigger[3]{%
3603   \glshasattribute{#1}{entrycount}%
3604   {%
3605     \ifnum\glentryprevcount{#1}>\glsetattribute{#1}{entrycount}\relax
3606     #3%
3607   \else
3608     #2%
3609   \fi
3610 }%
3611 {#3}%
3612 }

```

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

`\@@cgl@`

```

3613 \def\@@cgl@#1#2[#3]{%
3614   \glxtrifcounttrigger{#2}%
3615   {%
3616     \cglformat{#2}{#3}%
3617     \glunset{#2}%
3618   }%

```

```

3619  {%
3620    \@gls@{#1}{#2}[#3]%
3621  }%
3622 }%

```

\@@cglsp1@

```

3623 \def\@@cglsp1@#1#2[#3]{%
3624   \glstrifcounttrigger{#2}%
3625   {%
3626     \cglsp1format{#2}{#3}%
3627     \glunset{#2}%
3628   }%
3629   {%
3630     \@glsp1@{#1}{#2}[#3]%
3631   }%
3632 }%

```

\@@cGls@

```

3633 \def\@@cGls@#1#2[#3]{%
3634   \glstrifcounttrigger{#2}%
3635   {%
3636     \cGlsformat{#2}{#3}%
3637     \glunset{#2}%
3638   }%
3639   {%
3640     \@Gls@{#1}{#2}[#3]%
3641   }%
3642 }%

```

\@@cGlsp1@

```

3643 \def\@@cGlsp1@#1#2[#3]{%
3644   \glstrifcounttrigger{#2}%
3645   {%
3646     \cGlsp1format{#2}{#3}%
3647     \glunset{#2}%
3648   }%
3649   {%
3650     \@Glsp1@{#1}{#2}[#3]%
3651   }%
3652 }%

```

\@@cGLS@

```

3653 \def\@@cGLS@#1#2[#3]{%
3654   \glstrifcounttrigger{#2}%
3655   {%
3656     \cGLSformat{#2}{#3}%
3657     \glunset{#2}%
3658   }%
3659   {%

```

```

3660 \cGLS@{#1}{#2}[#3]%
3661 }%
3662 }%

```

\cGLSpl@

```

3663 \def\cGLSpl@#1#2[#3]{%
3664 \glxtrifcounttrigger{#2}%
3665 {%
3666 \cGLSplformat{#2}{#3}%
3667 \glset{#2}%
3668 }%
3669 {%
3670 \cGLSpl@{#1}{#2}[#3]%
3671 }%
3672 }%

```

Remove default warnings from \cglset etc so that it can be used interchangeable with \glset etc.

\cglset@

```

3673 \def\cglset@#1#2[#3]{\cglset@{#1}{#2}[#3]}

```

\cGLset@

```

3674 \def\cGLset@#1#2[#3]{\cGLset@{#1}{#2}[#3]}

```

\cglspl@

```

3675 \def\cglspl@#1#2[#3]{\cglspl@{#1}{#2}[#3]}

```

\cGLspl@

```

3676 \def\cGLspl@#1#2[#3]{\cGLspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

\cGLS

```

3677 \newrobustcmd*{\cGLS}{\cglsethyp@opt\cGLS}

```

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

```

3678 \newcommand*{\cGLS}[2][{}]{%
3679 \new@ifnextchar[\cGLS@{#1}{#2}]{\cGLS@{#1}{#2}[{}]}%
3680 }

```

\cGLS@

```

3681 \def\cGLS@#1#2[#3]{\cGLS@{#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.

```

3682 \newcommand*{\cGLSformat}[2]{%
3683 \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsetformat{#1}{#2}}%
3684 }

```

```
\cGLSp1
3685 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\@cGLSp1}
```

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

```
3686 \newcommand*{\@cGLSp1}[2][\%
3687 \new@ifnextchar[\@cGLSp1@{#1}{#2}]{\@cGLSp1@{#1}{#2}[1]}%
3688 }
```

\@cGLSp1@

```
3689 \def\@cGLSp1@#1#2[#3]{\@cGLSp1@{#1}{#2}[#3]}
```

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

```
3690 \newcommand*{\cGLSplformat}[2]{%
3691 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3692 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3693 \renewcommand*{\cglformat}[2]{%
3694 \glsifregular{#1}
3695 {\glsentryfirst{#1}}%
3696 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
3697 }
```

\cGlsformat

```
3698 \renewcommand*{\cGlsformat}[2]{%
3699 \glsifregular{#1}
3700 {\Glsentryfirst{#1}}%
3701 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
3702 }
```

\cglsplformat

```
3703 \renewcommand*{\cglsplformat}[2]{%
3704 \glsifregular{#1}
3705 {\glsentryfirstplural{#1}}%
3706 {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
3707 }
```

\cGlsplformat

```
3708 \renewcommand*{\cGlsplformat}[2]{%
3709 \glsifregular{#1}
3710 {\Glsentryfirstplural{#1}}%
3711 {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
3712 }
```

New code similar to above for unit counting.

defunitcounters

```
3713 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3714   \edef\@glo@countunit{\csuse{@glxtr@categoryattr@@\@glo@category @unitcount}}%
3715   \ifdefvoid\@glo@countunit
3716   {}%
3717   {%
3718     \@glxtr@ifunitcounter{\@glo@countunit}%
3719     {}%
3720     {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
3721   }%
3722 }
```

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

```
3723 \newcommand*{\@glxtr@unitcountlist}{}
```

@addunitcounter

```
3724 \newcommand*{\@glxtr@addunitcounter}[1]{%
3725   \listadd{\@glxtr@unitcountlist}{#1}%
3726   \ifcsundef{glxtr@theunit@#1}
3727   {%
3728     \ifcsdef{theH#1}%
3729     {\csdef{glxtr@theunit@#1}{\csuse{theH#1}}}%
3730     {\csdef{glxtr@theunit@#1}{\csuse{the#1}}}%
3731   }%
3732   {}%
3733 }
```

r@ifunitcounter

```
3734 \newcommand*{\@glxtr@ifunitcounter}[3]{%
3735   \xifinlist{#1}{\@glxtr@unitcountlist}{#2}{#3}%
3736 }
```

urrentunitcount

```
3737 \newcommand*\@glxtr@currentunitcount[1]{%
3738   glo@\glsdetoklabel{#1}@currunit@glsggetattribute{#1}{unitcount}.%
3739   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
3740 }
```

eviousunitcount

```
3741 \newcommand*\@glxtr@previousunitcount[1]{%
3742   glo@\glsdetoklabel{#1}@prevunit@glsggetattribute{#1}{unitcount}.%
3743   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
3744 }
```

t@currunitcount

```
3745 \newcommand*{\@gl@increment@currunitcount}[1]{%
3746   \glshasattribute{#1}{unitcount}%
3747   {%
```

```

3748 \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3749 \ifcsundef{\@glxtr@csname}%
3750 {%
3751 \csgdef{\@glxtr@csname}{1}%
3752 \listcsxadd
3753 {glo@glsdetoklabel{#1}@unitlist}%
3754 {\glsggetattribute{#1}{unitcount}.%
3755 \csuse{glxtr@theunit@glsggetattribute{#1}{unitcount}}}%
3756 }%
3757 }%
3758 {%
3759 \csxdef{\@glxtr@csname}%
3760 {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3761 }%
3762 }%
3763 {}%
3764 }

```

t@currunitcount

```

3765 \newcommand*{\@glx@local@increment@currunitcount}[1]{%
3766 \glshasattribute{#1}{unitcount}%
3767 {%
3768 \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3769 \ifcsundef{\@glxtr@csname}%
3770 {%
3771 \csdef{\@glxtr@csname}{1}%
3772 \listcseadd
3773 {glo@glsdetoklabel{#1}@unitlist}%
3774 {\glsggetattribute{#1}{unitcount}.%
3775 \csuse{glxtr@theunit@glsggetattribute{#1}{unitcount}}}%
3776 }%
3777 }%
3778 {%
3779 \csedef{\@glxtr@csname}%
3780 {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3781 }%
3782 }%
3783 {}%
3784 }

```

r@currunitcount

```

3785 \newcommand*{\@glxtr@currunitcount}[2]{%
3786 \ifcsundef
3787 {glo@glsdetoklabel{#1}@currunit@#2}%
3788 {0}%
3789 {\csuse{glo@glsdetoklabel{#1}@currunit@#2}}%
3790 }%

```

r@prevunitcount

```

3791 \newcommand*{\@glxstr@prevunitcount}[2]{%
3792   \ifcsundef
3793     {glo@\glsdetoklabel{##1}@prevunit@#2}%
3794     {0}%
3795     {\csuse{glo@\glsdetoklabel{##1}@prevunit@#2}}%
3796 }%

```

entryunitcount

```

3797 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3798   \appto@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3799   \renewcommand*{\gl@defdocnewglossaryentry}{%
3800     \renewcommand*{\newglossaryentry}[2]{%
3801       \PackageError{glossaries}{\string\newglossaryentry\space
3802         may only be used in the preamble when entry counting has
3803         been activated}{If you use \string\glsenableentryunitcount\space
3804         you must place all entry definitions in the preamble not in
3805         the document environment}%
3806     }%
3807 }%

```

New commands to access new fields:

```

3808 \newcommand*{\glentrycurrcount}[1]{%
3809   \@glxstr@currunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3810   \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3811 }%
3812 \newcommand*{\glentryprevcount}[1]{%
3813   \@glxstr@prevunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3814   \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3815 }%

```

Access total count:

```

3816 \newcommand*{\glentryprevtotalcount}[1]{%
3817   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3818     {0}%
3819     {%
3820       \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3821     }%
3822 }%

```

Access max value:

```

3823 \newcommand*{\glentryprevmaxcount}[1]{%
3824   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3825     {0}%
3826     {%
3827       \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3828     }%
3829 }%

```

Adjust post unset and reset:

```

3830 \let\@glxstr@entryunitcount@org@unset\glxstrpostunset
3831 \renewcommand*{\glxstrpostunset}[1]{%
3832   \@glxstr@entryunitcount@org@unset{##1}%
3833   \@glx@increment@currunitcount{##1}%
3834 }%
3835 \let\@glxstr@entryunitcount@org@localunset\glxstrpostlocalunset
3836 \renewcommand*{\glxstrpostlocalunset}[1]{%
3837   \@glxstr@entryunitcount@org@localunset{##1}%
3838   \@glx@local@increment@currunitcount{##1}%
3839 }%
3840 \let\@glxstr@entryunitcount@org@reset\glxstrpostreset
3841 \renewcommand*{\glxstrpostreset}[1]{%
3842   \glshasattribute{##1}{unitcount}%
3843   {%
3844     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3845     \ifcsundef{\@glxstr@csname}%
3846     {%
3847       {\csgdef{\@glxstr@csname}{0}}%
3848     }%
3849   }%
3850 }%
3851 \let\@glxstr@entryunitcount@org@localreset\glxstrpostlocalreset
3852 \renewcommand*{\glxstrpostlocalreset}[1]{%
3853   \@glxstr@entryunitcount@org@localreset{##1}%
3854   \glshasattribute{##1}{unitcount}%
3855   {%
3856     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3857     \ifcsundef{\@glxstr@csname}%
3858     {%
3859       {\csdef{\@glxstr@csname}{0}}%
3860     }%
3861   }%
3862 }%

```

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

```

3863 \let\@cgl@\@cgl@
3864 \let\@cgl@spl@\@cgl@spl@

3865 \let\@cGls@\@cGls@
3866 \let\@cGlspl@\@cGlspl@
3867 \let\@cGLS@\@cGLS@
3868 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3869 \AtEndDocument{\@glx@write@entryunitcounts}%
3870 \renewcommand*{\@glx@entry@unitcount}[3]{%
3871   \csgdef{glo@glx@detoklabel{##1}@prevunit@##3}{##2}%
3872   \ifcsundef{glo@glx@detoklabel{##1}@prevunittotal}%

```

```

3873 {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
3874 {%
3875   \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
3876     \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
3877   }%
3878   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3879   {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
3880   {%
3881     \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3882     \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3883     \fi
3884   }%
3885 }%
3886 \let\glsenableentryunitcount\relax
3887 \renewcommand*{\glsenableentrycount}{%
3888   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3889     can't be used with \string\glsenableentryunitcount}%
3890   {Use one or other but not both commands}%
3891 }%
3892 }
3893 \onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3894 \newcommand*{\@gls@entry@unitcount}[3]{%

```

ryunitcounts@do

```

3895 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3896   \immediate\write\@auxout
3897   {\string\@gls@entry@unitcount
3898     {\@glsentry}%
3899     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3900     }%
3901     {#1}}%
3902 }

```

entryunitcounts

```

3903 \newcommand*{\@gls@write@entryunitcounts}{%
3904   \immediate\write\@auxout
3905   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3906   \count@=0\relax
3907   \forallglsentries{\@glsentry}{%
3908     \glsattribute{\@glsentry}{unitcount}%
3909     {%
3910       \ifglsused{\@glsentry}%
3911       {%
3912         \forlistcsloop
3913         {\@gls@write@entryunitcounts@do}%
3914         {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3915       }%

```

```

3916      {}%
3917      \advance\count@ by \@ne
3918    }%
3919    {}%
3920  }%
3921  \ifnum\count@=0
3922    \GlossariesExtraWarningNoLine{Entry counting has been enabled
3923      \MessageBreak with \string\glsenableentryunitcount\space but the
3924      \MessageBreak attribute ‘unitcount’ hasn’t
3925      \MessageBreak been assigned to any of the defined
3926      \MessageBreak entries}%
3927  \fi
3928 }

```

`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.

```

3929 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3930 \glsenableentryunitcount

```

Redefine `\gls` etc:

```

3931 \renewcommand*{\gls}{\cgl}%
3932 \renewcommand*{\Gls}{\cGls}%
3933 \renewcommand*{\glspl}{\cglspl}%
3934 \renewcommand*{\Glspl}{\cGlspl}%
3935 \renewcommand*{\GLS}{\cGLS}%
3936 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3937 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3938 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
3939 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3940 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3941 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3942 {Use one or other but not both commands}}%
3943 }

```

`tcountunsetattr`

```

3944 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3945 \@for\@glsxtr@cat:=#1\do
3946 {%
3947 \ifdefempty{\@glsxtr@cat}{}%
3948 {%
3949 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3950 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3951 }%
3952 }%
3953 }

```

1.3.6 Acronym Modifications

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

`nericNewAcronym`

```

3954 \renewcommand*{\SetGenericNewAcronym}{%
3955   \let\@Gls@entryname\@Gls@acrenryname
3956   \renewcommand{\newacronym}[4][]{%
3957     \ifdefempty{\@glsacronymlists}%
3958     {%
3959       \def\@glo@type{\acronymtype}%
3960       \setkeys{glossentry}{##1}%
3961       \DeclareAcronymList{\@glo@type}%
3962     }%
3963   }%
3964   \glskeylisttok{##1}%
3965   \glslabeltok{##2}%
3966   \glsshorttok{##3}%
3967   \glslongtok{##4}%
3968   \newacronymhook
3969   \protected@edef\@do@newglossaryentry{%
3970     \noexpand\newglossaryentry{\the\glslabeltok}%
3971     {%
3972       type=\acronymtype,%
3973       name={\expandonce{\acronymentry{##2}}},%
3974       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3975       text={\the\glsshorttok},%
3976       short={\the\glsshorttok},%
3977       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3978       long={\the\glslongtok},%
3979       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3980       category=acronym,%
3981       \GenericAcronymFields,%
3982       \the\glskeylisttok
3983     }%
3984   }%
3985   \@do@newglossaryentry
3986 }%
3987 \renewcommand*{\acrfullfmt}[3]{%
3988   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}%
3989 \renewcommand*{\Acrfullfmt}[3]{%
3990   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}%
3991 \renewcommand*{\ACRfullfmt}[3]{%
3992   \glslink[##1]{##2}{%

```

```

3993     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3994 \renewcommand*{\acrfullplfmt}[3]{%
3995   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3996 \renewcommand*{\Acrfullplfmt}[3]{%
3997   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3998 \renewcommand*{\ACRfullplfmt}[3]{%
3999   \glslink[##1]{##2}{%
4000     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
4001 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
4002 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
4003 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
4004 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
4005 }

```

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

First save the original definitions:

```

4006 \let\@glsxtr@org@setacronymstyle\setacronymstyle
4007 \let\@glsxtr@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`.

```

4008 \newcommand*{\MakeAcronymsAbbreviations}{%
4009   \renewcommand*{\newacronym}[4][]{%
4010     \glsxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4011   }%
4012   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4013   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
4014   \renewcommand*{\setacronymstyle}[1]{%
4015     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
4016       unavailable.
4017       Use \string\setabbreviationstyle\space instead.
4018       The original acronym interface can be restored with
4019       \string\RestoreAcronyms}{}%
4020   }%
4021   \renewcommand*{\newacronymstyle}[1]{%
4022     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4023       available unless you restore the original acronym interface with
4024       \string\RestoreAcronyms}%
4025     \@glsxtr@org@newacronymstyle{##1}%
4026   }%
4027 }

```

Switch acronyms to abbreviations:

```

4028 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.


```

4029 \newcommand*\RestoreAcronyms}{%
4030   \SetGenericNewAcronym
4031   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
4032   \renewcommand{\acronymfont}[1]{##1}%
4033   \let\setacronymstyle\@glxtr@org@setacronymstyle
4034   \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.

```

4035   \renewcommand*\@gls@link@checkfirsthyper{%
4036     \ifglused{\glslabel}%
4037     {\let\glxtrifwasfirstuse\@secondoftwo}
4038     {\let\glxtrifwasfirstuse\@firstoftwo}%
4039     \@glxtr@org@checkfirsthyper
4040   }
4041   \glssetcategoryattribute{acronym}{regular}{false}%
4042   \setacronymstyle{long-short}%
4043 }

```

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

```

4044 \renewcommand*\glsacspace}[1]{%
4045   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4046   \ifdim\dimen@<\glsacspacemax~\else\space\fi
4047 }

```

`\glsacspacemax` Value used in the above.

```

4048 \newcommand*\glsacspacemax{3em}

```

1.3.7 Indexing and Displaying Glossaries

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

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

`\reg@glosslist`

```

4049 \newcommand*\@glxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

4050 \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`

```

4051 \renewcommand*\makeglossaries}[1][]{%
4052   \@glxtr@if@record@only

```

```

4053 {%
4054   \PackageError{glossaries-extra}{\string\makeglossaries\space
4055     not permitted\MessageBreak with record=\@glxtr@record@setting\space
4056     package option}%
4057   {You may only use \string\makeglossaries\space with
4058     record=off or record=alsoindex options}%
4059 }%
4060 {%
4061   \ifblank{#1}%
4062   {\@glxtr@org@makeglossaries}%
4063   {%
4064     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
4065     \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4066       not permitted\MessageBreak with record=alsoindex package option}%
4067     {You may only use the hybrid \string\makeglossaries[...]\space with
4068       record=off option}%
4069   }else
4070     \edef\@glxtr@reg@glosslist{#1}%
4071     \ifundef{\glswrite}{\newwrite\glswrite}{}%
4072     \protected@write\@auxout{}{\string\providecommand
4073       \string\@glxtr@reg@glosslist{#1}}
4074     \protected@write\@auxout{}{\string\providecommand
4075       \string\@glxtr@reg@glosslist{#1}}
4076     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4077     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4078     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4079     \write\@auxout{\string\providecommand\string\@glxtr@reg@glosslist{#1}}

```

Iterate through each supplied glossary type and activate it.

```

4080   \@for\@glo@type:=#1\do{%
4081     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4082   }%

```

New glossaries must be created before \makeglossaries:

```

4083   \renewcommand*\newglossary[4][]{%
4084     \PackageError{glossaries}{New glossaries
4085       must be created before \string\makeglossaries}{You need
4086       to move \string\makeglossaries\space after all your
4087       \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

4088   \let\@makeglossary\relax
4089   \let\makeglossary\relax
4090   \renewcommand\makeglossaries[1][]{%

```

Disable all commands that have no effect after \makeglossaries

```

4091   \@disable@onlypremakeg

```

Allow see key:

```

4092   \let\gls@checkseeallowed\relax

```

Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```

4093     \renewcommand*{\@do@seeglossary}[2]{%
4094         \glsdoifexists{##1}%
4095         {%
4096             \edef\@gls@label{\glsdetoklabel{##1}}%
4097             \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4098             \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4099             {\@glsxtr@org@doseeglossary{##1}{##2}}%
4100             {%
4101                 \@glsxtrwrglossmark
4102                 \protected@write\@auxout{}{%
4103                     \string\@gls@reference
4104                     {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
4105             }%
4106         }%
4107     }%
4108 }%

```

Adjust \@do@@wrglossary

```

4109     \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
4110     \def\@do@@wrglossary{%
4111         \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4112         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4113         {\@glsxtr@@do@@wrglossary}%
4114         {\@gls@noidxglossary}%
4115     }%

```

Suppress warning about no \makeglossaries

```

4116     \let\warn@nomakeglossaries\relax
4117     \def\warn@noprintglossary{%
4118         \GlossariesWarningNoLine{No \string\printglossary\space
4119             or \string\printglossaries\space
4120             found.^^J(Remove \string\makeglossaries\space if you don't want
4121             any glossaries.)^^JThis document will not have a glossary}%
4122     }%

```

Only warn for glossaries not listed.

```

4123     \renewcommand{\@gls@noref@warn}[1]{%
4124         \edef\@gls@type{##1}%
4125         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4126         {%
4127             \GlossariesExtraWarning{Can't use
4128                 \string\printnoidxglossary[type={\@gls@type}]
4129                 when '\@gls@type' is listed in the optional argument of
4130                 \string\makeglossaries}%
4131         }%
4132     {%
4133         \GlossariesWarning{Empty glossary for
4134             \string\printnoidxglossary[type={##1}].

```

```

4135         Rerun may be required (or you may have forgotten to use
4136         commands like \string\gls))}%
4137     }%
4138 }%

```

Adjust display number list to check for type:

```

4139     \renewcommand*\glsdisplaynumberlist}[1]{%
4140     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4141     {\@glsxtr@idx@displaynumberlist{##1}}}%
4142     {\@glsxtr@noidx@displaynumberlist{##1}}}%
4143 }%

```

Adjust entry list:

```

4144     \renewcommand*\glsentrynumberlist}[1]{%
4145     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4146     {\@glsxtr@idx@entrynumberlist{##1}}}%
4147     {\@glsxtr@noidx@entrynumberlist{##1}}}%
4148 }%

```

Adjust number list loop

```

4149     \renewcommand*\glsnumberlistloop}[2]{%
4150     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4151     {%
4152         \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4153         not available for glossary ‘##1’}{}%
4154     }%
4155     {\@glsxtr@noidx@numberlistloop{##1}{##2}}}%
4156 }%

```

Only sanitize sort for normal indexing glossaries.

```

4157     \renewcommand*\glsprestandardsort}[3]{%
4158     \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4159     {%
4160         \glsdosanitizesort
4161     }%
4162     {%
4163         \ifglssanitizesort
4164         \@gls@noidx@sanitizesort
4165         \else
4166         \@gls@noidx@nosanitizesort
4167         \fi
4168     }%
4169 }%

```

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

```

4170     \renewcommand*\new@glossaryentry[2]{%
4171     \PackageError{glossaries-extra}{Glossary entries must be defined
4172     in the preamble\MessageBreak when you use the optional argument
4173     of \string\makeglossaries}{Either move your definitions to the
4174     preamble or don't use the optional argument of

```

```

4175         \string\makeglossaries}%
4176     }%

```

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).

```

4177         \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
4178         \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

4179         \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
4180         type=\glsdefaulttype,\@end@glsxtr@gettype
4181         \def\@glo@sorttype{\@glo@default@sorttype}%
4182     }%

```

Check automake setting:

```

4183         \ifglsautomake
4184         \renewcommand*{\@gls@doautomake}{%
4185             \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
4186                 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
4187             }%
4188         }%
4189         \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

4190         \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4191         \fi
4192     }%
4193 }%
4194 }

```

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

rgprintglossary 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.)

```

4195 \newcommand{\@glsxtr@orgprintglossary}[2]{%
4196     \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

4197     \def\glossarytitle{%
4198         \ifcsdef{\@glo@type\@glo@type @title}%
4199         {\csuse{\@glo@type\@glo@type @title}}%
4200         {\glossaryname}}%
4201     \def\glossarytoctitle{\glossarytitle}%
4202     \let\org@glossarytitle\glossarytitle
4203     \def\@glossarystyle{%
4204         \ifx\@glossary@default@style\relax
4205             \GlossariesWarning{No default glossary style provided \MessageBreak
4206                 for the glossary '\@glo@type'. \MessageBreak

```

```

4207      Using deprecated fallback. \MessageBreak
4208      To fix this set the style with \MessageBreak
4209      \string\setglossarystyle\space or use the \MessageBreak
4210      style key=value option}%
4211  \fi
4212 }%
4213 \def\gls@dotocitle{\gls@settocitle{\@glo@type}}%
4214 \let\org@glossaryentrynumbers\glossaryentrynumbers
4215 \bgroup
4216   \@printgloss@setsort
4217   \setkeys{printgloss}{#1}%
4218   \ifx\glossarytitle\org@glossarytitle
4219   \else
4220     \cslet{\@glo@type \@title}{\glossarytitle}%
4221   \fi
4222   \let\currentglossary\@glo@type
4223   \let\org@glossaryentrynumbers\glossaryentrynumbers
4224   \let\glsnonextpages\@glsnonextpages
4225   \let\glsnextpages\@glsnextpages

4226   \glsxtractivatenopost
4227   \gls@dotocitle
4228   \@glossarystyle
4229   \let\gls@org@glossaryentryfield\glossentry
4230   \let\gls@org@glossarysubentryfield\subglossentry
4231   \renewcommand{\glossentry}[1]{%
4232     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4233     \gls@org@glossaryentryfield{##1}%
4234   }%
4235   \renewcommand{\subglossentry}[2]{%
4236     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4237     \gls@org@glossarysubentryfield{##1}{##2}%
4238   }%
4239   \@gls@preglossaryhook
4240   #2%
4241 \egroup
4242 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
4243 \global\let\warn@noprintglossary\relax
4244 }

```

`ractivatenopost` Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```

4245 \newcommand*{\glsxtractivatenopost}{%
4246   \let\nopostdesc\@nopostdesc
4247   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4248 }

```

`lsxtrnopostpunc`

```

4249 \newrobustcmd*{\glsxtrnopostpunc}{%

```

gsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

4250 \newcommand{\@gsxtr@nopostpunc}{%
4251   \let\@gsxtr@org@postdescription\glspostdescription
4252   \ifglsnopostdot
4253     \renewcommand{\glspostdescription}{%
4254       \glsnopostdottrue
4255       \let\glspostdescription\@gsxtr@org@postdescription
4256       \let\glxtrrestorepostpunc\@gsxtr@restore@postpunc
4257       \glxtrpostdescription
4258       \@gsxtr@nopostpunc@postdesc}%
4259   \else
4260     \renewcommand{\glspostdescription}{%
4261       \let\glspostdescription\@gsxtr@org@postdescription
4262       \let\glxtrrestorepostpunc\@gsxtr@restore@postpunc
4263       \glxtrpostdescription
4264       \@gsxtr@nopostpunc@postdesc}%
4265   \fi
4266   \glsnopostdotfalse
4267 }

```

stpunc@postdesc

```

4268 \newcommand*{\@gsxtr@nopostpunc@postdesc}{}

```

estore@postpunc

```

4269 \newcommand*{\@gsxtr@restore@postpunc}{%
4270   \def\@gsxtr@nopostpunc@postdesc{%
4271     \@gsxtr@org@postdescription
4272     \let\@gsxtr@nopostpunc@postdesc\@empty
4273     \let\glxtrrestorepostpunc\@empty
4274   }%
4275 }

```

restorepostpunc Does nothing outside of glossary.

```

4276 \newcommand*{\glxtrrestorepostpunc}{}

```

\@printglossary Redefine.

```

4277 \renewcommand{\@printglossary}[2]{%
4278   \def\@gsxtr@printglossopts{#1}%
4279   \@gsxtr@orgprintglossary{#1}{#2}%
4280 }

```

Add a key that switches off the entry targets:

```

4281 \define@choicekey{printgloss}{target}
4282 [\@gsxtr@printglossval\@gsxtr@printglossnr]%
4283 {true,false}[true]%
4284 {%
4285   \ifcase\@gsxtr@printglossnr

```

```

4286 \def\@glstarget{\@glsdohypertarget}%
4287 \else
4288 \let\@glstarget\@secondoftwo
4289 \fi
4290 }

```

hypernameprefix

```

4291 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:
4292 \define@key{printgloss}{targetnameprefix}{%
4293 \renewcommand{\@glxtrhypernameprefix}{#1}%
4294 }

4295 \define@key{printgloss}{prefix}{%
4296 \renewcommand{\@glo@linkprefix}{#1}%
4297 }

```

\glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```

4298 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4299 \renewcommand{\glsdohypertarget}[2]{%
4300 \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4301 }

```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```

4302 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4303 \def\@glstarget{\glsdohypertarget}%
4304 \fi
4305 %\end{macro}

```

@makeglossaries For the benefit of makeglossaries

```

4306 \newcommand*{\@glxtr@makeglossaries}[1]{}

```

@glxtr@gettype Get just the type.

```

4307 \def\@glxtr@gettype#1,type=#2,#3\end@glxtr@gettype{%
4308 \def\@glo@type{#2}%
4309 }

```

@assign@sortkey Assign the sort key.

```

4310 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4311 \edef\@glo@type{\@glo@type}%
4312 \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4313 {%
4314 \@glo@no@assign@sortkey{#1}%
4315 }%
4316 {%
4317 \@glo@assign@sortkey{#1}%
4318 }%
4319 }%

```


Display number list for the regular version:

splaynumberlist

```
4320 \let\@glstr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4321 \newcommand*{\@glstr@noidx@displaynumberlist}[1]{%
4322   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4323   \ifdef\@gls@loclist
4324   {%
4325     \def\@gls@noidx@loclist@sep{%
4326       \def\@gls@noidx@loclist@sep{%
4327         \def\@gls@noidx@loclist@sep{%
4328           \glsnumlistsep
4329         }%
4330         \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4331       }%
4332     }%
4333     \def\@gls@noidx@loclist@finalsep{}%
4334     \def\@gls@noidx@loclist@prev{}%
4335     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4336     \@gls@noidx@loclist@finalsep
4337     \@gls@noidx@loclist@prev
4338   }%
4339   {%
4340     \glstrundeftag
4341     \glsdoifexists{#1}%
4342     {%
4343       \GlossariesWarning{Missing location list for ‘#1’. Either
4344         a rerun is required or you haven’t referenced the entry.}%
4345     }%
4346   }%
4347 }%
4348
```

And for the number list loop:

@numberlistloop

```
4349 \newcommand*{\@glstr@noidx@numberlistloop}[3]{%
4350   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4351   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4352   \let\@gls@org@glsseeformat\glsseeformat
4353   \let\glsnoidxdisplayloc#2\relax
4354   \let\glsseeformat#3\relax
4355   \ifdef\@gls@loclist
4356   {%
4357     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4358   }%
```

```

4359 {%
4360     \glxtrundeftag
4361     \glsdoifexists{#1}%
4362     {%
4363         \GlossariesWarning{Missing location list for ‘##1’. Either
4364             a rerun is required or you haven’t referenced the entry.}%
4365     }%
4366 }%
4367 \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
4368 \let\glsseeformat\@gls@org\glsseeformat
4369 }%

```

Same for entry number list.

entrynumberlist

```

4370 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4371     \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4372     \ifdef\@gls@loclist
4373     {%
4374         \glsnoidxloclist{\@gls@loclist}%
4375     }%
4376     {%
4377         \glxtrundeftag
4378         \glsdoifexists{#1}%
4379         {%
4380             \GlossariesWarning{Missing location list for ‘#1’. Either
4381                 a rerun is required or you haven’t referenced the entry.}%
4382         }%
4383     }%
4384 }%

```

entrynumberlist

```

4385 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\@glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4386 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4387     \protected@edef\@glsxtr@titlelabel{#1}%
4388     \ifdefvoid\@glsxtr@titlelabel
4389     {}%
4390     {%
4391         \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4392     }%
4393     \ifdefvoid{\@glsxtr@titlelabel}%
4394     {%
4395         \DTLifint{#1}%
4396         {%
4397             \ifnum#1<256\relax
4398                 \edef#2{\char#1\relax}%

```

```

4399     \else
4400     \edef#2{#1}%
4401     \fi
4402 }%
4403 {%
4404     \ifcsundef{#1groupname}%
4405     {\def#2{#1}}%
4406     {\letcs#2{#1groupname}}%
4407 }%
4408 }%
4409 {%
4410     \let#2\@glsxtr@titlelabel
4411 }%
4412 }

```

`g@getgrouptitle` Save original definition of `\@gls@getgrouptitle`

```

4413 \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.

```

4414 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4415   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4416   \@onelevel@sanitize\@glsxtr@titlelabel
4417   \ifcsdef{\@glsxtr@titlelabel}
4418   {\letcs{#2}{\@glsxtr@titlelabel}}%
4419   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4420 }
4421 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

`trsetgrouptitle` Sets the title for the given group label.

```

4422 \newcommand{\glsxtrsetgrouptitle}[2]{%
4423   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4424   \@onelevel@sanitize\@glsxtr@titlelabel
4425   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4426 }

```

`alsetgrouptitle` As above put only locally defines the title.

```

4427 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4428   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4429   \@onelevel@sanitize\@glsxtr@titlelabel
4430   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4431 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4432 \renewcommand*{\glsnavigation}{%
4433   \def\@gls@between{}%
4434   \ifcsundef{\@gls@hypergroup@list@{\@gls@type}}%
4435   {%

```

```

4436 \def\@gls@list{%
4437 }%
4438 {%
4439 \expandafter\let\expandafter\@gls@list
4440 \csname @gls@hypergroup@list@\@glo@type\endcsname
4441 }%
4442 \@for\@gls@tmp:=\@gls@list\do{%
4443 \@gls@between
4444 \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4445 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4446 \let\@gls@between\glshypernavsep
4447 }%
4448 }

```

`@noidx@glossary`

```

4449 \renewcommand*{\@print@noidx@glossary}{%
4450 \ifcsdef{\@glsref@\@glo@type}%
4451 {%
4452 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4453 {%
4454 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4455 }%
4456 {%
4457 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4458 }%
4459 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4460 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4461 \def\@gls@currentlettergroup{%
4462 \begin{theglossary}%
4463 \glossaryheader
4464 \glsresetentrylist
4465 \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
4466 \end{theglossary}%
4467 \glossarypostamble
4468 }%
4469 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4470 \glsxtrifemptyglossary{\@glo@type}%
4471 }%
4472 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4473 \@gls@noref@warn{\@glo@type}%
4474 }%
4475 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4476 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4477   \setentrycounter[#1]{#2}%
4478   \@glsxtr@display@loc#3\empty\end@glsxtr@display@loc{#4}%
4479 }

```

`xtr@display@loc` Patch to check for range formations.

```

4480 \def\@glsxtr@display@loc#1#2\end@glsxtr@display@loc#3{%
4481   \ifx#1(\relax
4482     \glsxtrdisplaystartloc{#2}{#3}%
4483   \else
4484     \ifx#1)\relax
4485       \glsxtrdisplayendloc{#2}{#3}%
4486     \else
4487       \glsxtrdisplaysingleloc{#1#2}{#3}%
4488     \fi
4489   \fi
4490 }

```

`isplayingleloc` Single location.

```

4491 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4492   \csuse{#1}{#2}%
4493 }

```

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.

```

4494 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4495   \edef\glsxtrlocrangefmt{#1}%
4496   \ifx\glsxtrlocrangefmt\empty
4497     \def\glsxtrlocrangefmt{glsnumberformat}%
4498   \fi
4499   \expandafter\glsxtrdisplaysingleloc
4500   \expandafter{\glsxtrlocrangefmt}{#2}%
4501 }

```

`trdisplayendloc` End of a location range.

```

4502 \newcommand*{\glsxtrdisplayendloc}[2]{%
4503   \edef\@glsxtr@tmp{#1}%
4504   \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}}%
4505   \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4506   \else
4507     \GlossariesExtraWarning{Mismatched end location range
4508       (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
4509   \fi
4510   \expandafter\glsxtrdisplayendloohook\expandafter{\@glsxtr@tmp}{#2}%
4511   \expandafter\glsxtrdisplaysingleloc
4512   \expandafter{\glsxtrlocrangefmt}{#2}%
4513   \def\glsxtrlocrangefmt{}%
4514 }

```

`splayendlochook` Allow the user to hook into the end of range command.

```
4515 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```
4516 \newcommand*{\glsxtrlocrangefmt}{}%
```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```
4517 \renewcommand*{\setentrycounter}[2][{}]{%
4518   \def\glsxtrcounterprefix{#1}%
4519   \ifx\glsxtrcounterprefix\@empty
4520     \def\@glo@counterprefix{.}%
4521   \else
4522     \def\@glo@counterprefix{.#1.}%
4523   \fi
4524   \def\glsetentrycounter{#2}%
4525 }
```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```
4526 \def\@gls@removespaces#1 #2\@nil{%
4527   \toks@=\expandafter{\the\toks@#1}%
4528   \ifx\@#2\@nil
4529     \edef\x{\the\toks@}%
4530     \ifx\x\empty
4531       \else
```

Expand location (just in case `\toks@` is needed for something else).

```
4532     \expandafter\glxtrlocationhyperlink\expandafter
4533     \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4534     \fi
4535   \else
4536     \@gls@ReturnAfterFi{%
4537       \@gls@removespaces#2\@nil
4538     }%
4539   \fi
4540 }
```

`locationhyperlink` `\glxtrlocationhyperlink{<counter>}{<prefix>}{<location>}`

```
4541 \newcommand*{\glxtrlocationhyperlink}[3]{%
4542   \ifdefined\glxtrsupplocationurl
4543     {%
4544       \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4545     }%
4546   {%
4547     \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
4548   }%
4549 }
```

supphypernumber

```
4550 \newcommand*{\glxtrsupphypernumber}[1]{%
4551 {%
4552   \glshasattribute{\glscurrententrylabel}{externallocation}%
4553   {%
4554     \def\glxtrsupplocationurl{%
4555       \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4556   }%
4557   {%
4558     \def\glxtrsupplocationurl{}%
4559   }%
4560   \glshypernumber{#1}%
4561 }%
4562 }
```

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

@print@glossary

```
4563 \renewcommand{\@print@glossary}{%
4564   \makeatletter
4565   \@input@{\jobname.\csname @glo@type @in\endcsname}%
4566   \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
4567   {}%
4568   {\glxtrNoGlossaryWarning{\@glo@type}}%
4569   \ifglxindy
4570     \ifcsundef{@xdy@\@glo@type @language}%
4571     {%
4572       \edef\@do@auxoutstuff{%
4573         \noexpand\AtEndDocument{%
4574           \noexpand\immediate\noexpand\write\@auxout{%
4575             \string\providecommand\string\@xdylanguage[2]{}}%
4576           \noexpand\immediate\noexpand\write\@auxout{%
4577             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4578         }%
4579       }%
4580     }%
4581     {%
4582       \edef\@do@auxoutstuff{%
4583         \noexpand\AtEndDocument{%
4584           \noexpand\immediate\noexpand\write\@auxout{%
4585             \string\providecommand\string\@xdylanguage[2]{}}%
4586           \noexpand\immediate\noexpand\write\@auxout{%
4587             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4588               @language\endcsname}}%
4589         }%
4590       }%
4591     }%
4592     \@do@auxoutstuff
```

```

4593 \edef\@do@auxoutstuff{%
4594 \noexpand\AtEndDocument{%
4595 \noexpand\immediate\noexpand\write\@auxout{%
4596 \string\providecommand\string\@gls@codepage[2]{}}%
4597 \noexpand\immediate\noexpand\write\@auxout{%
4598 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4599 }%
4600 }%
4601 \@do@auxoutstuff
4602 \fi
4603 \renewcommand*{\@warn@nomakeglossaries}{%
4604 \GlossariesWarningNoLine{\string\makeglossaries\space
4605 hasn't been used,^^Jthe glossaries will not be updated}%
4606 }%
4607 }

```

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

`oGlsWarningHead` Header message.

```

4608 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4609 This document is incomplete. The external file associated with
4610 the glossary '#1' (which should be called \texttt{#2})
4611 hasn't been created.%
4612 }

```

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

```

4613 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4614 This has probably happened because there are no entries defined
4615 in this glossary.%
4616 }

```

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

```

4617 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4618 If you don't want this glossary,
4619 add \texttt{nomain} to your package option list when you load
4620 \texttt{glossaries-extra.sty}. For example:%
4621 }

```

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

```

4622 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4623 Did you forget to use \texttt{type=#1} when you defined your
4624 entries? If you tried to load entries into this glossary with
4625 \texttt{\string\loadglsentries} did you remember to use
4626 \texttt{[#1]} as the optional argument? If you did, check that
4627 the definitions in the file you loaded all had the type set
4628 to \texttt{\string\glsdefaulttype}.%
4629 }

```

`arningCheckFile` Advisory message to check the file contents.


```

4630 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4631   Check the contents of the file \texttt{#1}. If
4632   it's empty, that means you haven't indexed any of your entries in this
4633   glossary (using commands like \texttt{\string\gls} or
4634   \texttt{\string\glsadd}) so this list can't be generated.
4635   If the file isn't empty, the document build process hasn't been
4636   completed.%
4637 }

```

WarningAutoMake Message when automake option has been used.

```

4638 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4639   You may need to rerun \LaTeX. If you already have, it may be that
4640   \TeX's shell escape doesn't allow you to run
4641   \ifglxindy xindy\else makeindex\fi. Check the
4642   transcript file \texttt{\jobname.log}. If the shell escape is
4643   disabled, try one of the following:
4644
4645   \begin{itemize}
4646     \item Run the external (Lua) application:
4647
4648       \texttt{makeglossaries-lite.lua \string\jobname\string}
4649
4650     \item Run the external (Perl) application:
4651
4652       \texttt{makeglossaries \string\jobname\string}
4653   \end{itemize}
4654
4655   Then rerun \LaTeX\ on this document.
4656   \GlossariesExtraWarning{Rerun required to build the
4657   glossary '#1' or check TeX's shell escape allows
4658   you to run \ifglxindy xindy\else makeindex\fi}%
4659 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4660 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4661   You need to either replace \texttt{\string\makenoidxglossaries}
4662   with \texttt{\string\makeglossaries} or replace
4663   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4664   \texttt{\string\printnoidxglossary}
4665   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4666   this document.%
4667 }

```

WarningBuildInfo Build advice.

```

4668 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4669   Try one of the following:
4670   \begin{itemize}
4671     \item Add \texttt{automake} to your package option list when you load

```

```

4672      \texttt{glossaries-extra.sty}. For example:
4673
4674      \texttt{\string\usepackage[automake]%
4675              \glsopenbrace glossaries-extra\glsclosebrace}
4676
4677      \item Run the external (Lua) application:
4678
4679      \texttt{makeglossaries-lite.lua \string\jobname\string"}
4680
4681      \item Run the external (Perl) application:
4682
4683      \texttt{makeglossaries \string\jobname\string"}
4684  \end{itemize}
4685
4686  Then rerun \LaTeX\ on this document.%
4687 }

```

trRecordWarning Paragraph for record=only.

```

4688 \newcommand{\GlsXtrRecordWarning}[1]{%
4689 \texttt{\string\printglossary} doesn't work
4690 with the \texttt{record=only} package option
4691 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4692 instead (or change the package option).%
4693 }

```

oGlsWarningTail Final paragraph.

```

4694 \newcommand{\GlsXtrNoGlsWarningTail}{%
4695 This message will be removed once the problem has been fixed.%
4696 }

```

GlsWarningNoOut No out file created. Build advice.

```

4697 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4698 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4699 \texttt{\string\makeglossaries} or you have used
4700 \texttt{\string\nofiles}. If this is just a draft version of the
4701 document, you can suppress this message using the
4702 \texttt{\nomissinggls\text} package option.%
4703 }

```

glossarywarning

```

4704 \newcommand*{@glsxtr@defaultnoglossarywarning}[1]{%
4705 \glossarysection[\glossarytoctitle]{\glossarytitle}
4706 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glo@type @in\endcsname}
4707 \par
4708 \glsxtrifemptyglossary{#1}%
4709 {%
4710 \GlsXtrNoGlsWarningEmptyStart\space
4711 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4712 \medskip

```

```

4713 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4714 \glsopenbrace glossaries-extra\glsclosebrace}
4715 \medskip
4716 }%
4717 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4718 }%
4719 {%
4720 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4721 {%
4722 \GlsXtrNoGlsWarningCheckFile
4723 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4724
4725 \ifglsautomake
4726
4727 \GlsXtrNoGlsWarningAutoMake{#1}
4728
4729 \else
4730
4731 \ifthenelse{\equal{#1}{main}}%
4732 {%
4733 \GlsXtrNoGlsWarningEmptyMain\par
4734 \medskip
4735 \noindent\texttt{\string\usepackage[nomain]%
4736 \glsopenbrace glossaries-extra\glsclosebrace}
4737 \medskip
4738 }%
4739 {}%
4740
4741 \ifdefequal\makeglossaries\@no@makeglossaries
4742 {%
4743 \GlsXtrNoGlsWarningMisMatch
4744 }%
4745 {%
4746 \GlsXtrNoGlsWarningBuildInfo
4747 }%
4748 \fi
4749 }%
4750 {%
4751 \GlsXtrNoGlsWarningNoOut
4752 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4753 }%
4754 }%
4755 \par
4756 \GlsXtrNoGlsWarningTail
4757 }

```

glossarywarning Warn about using `\printglossary` with record

```

4758 \newcommand*{\@glsxtr@record@noglossarywarning}[1]{%
4759 \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak

```

```

4760 with record=only package option\MessageBreak(use
4761 \string\printunsrtglossary[type=#1])\MessageBreak
4762 instead (or change the package option)}%
4763 \glossarysection[\glossarytoctitle]{\glossarytitle}
4764 \GlsXtrRecordWarning{#1}
4765 \GlsXtrNoGlsWarningTail
4766 }

```

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.

```

4767 \newcommand*{\glsxtrresourcefile}[2] [] {%

```

The record option can't be set after this command.

```

4768 \disable@keys{glossaries-extra.sty}{record}%
4769 \glsxtr@writefields
4770 \protected@write\@auxout{\glsxtrresourceinit}{\string\glsxtr@resource{#1}{#2}}%
4771 \let\@glsxtr@org@see@noindex\@gl@see@noindex
4772 \let\@gl@see@noindex\relax
4773 \IfFileExists{#2.glstex}%
4774 {%

```

Can't scope \input so save and restore the category code of @ to allow for internal commands in the location list.

```

4775 \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4776 \makeatletter
4777 \@input{#2.glstex}%
4778 \@bibgls@restoreat

```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```

4779 \glsxtr@check@bibgls@nameref
4780 }%
4781 {%
4782 \GlossariesExtraWarning{No file '#2.glstex'}%
4783 }%
4784 \let\@gl@see@noindex\@glsxtr@org@see@noindex
4785 }
4786 \onlypreamble\glsxtrresourcefile

```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```

4787 \newcommand{\glsxtr@check@bibgls@nameref}{%
4788 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4789 \ifdef\bibglshrefchar
4790 {}%
4791 {%
4792 \GlossariesExtraWarning{record=nameref requires at least
4793 version 1.8 of bib2gls}%
4794 }%

```

```

4795 \fi
4796 \let\@glxstr@check@bibgls@nameref\relax
4797 }

xtrresourceinit Code used during the protected write operation.
4798 \newcommand*\glxtrresourceinit{}\}

trresourcecount
4799 \newcount\glxtrresourcecount

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.
4800 \newcommand*\GlsXtrLoadResources}[1][]{\%
4801 \ifnum\glxtrresourcecount=0\relax
4802 \glxtrresourcefile[#1]{\jobname}%
4803 \else
4804 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4805 \fi
4806 \advance\glxtrresourcecount by 1\relax
4807 }

glxstr@resource
4808 \newcommand*\glxstr@resource}[2]{\}

\glxstr@fields
4809 \newcommand*\glxstr@fields}[1]{\}

xtr@texencoding
4810 \newcommand*\glxstr@texencoding}[1]{\}

\glxstr@langtag
4811 \newcommand*\glxstr@langtag}[1]{\}

@pluralsuffixes
4812 \newcommand*\glxstr@pluralsuffixes}[4]{\}

tr@shortcutsval
4813 \newcommand*\glxstr@shortcutsval}[1]{\}

sxtr@linkprefix
4814 \newcommand*\glxstr@linkprefix}[1]{\}

xtr@writefields This information only needs to be written once, so disable it after it's been used.
4815 \newcommand*\glxstr@writefields{}\}

```

```

4816 \protected@write\@auxout{}%
4817   {\string\providecommand*{\string\glsxtr@fields}[1]{}%}
4818 \protected@write\@auxout{}%
4819   {\string\providecommand*{\string\glsxtr@resource}[2]{}%}
4820 \protected@write\@auxout{}%
4821   {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}%}
4822 \protected@write\@auxout{}%
4823   {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}%}
4824 \protected@write\@auxout{}%
4825   {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}%}
4826 \protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%

4827 \protected@write\@auxout{}%
4828   {\string\providecommand*{\string\glsxtr@record}[5]{}%}

4829 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4830   \protected@write\@auxout{}%
4831     {\string\providecommand*{\string\glsxtr@record@nameref}[8]{}%}
4832 \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`.

```

4833 \ifdef\CurrentTrackedLanguageTag
4834   {%
4835     \protected@write\@auxout{}{%
4836       \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
4837   }%
4838   {%
4839     \protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
4840       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4841       {\glsxtrabbrvpluralsuffix}}%
4842   \ifdef\inputencodingname
4843     {%
4844       \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
4845     }%
4846   {%

```

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.)

```

4847     \@ifpackageloaded{fontspec}%
4848     {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
4849     {}%
4850   }%
4851 \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.

```

4852 \AtBeginDocument
4853   {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
4854 \let\glsxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. The double-quotes around \jobname have been removed (v1.19) since \jobname will include double-quotes if the file name has spaces.

```
4855 \ifglautomake
4856 \IfFileExists{\jobname.aux}%
4857 {\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.

```
4858 \ifx\@gls@doautomake\@gls@doautomake@err
4859 \let\@gls@doautomake\relax
4860 \fi
4861 \fi
4862 }
```

do@automake@err

```
4863 \newcommand*{\@gls@doautomake@err}{%
4864 \PackageError{glossaries}{You must use
4865 \string\makeglossaries\space with automake=true}
4866 {%
4867 Either remove the automake=true setting or
4868 add \string\makeglossaries\space to your document preamble.%
4869 }%
4870 }
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```
4871 \newcommand*{\glsxtr@record}[5]{}%
```

@record@nameref Used with record=nameref to include current label information.

```
4872 \newcommand*{\glsxtr@record@nameref}[8]{}%
```

r@counterrecord Aux file command.

```
4873 \newcommand*{\glsxtr@counterrecord}[3]{%
4874 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4875 }
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
4876 \newcommand*{\@glsxtr@counterrecordhook}{}%
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4877 \newcommand*{\GlsXtrRecordCounter}[1]{%
4878 \@glsxtr@recordcounter{#1}%
4879 }
4880 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4881 \newcommand*{\@glstr@docounterrecord}[1]{%
4882   \protected@write\@auxout{}\string\glstr@counterrecord
4883     {\@glsl@label}{#1}{\csuse{the#1}}}%
4884 }
```

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.

```
4885 \newcommand*{\glstrglossentry}[1]{%
4886   \glstrtitleorpdforheading
4887   {\@glstrglossentry{#1}}%
4888   {\glsentryname{#1}}%
4889   {\glstrheadname{#1}}%
4890 }
```

lsxtrglossentry Another test is needed in case `\@glstrglossentry` has been written to the table of contents.

```
4891 \newrobustcmd*{\@glstrglossentry}[1]{%
4892   \glstrtitleorpdforheading
4893   {%
4894     \glstoifexists{#1}%
4895     {%
4896       \begingroup
4897         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4898         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4899         \ifglshasparent{#1}%
4900           {\GlsXtrStandaloneSubEntryItem{#1}}%
4901           {\glsentryitem{#1}}%
4902           \GlsXtrStandaloneEntryName{#1}%
4903         \endgroup
4904       }%
4905     }%
4906     {\glsentryname{#1}}%
4907     {\glstrheadname{#1}}%
4908 }
```

daloneEntryName

```
4909 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
4910   \glstarget{#1}{\glossentryname{#1}}%
4911 }
```

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.)


```
4912 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}
```

oneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```
4913 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4914   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}}%
4915 }
```

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.

```
4916 \newcommand*{\glsxtrglossentryother}[3]{%
4917   \ifstrempy{#1}%
4918   {%
4919     \ifcsdef{glsxtrhead#3}%
4920     {%
4921       \glsxtrtitleorpdforheading
4922       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4923       {\@gls@entry@field{#2}{#3}}%
4924       {\csuse{glsxtrhead#3}{#2}}%
4925     }%
4926     {%
4927       \glsxtrtitleorpdforheading
4928       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4929       {\@gls@entry@field{#2}{#3}}%
4930       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4931     }%
4932   }%
4933   {%
4934     \glsxtrtitleorpdforheading
4935     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4936     {\@gls@entry@field{#2}{#3}}%
4937     {#1}%
4938   }%
4939 }
```

glossentryother As \@glsxtrglossentry but uses a different field.

```
4940 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
4941   \glsxtrtitleorpdforheading
4942   {%
4943     \glsdoifexists{#1}%
4944     {%
4945       \begingroup
4946       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4947       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4948       \ifglshasparent{#1}%
4949       {\GlsXtrStandaloneSubEntryItem{#1}}%
4950       {\glsentryitem{#1}}%
4951       \GlsXtrStandaloneEntryOther{#1}%

```

```

4952     \endgroup
4953   }%
4954 }%
4955 {\@gls@entry@field{#1}{#2}}%
4956 {#3}%
4957 }

```

aloneEntryOther

```

4958 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
4959   \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4960 }

```

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

```

4961 \newcommand*{\printunsrtglossary}{%
4962   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4963 }

```

ntunsrtglossary Unstarred version.

```

4964 \newcommand*{\@printunsrtglossary}[1][ ]{%
4965   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4966 }

```

ntunsrtglossary Starred version.

```

4967 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4968   \begingroup
4969     #2%
4970     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4971   \endgroup
4972 }

```

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

```

4973 \newcommand*{\printunsrtglossaries}{%
4974   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4975 }

```

@unsrt@glossary

```

4976 \newcommand*{\@print@unsrt@glossary}{%
4977   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4978   \glossarypreamble
4979   check for empty list
4979   \glstrifemptyglossary{\@glo@type}%
4980   {%
4981     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4982   }%
4983   {%

```

```

4984 \key@ifundefined{glossentry}{group}%
4985 {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4986 {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4987 \def\@gls@currentlettergroup{}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

4988 \def\@glsxtr@doglossary{%
4989 \begin{theglossary}%
4990 \glossaryheader
4991 \glsresetentrylist
4992 }%
4993 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4994 : \expandafter=\csname glo@list@\@glo@type\endcsname\do{%
4995 \ifdefempty{\glscurrententrylabel}
4996 {}%
4997 {%

```

Provide a hook (for example to measure width).

```

4998 \let\glsxtr@process\@firstofone
4999 \let\printunsrtglossaryskipentry
5000 \@glsxtr@printunsrtglossaryskipentry
5001 \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5002 \glsxtr@process
5003 {%
5004 \ifglshasparent{\glscurrententrylabel}{}%
5005 {%
5006 \@glsxtr@checkgroup\glscurrententrylabel
5007 \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
5008 {\@glsxtr@groupheading}%
5009 }%
5010 \eappto\@glsxtr@doglossary{%
5011 \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5012 }%
5013 }%
5014 }%
5015 \appto\@glsxtr@doglossary{\end{theglossary}}%
5016 \printunsrtglossarypredoglossary
5017 \@glsxtr@doglossary
5018 }%
5019 \glossarypostamble
5020 }

```

entryprocesshook

```

5021 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{%

```

ossaryskipentry

```

5022 \newcommand*{\printunsrtglossaryskipentry}{%
5023 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space

```

```
5024 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5025 }
```

entryprocesshook

```
5026 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
5027   \let\glxtr@process\@gobble
5028 }
```

rypredoglossary

```
5029 \newcommand*{\printunsrtglossarypredoglossary}{}%
```

lossary@handler

```
5030 \newcommand{\@printunsrt@glossary@handler}[1]{%
5031   \xdef\glscurrententrylabel{#1}%
5032   \printunsrtglossaryhandler\glscurrententrylabel
5033 }
```

glossaryhandler

```
5034 \newcommand{\printunsrtglossaryhandler}[1]{%
5035   \glxtrunsrtdo{#1}%
5036 }
```

xtriflabelinlist

```
\glxtriflabelinlist{<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.

```
5037 \newrobustcmd*{\glxtriflabelinlist}[4]{%
5038   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
5039   \@glxtr@doiflabelinlist{#3}{#4}%
5040 }
```

srtglossaryunit

```
5041 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5042   \s@printunsrtglossary[type=\glstypedefaulttype,#1]{%
5043     \printunsrtglossaryunitsetup{#2}%
5044   }%
5045 }
```

ossaryunitsetup

```
5046 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
5047   \renewcommand{\printunsrtglossaryhandler}[1]{%
5048     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5049     {\glxtrunsrtdo{##1}}%
5050   }%
5051 }
```

Only the target names should have the prefixes adjusted as \gls etc need the original \glslinkprefix. The \@gobble part discards \glslinkprefix.

```

5052 \ifcsundef{theH#1}%
5053 {%
5054 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
5055 }%
5056 {%
5057 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
5058 }%
5059 \renewcommand*{\glossarysection}[2][{}]{%
5060 \appto\glossarypostamble{\glspar\medskip\glspar}%
5061 }

```

srtglossaryunit

```

5062 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
5063 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5064 requires the record=only or record=alsoindex package option}{}%
5065 }

```

t@getgrouptitle

```

5066 \newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
5067 \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
5068 \@onelevel@sanitize\@glsxtr@titlelabel
5069 \ifcsdef{\@glsxtr@titlelabel}
5070 {\letcs{#2}{\@glsxtr@titlelabel}}%
5071 {\def#2{#1}}%
5072 }

```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```

5073 \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).

```

5074 \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.

```

5075 \newcommand*{\@glsxtr@checkgroup}[1]{%
5076 \def\@glsxtr@groupheading{}%
5077 \key@ifundefined{glossentry}{group}%
5078 {%

```

```

5079 \letcs{\@gls@sort}{glo@glstetoklabel{#1}@sort}%
5080 \expandafter\glo@grabfirst\@gls@sort{}\@nil
5081 }%
5082 {%
5083 \protected@edef\@glo@thislettergrp{%
5084 \csuse{glo@glstetoklabel{#1}@glstxtrgroupfield}}%
5085 }%
5086 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5087 {}%
5088 {%
5089 \ifdefempty{\@gls@currentlettergroup}{}%
5090 {\def\@glstxtr@groupheading{\glsgroupskip}}%
5091 \eappto\@glstxtr@groupheading{%
5092 \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
5093 }%
5094 }%
5095 \let\@gls@currentlettergroup\@glo@thislettergrp
5096 }

```

trLocationField Stores the internal name of the location field.

```

5097 \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.

```

5098 \newcommand{\@glstxtr@noidx@do}[1]{%
5099 \ifglstentryexists{#1}%
5100 {%
5101 \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
5102 \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@GlsXtrLocationField}%
5103 \ifglshasparent{#1}%
5104 {%
5105 \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
5106 \ifdefvoid{\@gls@location}%
5107 {%
5108 \ifdefvoid{\@gls@loclist}%
5109 {%
5110 \subglossentry{\gls@level}{#1}{}%
5111 }%
5112 {%
5113 \subglossentry{\gls@level}{#1}%
5114 {%
5115 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5116 }%
5117 }%
5118 }%
5119 }%
5120 \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5121 }%

```

```

5122 }%
5123 {%

5124 \ifdefvoid{\@gls@location}%
5125 {%
5126 \ifdefvoid{\@gls@loclist}
5127 {%
5128 \glossentry{#1}{}%
5129 }%
5130 {%
5131 \glossentry{#1}%
5132 {%
5133 \glossaryentrynumbers{\@glsnoidxloclist{\@gls@loclist}}%
5134 }%
5135 }%
5136 }%
5137 {%
5138 \glossentry{#1}%
5139 {%
5140 \glossaryentrynumbers{\@gls@location}%
5141 }%
5142 }%
5143 }%
5144 }%
5145 {}%
5146 }

```

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`

```

5147 \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`

```

5148 \newcommand*{\@glsxtr@providenewgls}{%
5149 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
5150 \let\@glsxtr@providenewgls\relax
5151 }

```

`identifyglslike` Identify the command given in the second argument for the benefit of **bib2gls**.

```

5152 \newcommand{\@glsxtridentifyglslike}[2]{%
5153 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
5154 }%
5155 {%

```

```

5156 \@glxstr@providenewgls
5157 \protected@write\@auxout{}\string\@glxstr@newglslike{#1}\string#2}}%
5158 }%
5159 }

```

```
\@glxstrnewgls \glxstrnewgls[<options>]{<prefix>}{<cs>}{<inner cs name>}
```

```

5160 \newcommand*{\@glxstrnewgls}[4]{%
5161 \ifdef{#3}%
5162 {%
5163 \PackageError{glossaries-extra}{Command \string#3\space already
5164 defined}{}%
5165 }%
5166 {%

```

Write information to the aux file for bib2gls.

```

5167 \glxstridentifyglslike{#2}{#3}%
5168 \ifcsdef{@#4like@#2}%
5169 {%
5170 \advance\@glxstrnewgls@inner by \@ne
5171 \def\@glxstrnewgls@innercsname{@#4like\number\@glxstrnewgls@inner @#2}%
5172 }%
5173 {\def\@glxstrnewgls@innercsname{@#4like@#2}}%
5174 \expandafter\newrobustcmd\expandafter*\expandafter
5175 #3\expandafter{\expandafter\@glshyp@opt\csname\@glxstrnewgls@innercsname\endcsname}%
5176 \ifstrempy{#1}%
5177 {%
5178 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][]{%
5179 \new@ifnextchar[%
5180 {\csname @#4@\endcsname{##1}{#2##2}}%
5181 {\csname @#4@\endcsname{##1}{#2##2}[]}%
5182 }%
5183 }%
5184 {%
5185 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][]{%
5186 \new@ifnextchar[%
5187 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5188 {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
5189 }%
5190 }%
5191 }%
5192 }

```

```
\glxstrnewgls \glxstrnewgls[<options>]{<prefix>}{<cs>}
```


The first argument prepends to the options and the second argument is the prefix.

```
5193 \newrobustcmd*{\glstrnewgls}[3] [] {%
5194   \@glstrnewgls{#1}{#2}{#3}{gls}%
5195 }
```

glstrnewglslike 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.

```
5196 \newrobustcmd*{\glstrnewglslike}[6] [] {%
5197   \@glstrnewgls{#1}{#2}{#3}{gls}%
5198   \@glstrnewgls{#1}{#2}{#4}{glspl}%
5199   \@glstrnewgls{#1}{#2}{#5}{Gls}%
5200   \@glstrnewgls{#1}{#2}{#6}{Glspl}%
5201 }
```

glstrnewGLSlike 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.

```
5202 \newrobustcmd*{\glstrnewGLSlike}[4] [] {%
5203   \@glstrnewgls{#1}{#2}{#3}{GLS}%
5204   \@glstrnewgls{#1}{#2}{#4}{GLSpl}%
5205 }
```

\glstrnewrgls As `\glstrnewgls` but for `\rgls`.

```
5206 \newrobustcmd*{\glstrnewrgls}[3] [] {%
5207   \@glstrnewgls{#1}{#2}{#3}{rgls}%
5208 }
```

glstrnewrglslike As `\glstrnewglslike` but for `\rgls` etc.

```
5209 \newrobustcmd*{\glstrnewrglslike}[6] [] {%
5210   \@glstrnewgls{#1}{#2}{#3}{rgls}%
5211   \@glstrnewgls{#1}{#2}{#4}{rglspl}%
5212   \@glstrnewgls{#1}{#2}{#5}{rGls}%
5213   \@glstrnewgls{#1}{#2}{#6}{rGlspl}%
5214 }
```

glstrnewrGLSlike As `\glstrnewGLSlike` but for `\rGLS` etc.

```
5215 \newrobustcmd*{\glstrnewrGLSlike}[4] [] {%
5216   \@glstrnewgls{#1}{#2}{#3}{rGLS}%
5217   \@glstrnewgls{#1}{#2}{#4}{rGLSpl}%
5218 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
5219 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
5220   \ifcsdef{glo@glsdetoklabel{#1}@recordcount}%
5221   {\csname glo@glsdetoklabel{#1}@recordcount\endcsname}%
5222   {0}%
5223 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
5224 \newcommand*{\GlsXtrRecordCount}[2]{%
5225   \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2}%
5226   {\csname glo@glstetoklabel{#1}@recordcount.#2\endcsname}%
5227   {0}%
5228 }
```

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 `\glstxtrdetoklocation` can be set to something sensible.

```
5229 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
5230   \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2.\glstxtrdetoklocation{#3}}%
5231   {\csname glo@glstetoklabel{#1}@recordcount.#2.\glstxtrdetoklocation{#3}\endcsname}%
5232   {0}%
5233 }
```

trdetoklocation

```
5234 \newcommand*{\glstxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
5235 \newcommand*{\glstxtrenablerecordcount}{%
5236   \renewcommand*{\gls}{\rgls}%
5237   \renewcommand*{\Gls}{\rGls}%
5238   \renewcommand*{\glspl}{\rglspl}%
5239   \renewcommand*{\Glspl}{\rGlspl}%
5240   \renewcommand*{\GLS}{\rGLS}%
5241   \renewcommand*{\GLSpl}{\rGLSpl}%
5242 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
5243 \newcommand*{\glstxrrecordtriggervalue}[1]{%
5244   \GlsXtrTotalRecordCount{#1}%
5245 }
```

dCountAttribute

```
5246 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
5247   \@for\@glstxr@cat:=#1\do
5248   {%
5249     \ifdefempty{\@glstxr@cat}{}%
5250     {%
5251       \glsssetcategoryattribute{\@glstxr@cat}{recordcount}{#2}%
5252     }%
5253   }%
5254 }
```

trifrecordtrigger

<code>\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}</code>

```
5255 \newcommand*{\glxtrifrecordtrigger}[3]{%
5256   \glshasattribute{#1}{recordcount}%
5257   {%
5258     \ifnum\glxtrrecordtriggervalue{#1}>\glsetattribute{#1}{recordcount}\relax
5259     #3%
5260   \else
5261     #2%
5262   \fi
5263 }%
5264 {#3}%
5265 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
5266 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
5267   \edef\glslabel{\glsdetoklabel{#2}}%
5268   \let\@gls@link@label\glslabel
5269   \def\@glxtr@thevalue{%
5270     \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5271     \def\@glsnumberformat{glstriggerrecordformat}%
5272     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5273     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5274     \def\@glxtr@thevalue{%
5275       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5276       \glxtrinitwrgloss
5277       \glslinkpresetkeys
5278       \setkeys{glslink}{#1}%
5279       \glslinkpostsetkeys
5280       \ifdefempty{\@glxtr@thevalue}%
5281       {%
5282         \@gls@saveentrycounter
5283       }%
5284       {%
5285         \let\thegl Sentrycounter\@glxtr@thevalue
5286         \def\theHglSentrycounter{\@glxtr@theHvalue}%
5287       }%
5288       \ifglxtrinitwrglossbefore
5289         \@do@wrglossary{#2}%
5290       \fi
5291       #3%
5292       \ifglxtrinitwrglossbefore
5293       \else
5294         \@do@wrglossary{#2}%
5295       \fi
5296       \ifKV@glslink@local
5297         \glslocalunset{#2}%
5298       \fi
5299     }%
5300   }
```

```

5298 \else
5299 \glsunset{#2}%
5300 \fi
5301 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5302 \newcommand*{\glstriggerrecordformat}[1]{%

```

```

\rgls

```

```

5303 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rgls

```

```

5304 \newcommand*{\@rgls}[2][{}]{%
5305 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5306 }

```

```

\@rgls@

```

```

5307 \def\@rgls@#1#2[#3]{%
5308 \glstrifrecordtrigger{#2}%
5309 {%
5310 \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5311 }%
5312 {%
5313 \@gls@{#1}{#2}[#3]%
5314 }%
5315 }%

```

```

\rglspl

```

```

5316 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

```

```

\@rglspl

```

```

5317 \newcommand*{\@rglspl}[2][{}]{%
5318 \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5319 }

```

```

\@rglspl@

```

```

5320 \def\@rglspl@#1#2[#3]{%
5321 \glstrifrecordtrigger{#2}%
5322 {%
5323 \@glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5324 }%
5325 {%
5326 \@glspl@{#1}{#2}[#3]%
5327 }%
5328 }%

```

```

\rGls

```

```

5329 \newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}

```

```

\@rGls
5330 \newcommand*{\@rGls}[2][{}]{%
5331   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5332 }

\@rGls@
5333 \def\@rGls@#1#2[#3]{%
5334   \glstriferecordtrigger{#2}%
5335   {%
5336     \@glstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5337   }%
5338   {%
5339     \@Gls@{#1}{#2}[#3]%
5340   }%
5341 }%

\rGlspl
5342 \newrobustcmd*{\rGlspl}{\@glshyp@opt\@rGlspl}

\@rGlspl
5343 \newcommand*{\@rGlspl}[2][{}]{%
5344   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5345 }

\@rGlspl@
5346 \def\@rGlspl@#1#2[#3]{%
5347   \glstriferecordtrigger{#2}%
5348   {%
5349     \@glstr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5350   }%
5351   {%
5352     \@Glspl@{#1}{#2}[#3]%
5353   }%
5354 }%

\rGLS
5355 \newrobustcmd*{\rGLS}{\@glshyp@opt\@rGLS}

\@rGLS
5356 \newcommand*{\@rGLS}[2][{}]{%
5357   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5358 }

\@rGLS@
5359 \def\@rGLS@#1#2[#3]{%
5360   \glstriferecordtrigger{#2}%
5361   {%
5362     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%

```

```

5363 }%
5364 {%
5365 \@GLS@{#1}{#2}[#3]%
5366 }%
5367 }%

```

\rGLSp1

```
5368 \newrobustcmd*{\rGLSp1}{\@gls@hyp@opt\rGLSp1}
```

\@rGLSp1

```

5369 \newcommand*{\@rGLSp1}[2][{}]{%
5370 \new@ifnextchar[{\@rGLSp1@{#1}{#2}}{\@rGLSp1@{#1}{#2}[{}]}%
5371 }

```

\@rGLSp1@

```

5372 \def\@rGLSp1@#1#2[#3]{%
5373 \glsxtrifrecordtrigger{#2}%
5374 {%
5375 \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSp1format{#2}{#3}}%
5376 }%
5377 {%
5378 \@GLSp1@{#1}{#2}[#3]%
5379 }%
5380 }%

```

\rglsformat

```

5381 \newcommand*{\rglsformat}[2]{%
5382 \glsifregular{#1}
5383 {\glsentryfirst{#1}}%
5384 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5385 }

```

\rglsp1format

```

5386 \newcommand*{\rglsp1format}[2]{%
5387 \glsifregular{#1}
5388 {\glsentryfirstplural{#1}}%
5389 {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5390 }

```

\rGlsformat

```

5391 \newcommand*{\rGlsformat}[2]{%
5392 \glsifregular{#1}
5393 {\Glsentryfirst{#1}}%
5394 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5395 }

```

\rGlsplformat

```
5396 \newcommand*{\rGlsplformat}[2]{%
```

```

5397 \glsifregular{#1}
5398 {\Glsentryfirstplural{#1}}%
5399 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5400 }

```

\rGLSformat

```

5401 \newcommand*{\rGLSformat}[2]{%
5402 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5403 }

```

\rGLSplformat

```

5404 \newcommand*{\rGLSplformat}[2]{%
5405 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5406 }

```

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@glsxtr@linkcount@<label> where <label> is the entry’s label. Since this is performed within \@gls@link the label can be accessed with \glslabel.

```

5407 \newcommand{\@glsxtr@do@inc@linkcount}{%
    Does this entry have the linkcount attribute set?
5408 \glsifattribute{\glslabel}{linkcount}{true}%
5409 {%
    Does the counter exist?
5410 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5411 {%
    Counter doesn't exist, so define it.
5412 \newcounter{glsxtr@linkcount@\glslabel}%
    If linkcountmaster is set, add to counter reset.
5413 \glshasattribute{\glslabel}{linkcountmaster}%
5414 {%
    Need to ensure values are fully expanded.
5415 \begingroup
5416 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
5417 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
5418 \x
5419 }%

```

```

5420      {}%
5421    }%
    Increment counter:
5422    \glxstrinclinkcounter{glxstr@linkcount@glslabel}%
5423  }%
5424  {}%
5425 }

rlinkcounter  May be redefined to use \refstepcounter if required.
5426 \newcommand*{\glxstrinclinkcounter}[1]{\stepcounter{#1}}

linkCounterValue  Expands to the associated link counter register or 0 if not defined.
5427 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5428   \ifcsundef{c@glxstr@linkcount@#1}{0}{\csname c@glxstr@linkcount@#1\endcsname}%
5429 }

rTheLinkCounter  Expands to the display value of the associated link counter or 0 if not defined.
5430 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5431   \ifcsundef{theglsxtr@linkcount@#1}{0}%
5432   {\csname theglxstr@linkcount@#1\endcsname}%
5433 }

fLinkCounterDef  Tests if the counter has been defined
5434 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5435   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5436 }

LinkCounterName  Expands to the associated link counter name. (No check for existence.)
5437 \newcommand*{\GlsXtrLinkCounterName}[1]{glxstr@linkcount@#1}

ableLinkCounting  \GlsXtrEnableLinkCounting[master counter]{categories}

    Enable link counting for the given categories.
5438 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5439   \let\glxstr@inc@linkcount\@glxstr@do@inc@linkcount
5440   \@for\@glxstr@label:=#2\do
5441   {%
5442     \glsssetcategoryattribute{\@glxstr@label}{linkcount}{true}%
5443     \ifstrempy{#1}{}%
5444     {%
5445       \ifcsundef{c@#1}%
5446       {\@nocounterr{#1}}%
5447       {\glsssetcategoryattribute{\@glxstr@label}{linkcountmaster}{#1}}%
5448     }%
5449   }%
5450 }
5451 \@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.

```
5452 \@ifpackageloaded{glossaries-accsupp}
5453 {
```

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

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

```
5454 \newcommand*{\glsaccessname}[1]{%
5455 \glsnameaccessdisplay
5456 {%
5457 \glstryname{#1}%
5458 }%
5459 {#1}%
5460 }
```

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

```
5461 \newcommand*{\Glsaccessname}[1]{%
5462 \glsnameaccessdisplay
5463 {%
5464 \Glsstryname{#1}%
5465 }%
5466 {#1}%
5467 }
```

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

```
5468 \newcommand*{\GLSaccessname}[1]{%
5469 \glsnameaccessdisplay
5470 {%
5471 \mfirstucMakeUppercase{\glstryname{#1}}%
5472 }%
5473 {#1}%
5474 }
```

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

```
5475 \newcommand*{\glsaccesstext}[1]{%
5476 \glstextaccessdisplay
5477 {%
5478 \glstrytext{#1}%
5479 }%
5480 {#1}%
5481 }
```

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

```

5482 \newcommand*{\Glsaccesstext}[1]{%
5483   \glstextaccessdisplay
5484   {%
5485     \Glsentrytext{#1}%
5486   }%
5487   {#1}%
5488 }
```

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

```

5489 \newcommand*{\GLSaccesstext}[1]{%
5490   \glstextaccessdisplay
5491   {%
5492     \mfirstucMakeUppercase{\Glsentrytext{#1}}%
5493   }%
5494   {#1}%
5495 }
```

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

```

5496 \newcommand*{\glsaccessplural}[1]{%
5497   \glspluralaccessdisplay
5498   {%
5499     \Glsentryplural{#1}%
5500   }%
5501   {#1}%
5502 }
```

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

```

5503 \newcommand*{\GLSaccessplural}[1]{%
5504   \glspluralaccessdisplay
5505   {%
5506     \Glsentryplural{#1}%
5507   }%
5508   {#1}%
5509 }
```

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

```

5510 \newcommand*{\GLSaccessplural}[1]{%
5511   \glspluralaccessdisplay
5512   {%
5513     \mfirstucMakeUppercase{\Glsentryplural{#1}}%
5514   }%
5515   {#1}%
5516 }
```

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

```

5517 \newcommand*{\glsaccessfirst}[1]{%
5518   \glsfirstaccessdisplay
5519   {%
5520     \glstryfirst{#1}%
5521   }%
5522   {#1}%
5523 }

```

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

```

5524 \newcommand*{\Glsaccessfirst}[1]{%
5525   \glsfirstaccessdisplay
5526   {%
5527     \Glsstryfirst{#1}%
5528   }%
5529   {#1}%
5530 }

```

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

```

5531 \newcommand*{\GLSaccessfirst}[1]{%
5532   \glsfirstaccessdisplay
5533   {%
5534     \mfirstucMakeUppercase{\glstryfirst{#1}}%
5535   }%
5536   {#1}%
5537 }

```

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

```

5538 \newcommand*{\glsaccessfirstplural}[1]{%
5539   \glsfirstpluralaccessdisplay
5540   {%
5541     \glstryfirstplural{#1}%
5542   }%
5543   {#1}%
5544 }

```

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

```

5545 \newcommand*{\Glsaccessfirstplural}[1]{%
5546   \glsfirstpluralaccessdisplay
5547   {%
5548     \Glsstryfirstplural{#1}%
5549   }%
5550   {#1}%
5551 }

```

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

```

5552 \newcommand*{\GLSaccessfirstplural}[1]{%

```

```

5553 \glsfirstpluralaccessdisplay
5554 {%
5555     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5556 }%
5557 {#1}%
5558 }

```

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

```

5559 \newcommand*{\glsaccesssymbol}[1]{%
5560     \glssymbolaccessdisplay
5561     {%
5562         \glsentrysymbol{#1}%
5563     }%
5564     {#1}%
5565 }

```

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

```

5566 \newcommand*{\Glsaccesssymbol}[1]{%
5567     \glssymbolaccessdisplay
5568     {%
5569         \glsentrysymbol{#1}%
5570     }%
5571     {#1}%
5572 }

```

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

```

5573 \newcommand*{\GLSaccesssymbol}[1]{%
5574     \glssymbolaccessdisplay
5575     {%
5576         \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5577     }%
5578     {#1}%
5579 }

```

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

```

5580 \newcommand*{\glsaccesssymbolplural}[1]{%
5581     \glssymbolpluralaccessdisplay
5582     {%
5583         \glsentrysymbolplural{#1}%
5584     }%
5585     {#1}%
5586 }

```

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

```

5587 \newcommand*{\Glsaccesssymbolplural}[1]{%
5588     \glssymbolpluralaccessdisplay

```

```

5589   {%
5590     \Glsentrysymbolplural{#1}%
5591   }%
5592   {#1}%
5593 }

```

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

```

5594 \newcommand*{\GLSaccesssymbolplural}[1]{%
5595   \glssymbolpluralaccessdisplay
5596   {%
5597     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5598   }%
5599   {#1}%
5600 }

```

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

```

5601 \newcommand*{\glsaccessdesc}[1]{%
5602   \glsdescriptionaccessdisplay
5603   {%
5604     \glsentrydesc{#1}%
5605   }%
5606   {#1}%
5607 }

```

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

```

5608 \newcommand*{\Glsaccessdesc}[1]{%
5609   \glsdescriptionaccessdisplay
5610   {%
5611     \Glsentrydesc{#1}%
5612   }%
5613   {#1}%
5614 }

```

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

```

5615 \newcommand*{\GLSaccessdesc}[1]{%
5616   \glsdescriptionaccessdisplay
5617   {%
5618     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5619   }%
5620   {#1}%
5621 }

```

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

```

5622 \newcommand*{\glsaccessdescplural}[1]{%
5623   \glsdescriptionpluralaccessdisplay
5624   {%
5625     \glsentrydescplural{#1}%

```

```

5626     }%
5627     {#1}%
5628 }

```

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

```

5629 \newcommand*{\Glsaccessdescplural}[1]{%
5630   \glsdescriptionpluralaccessdisplay
5631   {%
5632     \Glsentrydescplural{#1}%
5633   }%
5634   {#1}%
5635 }

```

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

```

5636 \newcommand*{\GLSaccessdescplural}[1]{%
5637   \glsdescriptionpluralaccessdisplay
5638   {%
5639     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5640   }%
5641   {#1}%
5642 }

```

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

```

5643 \newcommand*{\glsaccessshort}[1]{%
5644   \glsshortaccessdisplay
5645   {%
5646     \glsentryshort{#1}%
5647   }%
5648   {#1}%
5649 }

```

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

```

5650 \newcommand*{\Glsaccessshort}[1]{%
5651   \glsshortaccessdisplay
5652   {%
5653     \Glsentryshort{#1}%
5654   }%
5655   {#1}%
5656 }

```

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

```

5657 \newcommand*{\GLSAccessshort}[1]{%
5658   \glsshortaccessdisplay
5659   {%
5660     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5661   }%

```

```

5662     {#1}%
5663 }

```

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

```

5664 \newcommand*{\lsaccessshortpl}[1]{%
5665     \glshortpluralaccessdisplay
5666     {%
5667         \glentryshortpl{#1}%
5668     }%
5669     {#1}%
5670 }

```

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

```

5671 \newcommand*{\Lsaccessshortpl}[1]{%
5672     \glshortpluralaccessdisplay
5673     {%
5674         \Glentryshortpl{#1}%
5675     }%
5676     {#1}%
5677 }

```

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

```

5678 \newcommand*{\LSaccessshortpl}[1]{%
5679     \glshortpluralaccessdisplay
5680     {%
5681         \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5682     }%
5683     {#1}%
5684 }

```

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

```

5685 \newcommand*{\glsaccesslong}[1]{%
5686     \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5687 }

```

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

```

5688
5689 \newcommand*{\Glsaccesslong}[1]{%
5690     \glslongaccessdisplay{\Glentrylong{#1}}{#1}%
5691 }

```

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

```

5692 \newcommand*{\GLSaccesslong}[1]{%
5693     \glslongaccessdisplay
5694     {%
5695         \mfirstucMakeUppercase{\glentrylong{#1}}%
5696     }%

```

```

5697     {#1}%
5698 }

glsaccesslongpl Display the long plural form (no link and no check for existence).
5699 \newcommand*{\glsaccesslongpl}[1]{%
5700   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5701 }

Glsaccesslongpl Display the long plural form (no link and no check for existence).
5702
5703 \newcommand*{\Glsaccesslongpl}[1]{%
5704   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5705 }

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.
5706 \newcommand*{\GLSaccesslongpl}[1]{%
5707   \glslongpluralaccessdisplay
5708   {%
5709     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5710   }%
5711   {#1}%
5712 }

Keys for accessibility support.
5713 \define@key{glsxtrabrv}{access}{%
5714   \def\@gls@nameaccess{#1}%
5715 }
5716 \define@key{glsxtrabrv}{textaccess}{%
5717   \def\@gls@textaccess{#1}%
5718 }
5719 \define@key{glsxtrabrv}{firstaccess}{%
5720   \def\@gls@firstaccess{#1}%
5721 }
5722 \define@key{glsxtrabrv}{shortaccess}{%
5723   \def\@gls@shortaccess{#1}%
5724 }
5725 \define@key{glsxtrabrv}{shortpluralaccess}{%
5726   \def\@gls@shortaccesspl{#1}%
5727 }

@initaccesskeys
5728 \newcommand*{\@gls@initaccesskeys}{%
5729   \def\@gls@nameaccess{}%
5730   \def\@gls@textaccess{}%
5731   \def\@gls@firstaccess{}%
5732   \def\@gls@shortaccess{}%
5733   \def\@gls@shortaccesspl{}%
5734 }

```


essattribute@set

```
\gls@ifaccessattribute@set{\<attribute>}{\<true>}{\<false>}
```

```
5735 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5736   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5737   {#2}%
5738   {%
5739     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5740     {#3}%
5741     {%
5742       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5743       {#2}%
5744       {#3}%
5745     }%
5746   }%
5747 }
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5748 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
5749 \ifdefempty\@gls@shortaccess
5750 {%
5751   \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5752   {%
5753     \@glsxtr@insertdots\@gls@shortaccess{#1}%
5754     \eappto\ExtraCustomAbbreviationFields{%
5755       shortaccess={\expandonce\@gls@shortaccess},}%
5756   }%
5757   {}%
5758 }%
5759 {}%
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5760 \ifdefempty\@gls@shortaccess
5761 {}%
5762 {%
5763   \ifdefempty\@gls@shortaccesspl
5764   {%
5765     \@gls@ifaccessattribute@set{aposplural}%
5766     {%
5767       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5768         \@gls@shortaccess'\abbrvpluralsuffix}%
5769     }%
5770   }%
5771   \@gls@ifaccessattribute@set{noshortplural}%
5772   {%
```

```

5773         \let\@gls@shortaccesspl\@gls@shortaccess
5774     }%
5775     {%
5776         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5777             \@gls@shortaccess\abbrvpluralsuffix}%
5778     }%
5779 }%
5780 \eappto\ExtraCustomAbbreviationFields{%
5781     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5782 }%
5783 {}%
5784 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5785 \ifdefempty\@gls@nameaccess
5786 {%
5787     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5788     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5789         \ifdefempty\@gls@shortaccess
5790         {}%
5791         {%
5792             \eappto\ExtraCustomAbbreviationFields{%
5793                 access={\expandonce\@gls@shortaccess},%
5794             }%
5795         }%
5796     }%
5797     {}%
5798 }%
5799 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5800 \ifdefempty\@gls@textaccess
5801 {%
5802     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5803     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5804         \ifdefempty\@gls@shortaccess
5805         {}%
5806         {%
5807             \eappto\ExtraCustomAbbreviationFields{%
5808                 textaccess={\expandonce\@gls@shortaccess},%
5809             }%
5810         }%
5811     }%
5812     {}%
5813 }%
5814 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5815 \ifdefempty\@gls@firstaccess
5816 {%
5817 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5818 {%
```

Do nothing if the shortaccess key hasn't been set.

```
5819 \ifdefempty\@gls@shortaccess
5820 {}%
5821 {%
5822 \eappto\ExtraCustomAbbreviationFields{%
5823 firstaccess={\expandonce\@gls@shortaccess},%
5824 }%
5825 }%
5826 }%
5827 {}%
5828 }%
5829 {}%
5830 }
```

End of if accsupp part

```
5831 }
5832 {
```

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

```
\glsaccessname Display the name value (no link and no check for existence).
5833 \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.
5834 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
5835 \newcommand*{\GLSaccessname}[1]{%
5836 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}}

\glsaccesstext Display the text value (no link and no check for existence).
5837 \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.
5838 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
5839 \newcommand*{\GLSaccesstext}[1]{%
5840 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}}

glsaccessplural Display the plural value (no link and no check for existence).
5841 \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.
5842 \newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}

GLSaccessplural Display the plural value (no link and no check for existence). converted to upper case.
5843 \newcommand*{\GLSaccessplural}[1]{%
5844 \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}

\glsaccessfirst Display the first value (no link and no check for existence).
5845 \newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}

\GLSaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.
5846 \newcommand*{\GLSaccessfirst}[1]{\Glsentryfirst{#1}}

\GLSaccessfirst Display the first value (no link and no check for existence). converted to upper case.
5847 \newcommand*{\GLSaccessfirst}[1]{%
5848 \protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}

cessfirstplural Display the firstplural value (no link and no check for existence).
5849 \newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5850 \newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}

cessfirstplural Display the firstplural value (no link and no check for existence). converted to upper case.
5851 \newcommand*{\GLSaccessfirstplural}[1]{%
5852 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}

glsaccesssymbol Display the symbol value (no link and no check for existence).
5853 \newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}

GLSaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.
5854 \newcommand*{\GLSaccesssymbol}[1]{\Glsentrysymbol{#1}}

GLSaccesssymbol Display the symbol value (no link and no check for existence). converted to upper case.
5855 \newcommand*{\GLSaccesssymbol}[1]{%
5856 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}

esssymbolplural Display the symbolplural value (no link and no check for existence).
5857 \newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
5858 \newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}

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

```
5859 \newcommand*{\GLSaccesssymbolplural}[1]{%
5860 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

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

```
5861 \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.

```
5862 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

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

```
5863 \newcommand*{\GLSaccessdesc}[1]{%
5864 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

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

```
5865 \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.

```
5866 \newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

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

```
5867 \newcommand*{\GLSaccessdescplural}[1]{%
5868 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

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

```
5869 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

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

```
5870 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

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

```
5871 \newcommand*{\GLSaccessshort}[1]{%
5872 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

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

```
5873 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

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

```
5874 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

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

```
5875 \newcommand*{\GLSaccessshortpl}[1]{%
5876 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

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

```
5877 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

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

```
5878 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

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

```
5879 \newcommand*{\GLSaccesslong}[1]{%
5880 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

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

```
5881 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

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

```
5882 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

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

```
5883 \newcommand*{\GLSaccesslongpl}[1]{%
5884 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
5885 \newcommand*{@gls@initaccesskeys}{}%
```

lt@short@access This does nothing if there's no accessibility support.

```
5886 \newcommand{\@gls@setup@default@short@access}[1]{%
End of else part
5887 }
```

1.6 Categories

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

```
5888 \glsaddstoragekey{category}{general}{\glscategory}
```

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

```
5889 \newcommand{\glsifcategory}[4]{%
5890 \ifglstest{#1}{category}{#2}{#3}{#4}%
5891 }
```

Categories can have attributes.

categoryattribute

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

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

```
5892 \newcommand*{\glissetcategoryattribute}[3]{%
5893   \csdef{@glsextr@categoryattr@@#1@#2}{#3}%
5894 }
```

categoryattribute

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

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

```
5895 \newcommand*{\glsggetcategoryattribute}[2]{%
5896   \csuse{@glsextr@categoryattr@@#1@#2}%
5897 }
```

categoryattribute

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

Tests if the category has the given attribute set.

```
5898 \newcommand*{\glshascategoryattribute}[4]{%
5899   \ifcsvoid{@glsextr@categoryattr@@#1@#2}{#4}{#3}%
5900 }
```

\glissetattribute

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

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

```
5901 \newcommand*{\glissetattribute}[3]{%
5902   \glissetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5903 }
```

\glsggetattribute

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

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

```
5904 \newcommand*{\glsggetattribute}[2]{%
5905   \glsggetcategoryattribute{\glscategory{#1}}{#2}%
5906 }
```

`\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.

```
5907 \newcommand*{\glshasattribute}[4]{%
5908   \ifglentryexists{#1}%
5909   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5910   {#4}%
5911 }
```

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

True if category has the attribute with the given value.

```
5912 \newcommand{\glcifcategoryattribute}[5]{%
5913   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5914   {#5}%
5915   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5916 }
```

`\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.

```
5917 \newcommand{\glcifattribute}[5]{%
5918   \ifglentryexists{#1}%
5919   {\glcifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5920   {#5}%
5921 }
```

Set attributes for the default general category:

```
5922 \glsetcategoryattribute{general}{regular}{true}
```

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

```
5923 \glsetcategoryattribute{acronym}{regular}{true}
```

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

```
5924 \newcommand*{\glsetregularcategory}[1]{%
5925   \glsetcategoryattribute{#1}{regular}{true}%
5926 }
```


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

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

```
5927 \newcommand{\glsifregularcategory}[3]{%
5928   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5929 }
```

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

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

```
5930 \newcommand{\glsifnotregularcategory}[3]{%
5931   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5932 }
```

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

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

```
5933 \newcommand{\glsifregular}[3]{%
5934   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
5935 }
```

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

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

```
5936 \newcommand{\glsifnotregular}[3]{%
5937   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5938 }
```

`\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.

```
5939 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```

5940 \forallglossaries[#1]{#3}%
5941 {%
5942     \forallsentries[#3]{#4}%
5943     {%
5944         \glsifcategory{#4}{#2}{#5}{}%
5945     }%
5946 }%
5947 }

```

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.

```

5948 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5949     \forallglossaries[#1]{#4}%
5950     {%
5951         \forallsentries[#4]{#5}%
5952         {%
5953             \glsifattribute{#5}{#2}{#3}{#6}{}%
5954         }%
5955     }%
5956 }

```

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

```

5957 \ifdef\newterm
5958 {%

```

`\newterm`

```

5959     \renewcommand*{\newterm}[2][ ]{%
5960         \newglossaryentry{#2}%
5961         {type=index,category=index,name={#2},%
5962          description={\glstrpostdescription\nopostdesc},#1}%
5963     }

```

Indexed terms are regular by default.

```

5964     \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5965     \newcommand*{\glstrpostdescindex}{}

5966 }
5967 {}

```

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.

```
5968 \ifdef\printsymbols
5969 {%
```

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

```
5970 \newcommand*{\glsxtrnewsymbol}[3] [] {%
5971 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5972 }
```

Symbols are regular by default.

```
5973 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
5974 \newcommand*{\glsxtrpostdescsymbol}{}

5975 }
5976 {%
```

Similar for the numbers option.

```
5977 \ifdef\printnumbers
5978 {%
```

`glsxtrnewnumber`

```
5979 \ifdef\printnumbers
5980 \newcommand*{\glsxtrnewnumber}[3] [] {%
5981 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5982 }
```

Numbers are regular by default.

```
5983 \glsssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
5984 \newcommand*{\glsxtrpostdescnumber}{}

5985 }
5986 {%
```

`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.

```
5987 \newcommand*{\glsxtrsetcategory}[2] {%
5988 \@for\@glsxtr@label:=#1\do
5989 {%
5990 \glsfieldxddef{\@glsxtr@label}{category}{#2}%
5991 }%
5992 }
```

`\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.

```

5993 \newcommand*{\glxtrsetcategoryforall}[2]{%
5994   \forallglossaries[#1]{\@glxtr@type}{%
5995     \forallglsentries[\@glxtr@type]{\@glxtr@label}%
5996     {%
5997       \glsfieldxdef{\@glxtr@label}{category}{#2}%
5998     }%
5999   }%
6000 }

```

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

Apply title casing to the contents of the given field.

```

6001 \newcommand*{\glxtrfieldtitlecase}[2]{%
6002   \expandafter\glxtrfieldtitlecasesecs\expandafter
6003   {\csname glo@\glsdetoklabel{#1}@\#2\endcsname}%
6004 }

```

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

```

6005 \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.

```

6006 \@ifpackageloaded{glossaries-accsupp}
6007 {
6008   \renewcommand*{\glossentrydesc}[1]{%
6009     \glsdoifexistsorwarn{#1}%
6010     {%
6011       \glssetabbrvfmt{\glscategory{#1}}%

```

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

```

6012   \glshasattribute{#1}{glossdescfont}%
6013   {%
6014     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6015     \ifcsdef{\@glxtr@attrval}%
6016     {%
6017       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6018     }%
6019     {%
6020       \GlossariesExtraWarning{Unknown control sequence name
6021         '\@glxtr@attrval' supplied in glossdescfont attribute

```

```

6022         for entry '#1'. Ignoring}%
6023         \let\@glxtr@glossdescfont\@firstofone
6024     }%
6025 }%
6026 {\let\@glxtr@glossdescfont\@firstofone}%
6027 \glsifattribute{#1}{glossdesc}{firstuc}%
6028 {%
6029     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6030 }%
6031 {%
6032     \glsifattribute{#1}{glossdesc}{title}%
6033     {%
6034         \@glxtr@do@titlecaps@warn
6035         \glsdescriptionaccessdisplay
6036         {%
6037             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6038         }%
6039         {#1}%
6040     }%
6041     {%
6042         \@glxtr@glossdescfont{\glsaccessdesc{#1}}%
6043     }%
6044 }%
6045 }%
6046 }
6047 }
6048 {
6049 \renewcommand*{\glossentrydesc}[1]{%
6050     \glsdoifexistsorwarn{#1}%
6051     {%
6052         \glssetabbrvfmt{\glscategory{#1}}%
6053         \glsattribute{#1}{glossdescfont}%
6054         {%
6055             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6056             \ifcsdef{\@glxtr@attrval}%
6057             {%
6058                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6059             }%
6060             {%
6061                 \GlossariesExtraWarning{Unknown control sequence name
6062                 '\@glxtr@attrval' supplied in glossdescfont attribute
6063                 for entry '#1'. Ignoring}%
6064                 \let\@glxtr@glossdescfont\@firstofone
6065             }%
6066         }%
6067         {\let\@glxtr@glossdescfont\@firstofone}%
6068         \glsifattribute{#1}{glossdesc}{firstuc}%
6069         {%
6070             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%

```

```

6071     }%
6072     {%
6073         \glsifattribute{#1}{glossdesc}{title}%
6074         {%
6075             \@glsxtr@do@titlecaps@warn
6076             \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
6077         }%
6078         {%
6079             \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
6080         }%
6081     }%
6082 }%
6083 }
6084 }

```

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

```

6085 \ifpackageloaded{glossaries-accsupp}
6086 {
6087   \renewcommand*{\glossentryname}[1]{%
6088     \@glsdoifexistsorwarn{#1}%
6089     {%
6090       \glssetabbrvfmt{\glscategory{#1}}%

```

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

```

6091   \glsifattribute{#1}{glossnamefont}%
6092   {%
6093     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6094     \ifcsdef{\@glsxtr@attrval}%
6095     {%
6096       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6097     }%
6098     {%
6099       \GlossariesExtraWarning{Unknown control sequence name
6100         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
6101         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6102       \let\@glsxtr@glossnamefont\glsnamefont
6103     }%
6104   }%
6105   {\let\@glsxtr@glossnamefont\glsnamefont}%
6106   \glsifattribute{#1}{glossname}{firstuc}%
6107   {%
6108     \glsnameaccessdisplay
6109     {%
6110       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6111     }%
6112     {#1}%
6113   }%
6114   {%
6115     \glsifattribute{#1}{glossname}{title}%

```

```

6116      {%
6117      \@glstr@do@titlecaps@warn
6118      \glsnameaccessdisplay
6119      {%
6120      \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
6121      }%
6122      {#1}%
6123      }%
6124      {%
6125      \glsifattribute{#1}{glossname}{uc}%
6126      {%
6127      \glsnameaccessdisplay
6128      {%

```

Hide the label from the upper-casing command.

```

6129      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6130      \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6131      }%
6132      {#1}%
6133      }%
6134      {%
6135      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6136      \glsnameaccessdisplay
6137      {%
6138      \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6139      }%
6140      {#1}%
6141      }%
6142      }%
6143      }%

```

Do post-name hook:

```

6144      \glstrpostnamehook{#1}%
6145      }%
6146  }
6147 }
6148 {
6149 \renewcommand*{\glossentryname}[1]{%
6150   \@glsdofexistsorwarn{#1}%
6151   {%
6152     \glsetabbrvfmt{\glscategory{#1}}%
6153     \glshasattribute{#1}{glossnamefont}%
6154     {%
6155       \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
6156       \ifcsdef{\@glstr@attrval}%
6157       {%
6158         \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6159       }%
6160     }%
6161     \GlossariesExtraWarning{Unknown control sequence name

```

```

6162         ‘\@glstr@attrval’ supplied in glossnamefont attribute
6163         for entry ‘#1’. Reverting to default \string\glstrfont}%
6164         \let\@glstr@glossnamefont\glstrfont
6165     }%
6166 }%
6167 {\let\@glstr@glossnamefont\glstrfont}%
6168 \glstrattribute{#1}{glossname}{firstuc}%
6169 {%
6170     \@glstr@glossnamefont{\Glsentryname{#1}}%
6171 }%
6172 {%
6173     \glstrattribute{#1}{glossname}{title}%
6174 {%
6175     \@glstr@do@titlecaps@warn
6176     \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
6177 }%
6178 {%
6179     \glstrattribute{#1}{glossname}{uc}%
6180     {%

```

Hide the label from the upper-casing command.

```

6181         \letcs{\glo@name}{glo\glstrdetoklabel{#1}@name}%
6182         \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6183     }%
6184     {%

```

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

```

6185         \letcs{\glo@name}{glo\glstrdetoklabel{#1}@name}%
6186         \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6187     }%
6188 }%
6189 }%

```

Do post-name hook.

```

6190     \glstrpostnamehook{#1}%
6191 }%
6192 }
6193 }

```

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

```

6194 \@ifpackageloaded{glossaries-accsupp}
6195 {
6196     \renewcommand*{\Glsentryname}[1]{%
6197         \@glstrdoifexistsorwarn{#1}%
6198         {%
6199             \glstrsetabbrvfmt{\glscategory{#1}}%

```

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

```

6200     \glshasattribute{#1}{glossnamefont}%
6201     {%

```



```

6202     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6203     \ifcsdef{\@glxtr@attrval}%
6204     {%
6205         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6206     }%
6207     {%
6208         \GlossariesExtraWarning{Unknown control sequence name
6209             '\@glxtr@attrval' supplied in glossnamefont attribute
6210             for entry '#1'. Reverting to default \string\glsnamefont}%
6211         \let\@glxtr@glossnamefont\glsnamefont
6212     }%
6213 }%
6214 {\let\@glxtr@glossnamefont\glsnamefont}%
6215 \glsnameaccessdisplay
6216 {%
6217     \@glxtr@glossnamefont{\Glsentryname{#1}}%
6218 }%
6219 {#1}%

```

Do post-name hook:

```

6220     \glxtrpostnamehook{#1}%
6221 }%
6222 }
6223 }
6224 {
6225     \renewcommand*{\Glossentryname}[1]{%
6226         \@glsdoifexistsorwarn{#1}%
6227     }%
6228     \glsssetabbrvfmt{\glscategory{#1}}%
6229     \glshasattribute{#1}{glossnamefont}%
6230     {%
6231         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6232         \ifcsdef{\@glxtr@attrval}%
6233         {%
6234             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6235         }%
6236         {%
6237             \GlossariesExtraWarning{Unknown control sequence name
6238                 '\@glxtr@attrval' supplied in glossnamefont attribute
6239                 for entry '#1'. Reverting to default \string\glsnamefont}%
6240             \let\@glxtr@glossnamefont\glsnamefont
6241         }%
6242     }%
6243     {\let\@glxtr@glossnamefont\glsnamefont}%
6244     \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6245     \glxtrpostnamehook{#1}%
6246 }%
6247 }

```

6248 }

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.

```
6249 \newcommand*{\glstrpostnamehook}[1]{%
6250   \let\@glsnumberformat\@glstr@defaultnumberformat
6251   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
6252   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
6253   \csuse{glstrpostname\glscategory{#1}}%
6254 }
```

`glsextrapostnamehook`

```
6255 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
6256 \newcommand*{\glsdefpostname}[2]{%
6257   \csdef{glstrpostname#1}{#2}%
6258 }
```

`glsaccessdisplay`

```
6259 \@ifpackageloaded{glossaries-accsupp}
6260 {
6261   \newcommand*{\glstr@setaccessdisplay}[1]{%
6262     \ifcsdef{gls#1accessdisplay}%
6263       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
6264       {%
```

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.

```
6265     \edef\@gls@thisval{#1}%
6266     \@for\@gls@map:=\@gls@keymap\do{%
6267       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6268       \ifdefequal{\@this@key}{\@gls@thisval}%
6269       {%
6270         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6271         \@endfortrue
6272       }%
6273     }%
6274   }%
6275   \ifcsdef{gls\@gls@thisval accessdisplay}%
6276   {\letcs\@glstr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```

6277      {\let\@glxtr@accessdisplay\@firstoftwo}%
6278    }%
6279  }
6280 }
6281 {%
6282 \newcommand*{\glxtr@setaccessdisplay}[1]{%
6283   \let\@glxtr@accessdisplay\@firstoftwo}
6284 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6285 \newrobustcmd*{\glossentrynameother}[2]{%
6286   \@glsdofexistsorwarn{#1}%
6287   {%

```

Accessibility support:

```

6288   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6289   \glsssetabbrfmt{\glscategory{#1}}%
6290   \glshasattribute{#1}{glossnamefont}%
6291   {%
6292     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6293     \ifcsdef{\@glxtr@attrval}%
6294     {%
6295       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6296     }%
6297     {%
6298       \GlossariesExtraWarning{Unknown control sequence name
6299         '\@glxtr@attrval' supplied in glossnamefont attribute
6300         for entry '#1'. Reverting to default \string\glsgnamefont}%
6301       \let\@glxtr@glossnamefont\glsgnamefont
6302     }%
6303   }%
6304   {\let\@glxtr@glossnamefont\glsgnamefont}%
6305   \glsgifattribute{#1}{glossname}{firstuc}%
6306   {%
6307     \@glxtr@accessdisplay
6308     {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6309     {#1}%
6310   }%
6311   {%
6312     \glsgifattribute{#1}{glossname}{title}%
6313     {%
6314       \@glxtr@do@titlecaps@warn
6315       \@glxtr@accessdisplay
6316       {\@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{#2}}}%
6317       {#1}%
6318     }%
6319     {%

```

```

6320      \glsifattribute{#1}{glossname}{uc}%
6321      {%
6322      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6323      \@glsxtr@accessdisplay
6324      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6325      {#1}%
6326      }%
6327      {%
6328      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6329      \@glsxtr@accessdisplay
6330      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6331      {#1}%
6332      }%
6333      }%
6334      }%

  Do post-name hook.
6335      \glsxtrpostnamehook{#1}%
6336      }%
6337 }

```

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

```

6338 \newif\if@glsxtr@format@override
6339 \@glsxtr@format@overridefalse

```

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

`xFormatOverride`

```

6340 \@ifpackageloaded{hyperref}
6341 {
  If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
  don't add it.
6342   \ifHy@hyperindex
6343     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6344       \@glsxtr@format@override true
6345       \appto\theindex{\let\glshypernumber\@firstofone}%
6346     }
6347   \else
6348     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6349       \@glsxtr@format@override true
6350       \appto\theindex{\let\glshypernumber\hyperpage}%
6351     }
6352   \fi
6353 }
6354 {
6355   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6356     \@glsxtr@format@override true
6357   }

```

```

6358 }
6359 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6360 \newcommand*{\glstrdoautoindexname}[2]{%
6361   \glshasattribute{#1}{#2}%
6362   {%

```

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.

```

6363   \@glstr@autoindex@setname{#1}%

```

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

```

6364   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6365   \if@glstr@format@override

6366     \ifx\@glstrnumberformat\@glstr@defaultnumberformat
6367     \else
6368       \let\@glstr@attrval\@glstrnumberformat
6369     \fi
6370   \fi
6371   \ifdefstring{\@glstr@attrval}{true}%
6372   {}%
6373   {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6374   \expandafter\glstrautoindex\expandafter{\@glo@name}%
6375   }%
6376   {}%
6377 }

```

glstrautoindex

```

6378 \newcommand*{\glstrautoindex}{\index}

```

xtrautoindexesc

```

6379 \newcommand{\glstrautoindexesc}{%
6380   \@gls@checkmkidxchars\@glo@sort
6381   \@glstr@autoindex@doextra@esc\@glo@sort
6382 }

```

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

```

6383 \newcommand*{\@glstr@autoindex@setname}[1]{%
6384   \protected@edef\@glo@name{\glstrautoindexentry{#1}}%
6385   \glstrautoindexassignsort{\@glo@sort}{#1}%
6386   \glstrautoindexesc
6387   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
6388 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6389 \newcommand*{\glstrautoindexentry}[1]{\string\glstryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```
6390 \newcommand*{\glxtrautoindexassignsort}[2]{%
6391   \glslentryfield{#1}{#2}{sort}%
6392 }
```

dex@doextra@esc

```
6393 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6394   \ifx\@glxtr@autoindex@esc\@gl@quotechar
6395   \else
6396     \def\@gl@checkedmkidx{}%
6397     \edef\@glxtr@checkspch{%
6398       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6399       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6400       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6401     \@glxtr@checkspch
6402     \let#1\@gl@checkedmkidx\relax
6403   \fi
```

Escape actual character unless it has already been escaped.

```
6404   \ifx\@glxtr@autoindex@at\@gl@actualchar
6405   \else
6406     \def\@gl@checkedmkidx{}%
6407     \edef\@glxtr@checkspch{%
6408       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6409       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6410       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6411     \@glxtr@checkspch
6412     \let#1\@gl@checkedmkidx\relax
6413   \fi
```

Escape level character unless it has already been escaped.

```
6414   \ifx\@glxtr@autoindex@level\@gl@levelchar
6415   \else
6416     \def\@gl@checkedmkidx{}%
6417     \edef\@glxtr@checkspch{%
6418       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6419       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6420       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6421     \@glxtr@checkspch
6422     \let#1\@gl@checkedmkidx\relax
6423   \fi
```

Escape encap character unless it has already been escaped.

```
6424   \ifx\@glxtr@autoindex@encap\@gl@encapchar
6425   \else
6426     \def\@gl@checkedmkidx{}%
6427     \edef\@glxtr@checkspch{%
6428       \noexpand\@glxtr@autoindex@escencap\expandonce{#1}%
6429       \noexpand\@empty\@glxtr@autoindex@encap\noexpand\@nnil
```

```

6430      \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6431      \@glsxtr@checkspch
6432      \let#1\@gls@checkedmkidx\relax
6433      \fi
6434 }

```

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`.

```
6435 \newcommand*{\@glsxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```

6436 \newcommand*{\GlsXtrSetActualChar}[1]{%
6437   \gdef\@glsxtr@autoindex@at{#1}%
6438   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6439     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6440   }%
6441 }
6442 \@onlypreamble\GlsXtrSetActualChar
6443 \makeatother
6444 \GlsXtrSetActualChar{@}
6445 \makeatletter

```

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

```
6446 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```

6447 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6448   \gdef\@glsxtr@autoindex@encap{#1}%
6449   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
6450     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6451   }%
6452 }
6453 \GlsXtrSetEncapChar{|}
6454 \@onlypreamble\GlsXtrSetEncapChar

```

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

```
6455 \newcommand*{\@glsxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```

6456 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6457   \gdef\@glsxtr@autoindex@level{#1}%
6458   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
6459     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
6460   }%
6461 }
6462 \GlsXtrSetLevelChar{!}
6463 \@onlypreamble\GlsXtrSetLevelChar

```

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

```
6464 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
6465 \newcommand*{\GlsXtrSetEscChar}[1]{%
6466   \gdef\@glsxtr@autoindex@esc{#1}%
6467   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6468     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6469   }%
6470 }
6471 \GlsXtrSetEscChar{"}
6472 \@onlypreamble\GlsXtrSetEscChar
```

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

```
6473 \ifdef\actualchar
6474 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6475 {}
```

Quote character \quotechar:

```
6476 \ifdef\quotechar
6477 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6478 {}
```

Level character \levelchar:

```
6479 \ifdef\levelchar
6480 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6481 {}
```

Encap character \encapchar:

```
6482 \ifdef\encapchar
6483 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6484 {}
```

leto@endescspch

```
6485 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

```
\@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
6486 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
6487   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
6488   \toks@={#3}%
6489   \ifx\@nnil#3\relax
6490     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
6491   \else
6492     \ifx\@nnil#4\relax
6493       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
6494       \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```



```

6495      #4#5\@glxtr@endescspch}%
6496  \else
6497      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
6498      \@glxtr@autoindex@esc#1}%
6499      \def\@@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6500  \fi
6501 \fi
6502 \@glxtr@checkspch
6503 }

```

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

```

6504 \renewcommand*{\Glossentrydesc}[1]{%
6505   \glsdoifexistsorwarn{#1}%
6506   {%
6507     \glssetabbrvfmt{\glscategory{#1}}%
6508     \Glsaccessdesc{#1}%
6509   }%
6510 }

```

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

```

6511 \renewcommand*{\Glossentrysymbol}[1]{%
6512   \glsdoifexistsorwarn{#1}%
6513   {%
6514     \glssetabbrvfmt{\glscategory{#1}}%
6515     \Glsaccesssymbol{#1}%
6516   }%
6517 }

```

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

```

6518 \renewcommand*{\Glossentrysymbol}[1]{%
6519   \glsdoifexistsorwarn{#1}%
6520   {%
6521     \glssetabbrvfmt{\glscategory{#1}}%
6522     \Glsaccesssymbol{#1}%
6523   }%
6524 }

```

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

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

```

6525 \newcommand*{\GlsXtrEnableInitialTagging}{%
6526   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6527 }
6528 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\GlsXtrEnableInitialTagging` Starred version undefines command.

```

6529 \newcommand*{\s@glstr@enabletagging}[2]{%
6530   \undef#2%
6531   \@glstr@enabletagging{#1}{#2}%
6532 }

```

`r@enabletagging` Internal command.

```

6533 \newcommand*{\@glstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6534   \@for\@glstr@cat:=#1\do
6535   {%
6536     \ifdefempty\@glstr@cat
6537     {}%
6538     {\glsssetcategoryattribute{\@glstr@cat}{tagging}{true}}%
6539   }%
6540   \newrobustcmd*#2[1]{##1}%
6541   \def\@glstr@taggingcs{#2}%
6542   \renewcommand*{\@glstr@activate@initialtagging{%
6543     \let#2\@glstr@tag
6544   }%
6545   \ifundef\@glstr@preglossaryhook
6546   {\GlossariesExtraWarning{Initial tagging requires at least
6547     glossaries.sty v4.19 to work correctly}}%
6548   {}%
6549 }

```

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.

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

```

6550 \ifundef\mfu@checkword@do
6551 {
6552   \newcommand*{\mfu@checkword@do}[1]{%
6553     \ifdefstring{\mfu@checkword@arg}{#1}%
6554     {%
6555       \let\@mfu@do@makefirstuc\@firstofone
6556       \listbreak
6557     }%
6558     {}%
6559   }

```

`\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.

```

6560 \ifundef\mfu@checkword
6561 {
6562   \newcommand{\@glstr@do@titlecaps@warn}{%
6563     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6564       support not available}%

```

One warning should suffice.

```
6565 \let\@glxtr@do@titlecaps@warn\relax
6566 }
6567 }
6568 {
6569 \renewcommand*{\mfu@checkword}[1]{%
6570 \def\mfu@checkword@arg{#1}%
6571 \let\@mfu@domakefirstuc\makefirstuc
6572 \forlistloop\mfu@checkword@do\@mfu@nocaplist
6573 }
6574 }
6575 }
6576 {}% no patch required
```

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

```
6577 \newcommand*{\@glxtr@do@titlecaps@warn}{}
```

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

```
6578 \newcommand*\@glxtr@activate@initialtagging{}
```

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

```
6579 \newrobustcmd*{\@glxtr@tag}[1]{%
6580 \gl@ifattribute{\gl@currententrylabel}{tagging}{true}%
6581 {\gl@xtr@tagfont{#1}}{#1}%
6582 }
```

\glxtr@tagfont Used in the glossary.

```
6583 \newcommand*\glxtr@tagfont[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.

```
6584 \ifdef\@gls@preglossaryhook
6585 {
6586 \renewcommand*{\@gls@preglossaryhook}{%
6587 \@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.

```
6588 \ifundef\@glxtr@org@postdescription
6589 {%
6590 \let\@glxtr@org@postdescription\gl@postdescription
6591 \renewcommand*{\gl@postdescription}{%
6592 \ifgl@entryexists{\gl@currententrylabel}%
6593 {%
6594 \glxtr@postdescription
6595 \@glxtr@org@postdescription
```

```

6596         }%
6597         {}%
6598     }%
6599     }%
6600     {}%

    Enable the options used by \@glsxtrp:
6601     \glossxtrsetpopts
6602 }%
6603 }
6604 {}

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.
6605 \newcommand*{\glsxtrpostdescription}{%
6606     \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6607 }

postdescgeneral
6608 \newcommand*{\glsxtrpostdescgeneral}{}

xtrpostdescterm
6609 \newcommand*{\glsxtrpostdescterm}{}

postdescacronym
6610 \newcommand*{\glsxtrpostdescacronym}{}

descabbreviation
6611 \newcommand*{\glsxtrpostdescabbreviation}{}

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.
6612 \newcommand*{\glsdefpostdesc}[2]{%
6613     \csdef{glsxtrpostdesc#1}{#2}%
6614 }

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.
6615 \renewcommand*{\glspostlinkhook}{%
6616     \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6617 }

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.
6618 \newcommand*{\glsxtrpostlinkhook}{%
6619     \glsxtrdiscardperiod{\glslabel}%

```

```

6620 {\glxtrpostlinkendsentence}%
6621 {\glxtrifcustomdiscardperiod
6622  {\glxtrifperiod{\glxtrpostlinkendsentence}{\glxtrpostlink}}}%
6623  {\glxtrpostlink}%
6624 }%
6625 }

```

`\omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```

6626 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}

```

`\glxtrpostlink`

```

6627 \newcommand*{\glxtrpostlink}{%
6628  \csuse{\glxtrpostlink\glscategory{\glslabel}}}%
6629 }

```

`\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`.

```

6630 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6631  \ifthenelse{\equal{#1}{}}{%
6632    {\PackageError{glossaries-extra}
6633     {Invalid empty category label in \string\glxtrpostlink}{}}}%
6634    {\csdef{\glxtrpostlink#1}{#2}}}%
6635 }

```

`\linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```

6636 \newcommand*{\glxtrpostlinkendsentence}{%
6637  \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
6638  {%
6639    \csuse{\glxtrpostlink\glscategory{\glslabel}}}%
  Put the full stop back.
6640  .\spacefactor\sfcode'\. \relax
6641 }%
6642 {%

```

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

```

6643  \spacefactor\sfcode'\. \relax
6644 }%
6645 }

```

`\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.

```

6646 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6647  \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}}%
6648 }

```

`\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.

```

6649 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6650   \glxtrifwasfirstuse
6651   {%
6652     \ifglshassymbol{\glslabel}%
6653     {\space\glxtrparen{\glsaccesssymbol{\glslabel}}}%
6654     {}%
6655   }%
6656   {}%
6657 }

```

`\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.

```

6658 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6659   \glxtrifwasfirstuse
6660   {%
6661     \space\glxtrparen
6662     {%
6663       \ifglshassymbol{\glslabel}%
6664       {\glsaccesssymbol{\glslabel}, }%
6665       {}%
6666       \glsaccessdesc{\glslabel}%
6667     }%
6668   }%
6669   {}%
6670 }

```

`\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).

```

6671 \newcommand*{\glxtrdiscardperiod}[3]{%
6672   \glxtrifwasfirstuse
6673   {%
6674     \glsifattribute{#1}{retainfirstuseperiod}{true}%
6675     {#3}%
6676     {%
6677       \glsifattribute{#1}{discardperiod}{true}%
6678       {%
6679         \glsifplural
6680         {%
6681           \glsifattribute{#1}{pluraldiscardperiod}{true}%
6682           {\glxtrifperiod{#2}{#3}}%
6683           {#3}%
6684         }%
6685         {%
6686           \glxtrifperiod{#2}{#3}%

```

```

6687     }%
6688     }%
6689     {#3}%
6690     }%
6691 }%
6692 {%
6693   \glsifattribute{#1}{discardperiod}{true}%
6694   {%
6695     \glsifplural
6696     {%
6697       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6698       {\glsxtrifperiod{#2}{#3}}%
6699       {#3}%
6700     }%
6701     {%
6702       \glsxtrifperiod{#2}{#3}%
6703     }%
6704   }%
6705   {#3}%
6706 }%
6707 }

```

`\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`

```
6708 \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 ' ').

```
6709 \newcommand*{\glsxtr@punclist}{.,:;?!}
```

`punctuationmark` Add character to punctuation list.

```
6710 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`punctuationmarks` Reset the punctuation list.

```
6711 \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`.)

```

6712 \newcommand*{\glsxtrifnextpunc}[2]{%
6713   \def\reserved@a{#1}%
6714   \def\reserved@b{#2}%
6715   \futurelet\@glspunc@token\glsxtr@ifnextpunc
6716 }

```

sxtr@ifnextpunc

```
6717 \newcommand*{\glsxtr@ifnextpunc}{%
6718   \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6719   \reserved@b
6720 }
```

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

```
6721 \newcommand*{\glsxtr@ifpunctoken}[1]{%
6722   \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
6723 }
```

xtr@ifpunctoken

```
6724 \def\@glsxtr@ifpunctoken#1#2{%
6725   \let\reserved@d=#2%
6726   \ifx\reserved@d\@nnil
6727     \let\glsxtr@next\@glsxtr@notfoundinlist
6728   \else
6729     \ifx#1\reserved@d
6730       \let\glsxtr@next\@glsxtr@foundinlist
6731     \else
6732       \let\glsxtr@next\@glsxtr@ifpunctoken
6733     \fi
6734   \fi
6735   \glsxtr@next#1%
6736 }
```

xtr@foundinlist

```
6737 \def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6738 \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.

```
6739 \newcommand{\glsxtrdopostpunc}[1]{%
6740   \glsxtr@ifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
6741 }
```

@glsxtr@swaptwo

```
6742 \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.

```
6743 \define@key{glstrabbrv}{category}{%
6744   \edef\glscategorylabel{#1}%
6745   \ifcsdef{@glssabbrv@current@#1}%
6746   {%
```

Warning should already have been issued.

```
6747   \let\@glstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6748   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6749   \glstr@applyabbrvstyle{\csname @glssabbrv@current@#1\endcsname}%
6750   \let\GlsXtrWarnDeprecatedAbbrStyle\@glstr@orgwarndep
6751 }%
6752 }%
6753 }
```

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

```
6754 \define@key{glstrabbrv}{shortplural}{%
6755   \def\@glss@shortpl{#1}%
6756 }
```

Similarly for the long plural form.

```
6757 \define@key{glstrabbrv}{longplural}{%
6758   \def\@glss@longpl{#1}%
6759 }
```

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

`\glssshortpltok`

```
6760 \newtoks\glssshortpltok
```

`\glslongpltok`

```
6761 \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.

```
6762 \newcommand*{\@glstr@insertdots}[2]{%
6763   \def#1{%
6764     \@glstr@insert@dots#1#2\@nnil
6765   }
```

xtr@insert@dots

```
6766 \newcommand*{\@glsxtr@insert@dots}[2]{%
6767   \ifx\@nnil#2\relax
6768   \let\@glsxtr@insert@dots@next\@gobble
6769   \else
6770   \ifx\relax#2\relax
6771   \else
6772     \appto#1{#2.}%
6773   \fi
6774   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6775   \fi
6776   \@glsxtr@insert@dots@next#1%
6777 }
```

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be defined:

\glsxtrwordsep

```
6778 \newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

\glsxtrword

```
6779 \newcommand*{\glsxtrword}[1]{#1}
```

tr@markwordseps

```
6780 \newcommand*{\@glsxtr@markwordseps}[2]{%
6781   \def#1{}%
6782   \@glsxtr@mark@wordseps#1#2 \@nnil
6783 }
```

r@mark@wordseps

```
6784 \def\@glsxtr@mark@wordseps#1#2 #3{%
6785   \ifdefempty{#1}%
6786   {\def#1{\protect\glsxtrword{#2}}}%
6787   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
6788   \ifx\@nnil#3\relax
6789   \let\@glsxtr@mark@wordseps@next\relax
6790   \else
6791     \def\@glsxtr@mark@wordseps@next{%
6792       \@glsxtr@mark@wordseps#1#3}%
6793   \fi
6794   \@glsxtr@mark@wordseps@next
6795 }
```

newabbreviation Define a new generic abbreviation.

```
6796 \newcommand*{\newabbreviation}[4][ ]{%
6797   \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
6798 }
```

`\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.)

```
6799 \newcommand*{\glxtr@newabbreviation}[4]{%
6800   \glstoklisttok{#1}%
6801   \glstoklabeltok{#2}%
6802   \glstokshorttok{#3}%
6803   \glstoklongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
6804   \def\glxtrorgshort{#3}%
6805   \def\glxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
6806   \def\ExtraCustomAbbreviationFields{}
```

Initialise accessibility settings if required.

```
6807   \@gls@initaccesskeys
```

Get the category.

```
6808   \def\glscategorylabel{abbreviation}%
6809   \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
6810   \setkeys*{\glxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
6811   \def\@gls@longpl{#4\glspluralsuffix}%
6812   \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
6813   \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6814   {%
6815     \@glsxtr@markwordseps\@gls@long{#4}%
6816     \expandafter\def\expandafter\@gls@longpl\expandafter
6817       {\@gls@long\glspluralsuffix}%
6818     \let\@gls@default@longpl\@gls@longpl
```

Update `\glslongtok`.

```
6819     \expandafter\glslongtok\expandafter{\@gls@long}%
6820   }%
6821   {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6822   \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6823   {%
6824     \@glsxtr@markwordseps\@gls@short{#3}%
6825   }%
6826   {}%
```

Has the insertdots attribute been set?

```
6827   \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6828   {%
6829     \@glsxtr@insertdots\@gls@short{#3}%
```

```

6830      \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6831    }%
6832    {\def\@gls@short{#3}}%
6833  }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

6834  \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6835  {%
6836    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6837      '\abbrvpluralsuffix}%
6838  }%
6839  {%

```

Has the noshortplural attribute been set?

```

6840  \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6841  {%
6842    \let\@gls@shortpl\@gls@short
6843  }%
6844  {%
6845    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6846      '\abbrvpluralsuffix}%
6847  }%
6848  }%

```

Update \glsshorttok:

```

6849  \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

6850  \glxtrnewabbrevpresetkeyhook{#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).

```

6851  \setkeys*{\glxtrabbrv}[category]{#1}%

```

Has the plural been explicitly set?

```

6852  \ifx\@gls@default@longpl\@gls@longpl
6853  \else

```

Has the markwords attribute been set?

```

6854  \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6855  {%
6856    \expandafter\@glxtr@markwordseps\expandafter\@gls@longpl\expandafter
6857    {\@gls@longpl}%
6858  }%
6859  {}%
6860  \fi

```

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

```

6861  \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6862  \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```

6863  \@gls@setup@default@short@access{#3}%

```

Do any extra setup provided by hook:

6864 \newabbreviationhook

Define this entry:

```
6865 \protected@edef\@do@newglossaryentry{%
6866   \noexpand\newglossaryentry{\the\glslabeltok}%
6867   {%
6868     type=\glstrabbrvtype,%
6869     category=abbreviation,%
6870     short={\the\glsshorttok},%
6871     shortplural={\the\glsshortpltok},%
6872     long={\the\glslongtok},%
6873     longplural={\the\glslongpltok},%
6874     name={\the\glsshorttok},%
6875     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6876   \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6877   \the\glskeylisttok
6878 }%
6879 }%
6880 \do@newglossaryentry
6881 \GlsXtrPostNewAbbreviation
6882 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

```
6883 \newcommand*{\glstrnewabbrevpresetkeyhook}[3]{}
```

NewAbbreviation Hook used by abbreviation styles.

```
6884 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

bbreviationhook Hook for use with \newabbreviation.

```
6885 \newcommand*{\newabbreviationhook}{}%
```

reviationFields

```
6886 \newcommand*{\CustomAbbreviationFields}{}%
```

\glstrparen For the parenthetical styles.

```
6887 \newcommand*{\glstrparen}[1]{(#1)}
```

lsxtrfullformat Full format without case change.

```
6888 \newcommand*{\glstrfullformat}[2]{%
6889   \glsfirstlongfont{\glsaccesslong{#1}}#2\glstrfullsep{#1}%
6890   \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6891 }
```

`\lsxtrfullformat` Full format with case change.

```
6892 \newcommand*{\Glsxtrfullformat}[2]{%
6893   \glsfirstlongfont{\Glsaccesslong{#1}}#2\gsxtrfullsep{#1}%
6894   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
6895 }
```

`\xtrfullplformat` Plural full format without case change.

```
6896 \newcommand*{\Glsxtrfullplformat}[2]{%
6897   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6898   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6899 }
```

`\xtrfullplformat` Plural full format with case change.

```
6900 \newcommand*{\Glsxtrfullplformat}[2]{%
6901   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6902   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6903 }
```

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

```
6904 \newcommand*{\glxtrfullsep}[1]{\space}
```

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

`\nlinefullformat` Full format without case change.

```
6905 \newcommand*{\Glsxtrinlinefullformat}{\glxtrfullformat}
```

`\nlinefullformat` Full format with case change.

```
6906 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\xtrfullplformat` Plural full format without case change.

```
6907 \newcommand*{\Glsxtrinlinefullplformat}{\glxtrfullplformat}
```

`\inefullplformat` Plural full format with case change.

```
6908 \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 `\glxtrfull` set of commands instead.

`\glentryfull`

```
6909 \renewcommand*{\glentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
6910 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glentryfullpl`

```
6911 \renewcommand*{\glentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

6912 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

6913 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

6914 `\newcommand*{\glsabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.

6915 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`

6916 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.

6917 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.

6918 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.

6919 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`

6920 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

6921 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glsbrvpluralsuffix` Default plural suffix.

6922 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).

6923 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`

6924 `\newcommand*{\ns@glsxtrfull}[2][{}]{%`

6925 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%`

6926 `{\@glsxtr@full{#1}{#2}}[{}]{%`

6927 `}`

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

6928 `\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).

```

6929 \@glxtr@record{#1}{#2}{glslink}%
6930 \glsdoifexists{#2}%
6931 {%
6932   \glsetabbrfmt{\glscategory{#2}}%
6933   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6934   \let\gl@ifplural\@secondoftwo
6935   \let\gl@scapscase\@firstofthree
6936   \let\gl@insert\@empty
6937   \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.

```

6938   \glxtrsetupfulldefs
6939   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6940 }%
6941 \glspostlinkhook
6942 }

```

trsetupfulldefs

```

6943 \newcommand*{\glxtrsetupfulldefs}{%
6944   \let\glxtrifwasfirstuse\@firstoftwo
6945 }

```

\Glsxtrfull Full form (first letter uppercase).

```

6946 \newrobustcmd*{\Glsxtrfull}{\@gl@hyp@opt\@ns@Glsxtrfull}
6947 \newcommand*\ns@Glsxtrfull[2][{}]{%
6948   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
6949     {\@Glsxtr@full{#1}{#2}[]}%
6950 }

```

\@Glsxtr@full Low-level macro:

```

6951 \def\@Glsxtr@full#1#2[#3]{%
6952   \glsdoifexists{#2}%
6953   {%
6954     \glsetabbrfmt{\glscategory{#2}}%
6955     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6956     \let\gl@ifplural\@secondoftwo
6957     \let\gl@scapscase\@secondofthree
6958     \let\gl@insert\@empty
6959     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6960     \glxtrsetupfulldefs
6961     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6962   }%
6963   \glspostlinkhook
6964 }

```


`\GLSxtrfull` Full form (all uppercase).

```
6965 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\@ns@GLSxtrfull}
6966 \newcommand*\ns@GLSxtrfull[2][\%
6967   \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{\%
6968     {\@GLSxtr@full{#1}{#2}[]}\%
6969 }
```

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

```
6970 \def\@GLSxtr@full#1#2[#3]{\%
6971   \glsdoifexists{#2}\%
6972   {\%
6973     \glssetabbrvfmt{\glscategory{#2}}\%
6974     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6975     \let\glsifplural\@secondoftwo
6976     \let\glsupcase\@thirdofthree
6977     \let\glsinsert\@empty
6978     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}\%
6979     \glsxtrsetupfulldefs
6980     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}\%
6981   }\%
6982   \glspostlinkhook
6983 }
```

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

```
6984 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\@ns@glsxtrfullpl}
6985 \newcommand*\ns@glsxtrfullpl[2][\%
6986   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{\%
6987     {\@glsxtr@fullpl{#1}{#2}[]}\%
6988 }
```

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

```
6989 \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).
6990   \@glsxtr@record{#1}{#2}{\glslink}\%
6991   \glsdoifexists{#2}\%
6992   {\%
6993     \glssetabbrvfmt{\glscategory{#2}}\%
6994     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6995     \let\glsifplural\@firstoftwo
6996     \let\glsupcase\@firstofthree
6997     \let\glsinsert\@empty
6998     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}\%
6999     \glsxtrsetupfulldefs
7000     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}\%
7001   }\%
7002   \glspostlinkhook
7003 }
```

```

\Glsxtrfullpl Plural full form (first letter uppercase).
7004 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
7005 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
7006   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}]{%
7007     {\@Glsxtr@fullpl{#1}{#2}[]}%
7008 }

\@Glsxtr@fullpl Low-level macro:
7009 \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).
7010   \@glsxtr@record{#1}{#2}{glslink}%
7011   \glsdoifexists{#2}%
7012   {%
7013     \glssetabbrvfmt{\gls@category{#2}}%
7014     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7015     \let\glsifplural\@firstoftwo
7016     \let\gls@scapscase\@secondofthree
7017     \let\glsinsert\@empty
7018     \def\gls@customtext{\Glsxtrinlinefullplformat{#2}{#3}}%
7019     \glsxtrsetupfulldefs
7020     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7021   }%
7022   \gls@postlinkhook
7023 }

\GLSxtrfullpl Plural full form (all upper case).
7024 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
7025 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
7026   \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}]{%
7027     {\@GLSxtr@fullpl{#1}{#2}[]}%
7028 }

\@GLSxtr@fullpl Low-level macro:
7029 \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).
7030   \@glsxtr@record{#1}{#2}{glslink}%
7031   \glsdoifexists{#2}%
7032   {%
7033     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7034     \let\glsifplural\@firstoftwo
7035     \let\gls@scapscase\@thirdofthree
7036     \let\glsinsert\@empty
7037     \def\gls@customtext{%
7038       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
7039     \glsxtrsetupfulldefs

```

```

7040 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7041 }%
7042 \glspostlinkhook
7043 }

```

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

`\glsxtrshort`

```

7044 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

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

```

7045 \newcommand*{\ns@glsxtrshort}[2] [] {%
7046 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2} []}%
7047 }

```

Read in the final optional argument:

```

7048 \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).

```

7049 \glsxtr@record{#1}{#2}{glslink}%
7050 \glsdoifexists{#2}%
7051 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

7052 \glssetabbrvfmt{\glscategory{#2}}%
7053 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7054 \let\glsxtrifwasfirstuse\@secondoftwo
7055 \let\glsifplural\@secondoftwo
7056 \let\glscapscase\@firstofthree
7057 \let\glsinsert\@empty
7058 \def\glscustomtext{%
7059 \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7060 \ifglsxtrininsertinside\else#3\fi
7061 }%
7062 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7063 }%
7064 \glspostlinkhook
7065 }

```

`\Glsxtrshort`

```

7066 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

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

```

7067 \newcommand*{\ns@Glsxtrshort}[2] [] {%
7068 \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
7069 }

```

Read in the final optional argument:

```

7070 \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).

```

7071 \@glsxtr@record{#1}{#2}{glslink}%
7072 \glsdoifexists{#2}%
7073 {%
7074 \glssetabbrvfmt{\glscategory{#2}}%
7075 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7076 \let\glsxtrifwasfirstuse\@secondoftwo
7077 \let\glsifplural\@secondoftwo
7078 \let\glsapscase\@secondofthree
7079 \let\glsinsert\@empty
7080 \def\glscustomtext{%
7081 \glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7082 \ifglsxtrininsertinside\else#3\fi
7083 }%
7084 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7085 }%
7086 \glspostlinkhook
7087 }

```

\GLSxtrshort

```

7088 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}

```

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

```

7089 \newcommand*{\ns@GLSxtrshort}[2] [] {%
7090 \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} []}]%
7091 }

```

Read in the final optional argument:

```

7092 \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).

```

7093 \@glsxtr@record{#1}{#2}{glslink}%
7094 \glsdoifexists{#2}%
7095 {%
7096 \glssetabbrvfmt{\glscategory{#2}}%
7097 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7098 \let\glsxtrifwasfirstuse\@secondoftwo
7099 \let\glsifplural\@secondoftwo
7100 \let\glsapscase\@thirdofthree
7101 \let\glsinsert\@empty
7102 \def\glscustomtext{%
7103 \mfirstucMakeUppercase
7104 {\glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7105 \ifglsxtrininsertinside\else#3\fi
7106 }%
7107 }%
7108 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7109 }%

```

```

7110 \glspostlinkhook
7111 }

```

\glsxtrlong

```

7112 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7113 \newcommand*{\ns@glxtrlong}[2] [] {%
7114   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
7115 }

```

Read in the final optional argument:

```

7116 \def\@glxtrlong#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).
7117   \@glxtr@record{#1}{#2}{glslink}%
7118   \glstoifexists{#2}%
7119   {%
7120     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7121     \let\glxtrifwasfirstuse\@secondoftwo
7122     \let\glxtrifplural\@secondoftwo
7123     \let\glscapscase\@firstofthree
7124     \let\glxtrinsert\@empty
7125     \def\glscustomtext{%
7126       \glslongfont{\glssaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7127       \ifglxtrinsertinside\else#3\fi
7128     }%
7129     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7130   }%
7131   \glspostlinkhook
7132 }

```

\Glsxtrlong

```

7133 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7134 \newcommand*{\ns@Glsxtrlong}[2] [] {%
7135   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
7136 }

```

Read in the final optional argument:

```

7137 \def\@Glsxtrlong#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).
7138   \@glxtr@record{#1}{#2}{glslink}%
7139   \glstoifexists{#2}%
7140   {%
7141     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7142     \let\glxtrifwasfirstuse\@secondoftwo

```

```

7143 \let\glsifplural\@secondoftwo
7144 \let\glsifscapscase\@secondofthree
7145 \let\glsinsert\@empty
7146 \def\glscustomtext{%
7147   \glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7148   \ifglstrinsertinside\else#3\fi
7149 }%
7150 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7151 }%
7152 \glspostlinkhook
7153 }

```

\GLSxtrlong

```

7154 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}

```

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

```

7155 \newcommand*{\ns@GLSxtrlong}[2] [] {%
7156   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
7157 }

```

Read in the final optional argument:

```

7158 \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).

```

7159   \@glsxtr@record{#1}{#2}{glslink}%
7160   \glsdoifexists{#2}%
7161   {%
7162     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7163     \let\glsxtrifwasfirstuse\@secondoftwo
7164     \let\glsifplural\@secondoftwo
7165     \let\glsifscapscase\@thirdofthree
7166     \let\glsinsert\@empty
7167     \def\glscustomtext{%
7168       \mfirstucMakeUppercase
7169       {\glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7170       \ifglstrinsertinside\else#3\fi
7171     }%
7172     }%
7173     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7174   }%
7175   \glspostlinkhook
7176 }

```

Plural short forms:

\glsxtrshortpl

```

7177 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}

```

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

```

7178 \newcommand*{\ns@glsxtrshortpl}[2] [] {%

```

```

7179 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7180 }

```

Read in the final optional argument:

```

7181 \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).

```

7182 \@glsxtr@record{#1}{#2}{glslink}%
7183 \glsdoifexists{#2}%
7184 {%
7185 \glssetabbrvfmt{\glscategory{#2}}%
7186 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7187 \let\glsxtrifwasfirstuse\@secondoftwo
7188 \let\glsifplural\@firstoftwo
7189 \let\glsapscase\@firstofthree
7190 \let\glsinsert\@empty
7191 \def\glscustomtext{%
7192 \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7193 \ifglsxtrininsertinside\else#3\fi
7194 }%
7195 \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7196 }%
7197 \glspostlinkhook
7198 }

```

\Glsxtrshortpl

```

7199 \newrobustcmd*{\Glsxtrshortpl}{\@glsl@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

7200 \newcommand*{\@ns@Glsxtrshortpl}[2] []{%
7201 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7202 }

```

Read in the final optional argument:

```

7203 \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).

```

7204 \@glsxtr@record{#1}{#2}{glslink}%
7205 \glsdoifexists{#2}%
7206 {%
7207 \glssetabbrvfmt{\glscategory{#2}}%
7208 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7209 \let\glsxtrifwasfirstuse\@secondoftwo
7210 \let\glsifplural\@firstoftwo
7211 \let\glsapscase\@secondofthree
7212 \let\glsinsert\@empty
7213 \def\glscustomtext{%
7214 \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7215 \ifglsxtrininsertinside\else#3\fi

```

```

7216 }%
7217 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7218 }%
7219 \glspostlinkhook
7220 }

```

\GLSxtrshortpl

```

7221 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7222 \newcommand*{\ns@GLSxtrshortpl}[2][{}]{%
7223   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
7224 }

```

Read in the final optional argument:

```

7225 \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).
7226 \@glsxtr@record{#1}{#2}{glslink}%
7227 \glsdoifexists{#2}%
7228 {%
7229   \glssetabbrvfmt{\glscategory{#2}}%
7230   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7231   \let\glsxtrifwasfirstuse\@secondoftwo
7232   \let\glsifplural\@firstoftwo
7233   \let\glscapscase\@thirdofthree
7234   \let\glsinsert\@empty
7235   \def\glscustomtext{%
7236     \mfirstucMakeUppercase
7237     {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7238     \ifglsxtrininsertinside\else#3\fi
7239   }%
7240 }%
7241 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7242 }%
7243 \glspostlinkhook
7244 }

```

Plural long forms:

\glsxtrlongpl

```

7245 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7246 \newcommand*{\ns@glsxtrlongpl}[2][{}]{%
7247   \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
7248 }

```

Read in the final optional argument:

```

7249 \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).

```

7250 \@glstr@record{#1}{#2}{glslink}%
7251 \glsoifexists{#2}%
7252 {%
7253 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7254 \let\glstrifwasfirstuse\@secondoftwo
7255 \let\gl@ifplural\@firstoftwo
7256 \let\glscapscase\@firstofthree
7257 \let\glinsert\@empty
7258 \def\glscustomtext{%
7259 \glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7260 \ifglstrinsertinside\else#3\fi
7261 }%
7262 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7263 }%
7264 \glspostlinkhook
7265 }

```

\Glsxtrlongpl

```

7266 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}

```

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

```

7267 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7268 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7269 }

```

Read in the final optional argument:

```

7270 \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).

```

7271 \@glstr@record{#1}{#2}{glslink}%
7272 \glsoifexists{#2}%
7273 {%
7274 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7275 \let\glstrifwasfirstuse\@secondoftwo
7276 \let\gl@ifplural\@firstoftwo
7277 \let\glscapscase\@secondofthree
7278 \let\glinsert\@empty
7279 \def\glscustomtext{%
7280 \glslongfont{\Glsaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7281 \ifglstrinsertinside\else#3\fi
7282 }%
7283 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7284 }%
7285 \glspostlinkhook
7286 }

```

\GLSxtrlongpl

```

7287 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7288 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7289   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7290 }

    Read in the final optional argument:
7291 \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).
7292   \@glsxtr@record{#1}{#2}{glslink}%
7293   \glsdoifexists{#2}%
7294   {%
7295     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7296     \let\glsxtrifwasfirstuse\@secondoftwo
7297     \let\glsifplural\@firstoftwo
7298     \let\glsapscase\@thirdofthree
7299     \let\glsinsert\@empty
7300     \def\glscustomtext{%
7301       \mfirstucMakeUppercase
7302       {\glsfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7303       \ifglsxtrininsertinside\else#3\fi
7304     }%
7305     }%
7306     \@gls@link{#1}{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7307   }%
7308   \glspostlinkhook
7309 }

```

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

```

7310 \newcommand*{\glssetabbrvfmt}[1]{%
7311   \ifcsdef{\glsabbrv@current@#1}%
7312   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7313   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7314 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7315 \newrobustcmd*{\glsuseabbrvfont}[2]{{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```

7316 \newrobustcmd*{\glsuselongfont}[2]{{\glssetabbrvfmt{#2}\glsfont{#1}}}

```

sxtrgenabbrvfmt Similar to \glsacfm, but for abbreviations.

```

7317 \newcommand*{\glsxtrgenabbrvfmt}{%
7318   \ifdefempty\glscustomtext
7319   {%
7320     \ifglsused\glslabel
7321     {%

```

Subsequent use:

7322 \glsifplural
7323 {%

Subsequent plural form:

7324 \glscapscase
7325 {%

Subsequent plural form, don't adjust case:

7326 \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7327 }%
7328 {%

Subsequent plural form, make first letter upper case:

7329 \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7330 }%
7331 {%

Subsequent plural form, all caps:

7332 \mfirstucMakeUppercase
7333 {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7334 }%
7335 }%
7336 {%

Subsequent singular form

7337 \glscapscase
7338 {%

Subsequent singular form, don't adjust case:

7339 \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7340 }%
7341 {%

Subsequent singular form, make first letter upper case:

7342 \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7343 }%
7344 {%

Subsequent singular form, all caps:

7345 \mfirstucMakeUppercase
7346 {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7347 }%
7348 }%
7349 }%
7350 {%

First use:

7351 \glsifplural
7352 {%

First use plural form:

7353 \glscapscase
7354 {%

First use plural form, don't adjust case:

```
7355      \glstrfullplformat{\glslabel}{\glsinsert}%
7356      }%
7357      {%
```

First use plural form, make first letter upper case:

```
7358      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
7359      }%
7360      {%
```

First use plural form, all caps:

```
7361      \mfirstucMakeUppercase
7362      {\glstrfullplformat{\glslabel}{\glsinsert}}%
7363      }%
7364      }%
7365      {%
```

First use singular form

```
7366      \glscapscase
7367      {%
```

First use singular form, don't adjust case:

```
7368      \glstrfullformat{\glslabel}{\glsinsert}%
7369      }%
7370      {%
```

First use singular form, make first letter upper case:

```
7371      \Glsxtrfullformat{\glslabel}{\glsinsert}%
7372      }%
7373      {%
```

First use singular form, all caps:

```
7374      \mfirstucMakeUppercase
7375      {\glstrfullformat{\glslabel}{\glsinsert}}%
7376      }%
7377      }%
7378      }%
7379      }%
7380      {%
```

User supplied text.

```
7381      \glscustomtext
7382      }%
7383 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7384 \newcommand*{\glxtrsubsequentfmt}[2]{%
7385   \glsabbrvfont{\glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7386   \ifglxtrininsertinside \else#2\fi
7387 }
7388 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

```

subsequentplfmt Subsequent use format (plural no case change).
7389 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7390   \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7391   \ifglxtrinsertinside \else#2\fi
7392 }
7393 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
7394 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7395   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7396   \ifglxtrinsertinside \else#2\fi
7397 }
7398 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).
7399 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7400   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7401   \ifglxtrinsertinside \else#2\fi
7402 }
7403 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

```

breviationstyle
7404 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7405   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
7406   {%
7407     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7408   }%
7409   {%
7410     Have abbreviations already been defined for this category?
7411     \ifcsstring{@glsabbrv@current@#1}{#2}%
7412     {%
7413       Style already set.
7414       }%
7415       \def\@glxtr@dostylewarn{}%
7416       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7417       {%
7418         \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7419           style has been switched \MessageBreak
7420           for category ‘#1’, \MessageBreak
7421           but there have already been entries \MessageBreak
7422           defined for this category. Unwanted \MessageBreak
7423           side-effects may result}}%
7424         \endfortrue
7425       }%
7426       \@glxtr@dostylewarn

```

Set up the style for the given category.

```

7426      \csdef{@glsabbrv@current@#1}{#2}%
7427      \glsxtr@applyabbrvstyle{#2}%
7428      }%
7429      }%
7430 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7431 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7432   \csuse{@glsabbrv@dispstyle@setup@#1}%
7433   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7434 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

7435 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7436   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7437 }

```

`\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.

```

7438 \newcommand*{\newabbreviationstyle}[3]{%
7439   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7440   {%
7441     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7442       defined}{}%
7443   }%
7444   {%
7445     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7446     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7447     #2}%
7448     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7449     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7450     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7451     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7452     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7453     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7454     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7455     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7456     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7457     #3}%
7458   }%
7459 }

```

renewabbreviationstyle

```
7460 \newcommand*{\renewabbreviationstyle}[3]{%
7461   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7462   {%
7463     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7464   }%
7465   {%
7466     \csdef{@glsabbrv@dispstyle@setup@#1}{%
        Initialise hook to do nothing. The style may change this.
7467       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7468       #2}%
7469     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
        Assume in-line form is the same as first use. The style may change this.
7470       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7471       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7472       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7473       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7474       #3}%
7475     }%
7476 }
```

renewabbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
7477 \newcommand*{\letabbreviationstyle}[2]{%
7478   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7479   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7480 }
```

renewdeprecated@abbrstyle

`\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```
7481 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7482   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7483     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7484     \csuse{@glsabbrv@dispstyle@setup@#2}%
7485   }%
7486   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7487 }
```

renewdeprecatedAbbrStyle Generate warning for deprecated style use.

```
7488 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7489   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7490     use ‘#2’ instead}%
7491 }
```

eAbbrStyleSetup

```
7492 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7493   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7494   {%
7495     \PackageError{glossaries-extra}%
7496     {Unknown abbreviation style definitions ‘#1’}{}%
7497   }%
7498   {%
7499     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7500   }%
7501 }
```

seAbbrStyleFmts

```
7502 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7503   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7504   {%
7505     \PackageError{glossaries-extra}%
7506     {Unknown abbreviation style formats ‘#1’}{}%
7507   }%
7508   {%
7509     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7510   }%
7511 }
```

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.

```
7512 \newif\ifglsxtrinsertinside
7513 \glsxtrinsertinsidefalse
```

trlongshortname

```
7514 \newcommand*{\glsxtrlongshortname}{%
7515   \protect\glsabbrvfont{\the\glsshorttok}%
7516 }
```

long-short

```
7517 \newabbreviationstyle{long-short}%
7518 {%
```



```

7519 \renewcommand*{\CustomAbbreviationFields}{%
7520   name={\glxtrlongshortname},
7521   sort={\the\glsshorttok},
7522   first={\protect\glsfirstlongfont{\the\glslongtok}%
7523     \protect\glxtrfullsep{\the\glslabeltok}%
7524     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7525   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7526     \protect\glxtrfullsep{\the\glslabeltok}%
7527     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

7528   plural={\protect\glsabbrvfont{\the\glsshortpltok}}},%
7529   description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7530 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7531   \glshasattribute{\the\glslabeltok}{regular}%
7532   {%
7533     \glssetattribute{\the\glslabeltok}{regular}{false}%
7534   }%
7535   {}%
7536 }%
7537 }%
7538 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7539 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7540 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7541 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7542 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7543 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7544 \renewcommand*{\glxtrfullformat}[2]{%
7545   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7546   \ifglxtrininsertinside\else##2\fi
7547   \glxtrfullsep{##1}%
7548   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7549 }%
7550 \renewcommand*{\glxtrfullplformat}[2]{%
7551   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7552   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7553   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7554 }%
7555 \renewcommand*{\Glsxtrfullformat}[2]{%
7556   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7557   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7558   \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7559 }%
7560 \renewcommand*{\Glsxtrfullplformat}[2]{%
7561   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7562   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7563     \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7564   }%
7565 }

```

Set this as the default style for general abbreviations:

```

7566 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7567 \newcommand*{\glsxtrlongshortdescsort}{%
7568   \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7569 }

```

ngshortdescname

```

7570 \newcommand*{\glsxtrlongshortdescname}{%
7571   \protect\glslongfont{\the\glslongtok}
7572   \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7573 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7574 \newabbreviationstyle{long-short-desc}%
7575 {%
7576   \renewcommand*{\CustomAbbreviationFields}{%
7577     name={\glsxtrlongshortdescname},
7578     sort={\glsxtrlongshortdescsort},%
7579     first={\protect\glsfirstlongfont{\the\glslongtok}%
7580       \protect\glsxtrfullsep{\the\glslabeltok}%
7581       \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7582     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7583       \protect\glsxtrfullsep{\the\glslabeltok}%
7584       \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7585     text={\protect\glsabbrvfont{\the\glsshorttok}},%
7586     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7587   }%

```

Unset the regular attribute if it has been set.

```

7588 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7589   \glshasattribute{\the\glslabeltok}{regular}%
7590   {%
7591     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7592   }%
7593   {}%
7594 }%
7595 }%
7596 {%
7597   \GlsXtrUseAbbrStyleFmts{long-short}%
7598 }

```

trshortlongname

```
7599 \newcommand*{\glxtrshortlongname}{%
7600   \protect\glsabbrvfont{\the\glsshorttok}%
7601 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7602 \newabbreviationstyle{short-long}%
7603 {%
7604   \renewcommand*{\CustomAbbreviationFields}{%
7605     name={\glxtrshortlongname},
7606     sort={\the\glsshorttok},
7607     description={\the\glslongtok},%
7608     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7609       \protect\glxtrfullsep{\the\glslabeltok}%
7610       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7611     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7612       \protect\glxtrfullsep{\the\glslabeltok}%
7613       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7614     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7615 }
```

Unset the regular attribute if it has been set.

```
7615 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7616   \glshasattribute{\the\glslabeltok}{regular}%
7617   {%
7618     \glissetattribute{\the\glslabeltok}{regular}{false}%
7619   }%
7620   {}%
7621 }%
7622 }%
7623 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7624 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7625 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7626 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7627 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7628 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7629 \renewcommand*{\glxtrfullformat}[2]{%
7630   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7631   \ifglxtrininsertinside\else##2\fi
7632   \glxtrfullsep{##1}%
7633   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7634 }%
7635 \renewcommand*{\glxtrfullplformat}[2]{%
7636   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7637   \ifglxtrininsertinside\else##2\fi
7638   \glxtrfullsep{##1}%
7639 }
```

```

7639 \glstrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
7640 }%
7641 \renewcommand*{\Glsxtrfullformat}[2]{%
7642 \glfirstabbrvfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
7643 \ifglstrinsertinside\else##2\fi\glxtrfullsep{##1}%
7644 \glstrparen{\glfirstlongfont{\glaccesslong{##1}}}%
7645 }%
7646 \renewcommand*{\Glsxtrfullplformat}[2]{%
7647 \glfirstabbrvfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
7648 \ifglstrinsertinside\else##2\fi\glxtrfullsep{##1}%
7649 \glstrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
7650 }%
7651 }

```

ortlongdescsort

```

7652 \newcommand*{\glstrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7653 \newcommand*{\glstrshortlongdescname}{%
7654 \protect\glsabbrvfont{\the\glsshorttok}
7655 \glstrparen{\protect\glslongfont{\the\glslongtok}}}%
7656 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7657 \newabbreviationstyle{short-long-desc}%
7658 {%
7659 \renewcommand*{\CustomAbbreviationFields}{%
7660 name={\glstrshortlongdescname},
7661 sort={\glstrshortlongdescsort},
7662 first={\protect\glfirstabbrvfont{\the\glsshorttok}%
7663 \protect\glxtrfullsep{\the\glslabeltok}%
7664 \glstrparen{\protect\glfirstlongfont{\the\glslongtok}}},%
7665 firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}%
7666 \protect\glxtrfullsep{\the\glslabeltok}%
7667 \glstrparen{\protect\glfirstlongfont{\the\glslongpltok}}},%
7668 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7669 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7670 }%

```

Unset the regular attribute if it has been set.

```

7671 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7672 \glshasattribute{\the\glslabeltok}{regular}%
7673 {%
7674 \glissetattribute{\the\glslabeltok}{regular}{false}%
7675 }%
7676 {}%
7677 }%

```

```

7678 }%
7679 {%
7680   \GlsXtrUseAbbrStyleFmts{short-long}%
7681 }

```

ongfootnotefont Only used by the “footnote” styles.

```

7682 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

ongfootnotefont Only used by the “footnote” styles.

```

7683 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

xtrabbrvfootnote `\glsxtrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument *<long>* includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

7684 \newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}

```

xtrfootnotename

```

7685 \newcommand*{\glsxtrfootnotename}{%
7686   \protect\glsabbrvfont{\the\glsshorttok}%
7687 }

```

footnote Short form followed by long form in footnote on first use.

```

7688 \newabbreviationstyle{footnote}%
7689 {%
7690   \renewcommand*{\CustomAbbreviationFields}{%
7691     name={\glsxtrfootnotename},
7692     sort={\the\glsshorttok},
7693     description={\the\glslongtok},%

7694     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7695     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7696     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7697     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7698     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7699     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

7700     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.

```

7701 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7702   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7703   \glsattribute{\the\glslabeltok}{regular}%

```

```

7704    {%
7705        \glssetattribute{\the\glslabeltok}{regular}{false}%
7706    }%
7707    {}%
7708 }%
7709 }%
7710 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7711 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7712 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7713 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7714 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7715 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7716 \renewcommand*\glsxtrfullformat[2]{%
7717     \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7718     \ifglsxtrininsertinside\else##2\fi
7719     \protect\glsxtrabbrvfootnote{##1}%
7720     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7721 }%
7722 \renewcommand*\glsxtrfullplformat[2]{%
7723     \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7724     \ifglsxtrininsertinside\else##2\fi
7725     \protect\glsxtrabbrvfootnote{##1}%
7726     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7727 }%
7728 \renewcommand*\Glsxtrfullformat[2]{%
7729     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7730     \ifglsxtrininsertinside\else##2\fi
7731     \protect\glsxtrabbrvfootnote{##1}%
7732     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7733 }%
7734 \renewcommand*\Glsxtrfullplformat[2]{%
7735     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7736     \ifglsxtrininsertinside\else##2\fi
7737     \protect\glsxtrabbrvfootnote{##1}%
7738     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7739 }%

```

The first use full form and the inline full form use the short (long) style.

```

7740 \renewcommand*\glsxtrininlinefullformat[2]{%
7741     \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7742     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7743     \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7744 }%
7745 \renewcommand*\glsxtrininlinefullplformat[2]{%
7746     \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7747     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7748     \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

7749 }%
7750 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7751   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7752   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7753   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7754 }%
7755 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7756   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7757   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7758   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7759 }%
7760 }

```

short-footnote

```

7761 \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.

```

7762 \newabbreviationstyle{postfootnote}%
7763 {%
7764   \renewcommand*{\CustomAbbreviationFields}{%
7765     name={\glxtrfootnotename},
7766     sort={\the\glsshorttok},
7767     description={\the\glslongtok},%
7768     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7769     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7770     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.

```

7771 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7772   \csdef{glxtrpostlink\glscategorylabel}{%
7773     \glxtrifwasfirstuse
7774     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7775       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7776       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7777       }%
7778     }%
7779   }%
7780   \glshasattribute{\the\glslabeltok}{regular}%
7781   {%
7782     \glssetattribute{\the\glslabeltok}{regular}{false}%
7783   }%
7784   }%
7785 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7786 \renewcommand*{\glxtrsetupfulldefs}{%
7787   \let\glxtrifwasfirstuse\@secondoftwo
7788 }%
7789 }%
7790 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7791 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7792 \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvdefaultfont{##1}}%
7793 \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvdefaultfont{##1}}%
7794 \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlongfootnotefont{##1}}%
7795 \renewcommand*{\glxtrlongfont}[1]{\glxtrlongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7796 \renewcommand*{\glxtrfullformat}[2]{%
7797   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7798   \ifglxtrininsertinside\else##2\fi
7799 }%
7800 \renewcommand*{\glxtrfullplformat}[2]{%
7801   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7802   \ifglxtrininsertinside\else##2\fi
7803 }%
7804 \renewcommand*{\Glsxtrfullformat}[2]{%
7805   \glxtrfirstabbrvfont{\Glsxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7806   \ifglxtrininsertinside\else##2\fi
7807 }%
7808 \renewcommand*{\Glsxtrfullplformat}[2]{%
7809   \glxtrfirstabbrvfont{\Glsxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7810   \ifglxtrininsertinside\else##2\fi
7811 }%
```

The first use full form and the inline full form use the short (long) style.

```
7812 \renewcommand*{\glxtrininlinefullformat}[2]{%
7813   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7814   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7815   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslong{##1}}}%
7816 }%
7817 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7818   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7819   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7820   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslongpl{##1}}}%
7821 }%
7822 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
7823   \glxtrfirstabbrvfont{\Glsxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7824   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7825   \glxtrparen{\glxtrfirstlongfootnotefont{\Glsxtraccesslong{##1}}}%
7826 }%
7827 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```



```

7828 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7829 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7830 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7831 }%
7832 }

```

rt-postfootnote

```

7833 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

shortnolongname

```

7834 \newcommand*{\glxtrshortnolongname}{%
7835 \protect\glsabbrvfont{\the\glsshorttok}%
7836 }

```

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.

```

7837 \newabbreviationstyle{short}%
7838 {%
7839 \renewcommand*{\CustomAbbreviationFields}{%
7840 name={\glxtrshortnolongname},
7841 sort={\the\glsshorttok},
7842 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7843 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7844 text={\protect\glsabbrvfont{\the\glsshorttok}},
7845 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7846 description={\the\glslongtok}}%
7847 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7848 \glssetattribute{\the\glslabeltok}{regular}{true}}%
7849 }%
7850 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7851 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7852 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7853 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7854 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7855 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7856 \renewcommand*{\glxtrinlinefullformat}[2]{%
7857 \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7858 \ifglxtrinsertinside##2\fi}%
7859 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7860 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7861 }%
7862 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7863 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%

```

```

7864 \ifglxtrinsertinside##2\fi}%
7865 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7866 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7867 }%
7868 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7869 \protect\glsfirstabbrvfont{\glssaccessshort{##1}%
7870 \ifglxtrinsertinside##2\fi}%
7871 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7872 \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7873 }%
7874 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7875 \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}%
7876 \ifglxtrinsertinside##2\fi}%
7877 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7878 \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7879 }%

```

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.

```

7880 \renewcommand*{\glxtrfullformat}[2]{%
7881 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7882 \ifglxtrinsertinside\else##2\fi
7883 }%
7884 \renewcommand*{\glxtrfullplformat}[2]{%
7885 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7886 \ifglxtrinsertinside\else##2\fi
7887 }%
7888 \renewcommand*{\Glsxtrfullformat}[2]{%
7889 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7890 \ifglxtrinsertinside\else##2\fi
7891 }%
7892 \renewcommand*{\Glsxtrfullplformat}[2]{%
7893 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7894 \ifglxtrinsertinside\else##2\fi
7895 }%
7896 }

```

Set this as the default style for acronyms:

```

7897 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7898 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7899 \newabbreviationstyle{short-nolong-noreg}%
7900 {%
7901 \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7902 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

7903 \glshasattribute{\the\glslabeltok}{regular}%
7904 {%
7905 \glissetattribute{\the\glslabeltok}{regular}{false}%
7906 }%
7907 {}%
7908 }%
7909 }%
7910 {%
7911 \GlsXtrUseAbbrStyleFmts{short-nolong}%
7912 }

```

trshortdescname

```

7913 \newcommand*{\glxtrshortdescname}{%
7914 \protect\glsabbrvfont{\the\glsshorttok}%
7915 }

```

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.

```

7916 \newabbreviationstyle{short-desc}%
7917 {%
7918 \renewcommand*{\CustomAbbreviationFields}{%
7919 name={\glxtrshortdescname},
7920 sort={\the\glsshorttok},
7921 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7922 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7923 text={\protect\glsabbrvfont{\the\glsshorttok}},
7924 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7925 description={\the\glslongtok}}%
7926 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7927 \glissetattribute{\the\glslabeltok}{regular}{true}}%
7928 }%
7929 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7930 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7931 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7932 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7933 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7934 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7935 \renewcommand*{\glxtrinlinefullformat}[2]{%
7936 \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
7937 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7938 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7939 }%
7940 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7941 \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
7942 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7943 \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%

```

```

7944 }%
7945 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7946   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7947   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7948   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7949 }%
7950 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7951   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7952   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7953   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7954 }%

```

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.

```

7955 \renewcommand*{\glxtrfullformat}[2]{%
7956   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7957   \ifglxtrinsertinside\else##2\fi
7958 }%
7959 \renewcommand*{\glxtrfullplformat}[2]{%
7960   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7961   \ifglxtrinsertinside\else##2\fi
7962 }%
7963 \renewcommand*{\Glsxtrfullformat}[2]{%
7964   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7965   \ifglxtrinsertinside\else##2\fi
7966 }%
7967 \renewcommand*{\Glsxtrfullplformat}[2]{%
7968   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7969   \ifglxtrinsertinside\else##2\fi
7970 }%
7971 }

```

ort-nolong-desc

```

7972 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7973 \newabbreviationstyle{short-nolong-desc-noreg}%
7974 {%
7975   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7976 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7977   \glshasattribute{\the\glslabeltok}{regular}%
7978   {%
7979     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7980   }%
7981   {}%
7982 }%
7983 }%
7984 {%

```

```

7985 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7986 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7987 \newabbreviationstyle{nolong-short}%
7988 {%
7989 \GlsXtrUseAbbrStyleSetup{short-nolong}%
7990 }%
7991 {%
7992 \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7993 \renewcommand*{\glxtrinlinefullformat}[2]{%
7994 \protect\glsfirstlongfont{\glsaccesslong{##1}%
7995 \ifglxtrininsertinside##2\fi}%
7996 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7997 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7998 }%
7999 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8000 \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8001 \ifglxtrininsertinside##2\fi}%
8002 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8003 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8004 }%
8005 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8006 \protect\glsfirstlongfont{\glsaccesslong{##1}%
8007 \ifglxtrininsertinside##2\fi}%
8008 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8009 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8010 }%
8011 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8012 \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8013 \ifglxtrininsertinside##2\fi}%
8014 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8015 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8016 }%
8017 }

```

ong-short-noreg Like nolong-short but doesn't set the regular attribute.

```

8018 \newabbreviationstyle{nolong-short-noreg}%
8019 {%
8020 \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

8021 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8022 \glshasattribute{\the\glslabeltok}{regular}%
8023 {%
8024 \glssetattribute{\the\glslabeltok}{regular}{false}%
8025 }%

```

```

8026     {}%
8027   }%
8028 }%
8029 {%
8030   \GlsXtrUseAbbrStyleFmts{nolong-short}%
8031 }

```

noshortdescname

```

8032 \newcommand*{\glxtrlongnoshortdescname}{%
8033   \protect\glslongfont{\the\glslongtok}%
8034 }

```

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.

```

8035 \newabbreviationstyle{long-desc}%
8036 {%
8037   \renewcommand*{\CustomAbbreviationFields}{%
8038     name={\glxtrlongnoshortdescname},
8039     sort={\the\glslongtok},
8040     first={\protect\glslongfont{\the\glslongtok}},
8041     firstplural={\protect\glslongfont{\the\glslongpltok}},
8042     text={\glslongfont{\the\glslongtok}},
8043     plural={\glslongfont{\the\glslongpltok}}}%
8044   }%
8045   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8046     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8047 }%
8048 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8049 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8050 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
8051 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
8052 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8053 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8054 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8055   \glslongfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8056   \ifglxtrininsertinside \else##2\fi
8057 }%
8058 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8059   \glslongfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8060   \ifglxtrininsertinside \else##2\fi
8061 }%
8062 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8063   \glslongfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8064   \ifglxtrininsertinside \else##2\fi
8065 }%

```

```

8066 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8067   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
8068   \ifglsxtrinsertinside \else##2\fi
8069 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8070 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8071   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8072   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8073   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8074 }%
8075 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8076   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8077   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8078   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8079 }%
8080 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8081   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8082   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8083   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8084 }%
8085 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8086   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8087   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8088   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8089 }%

```

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.

```

8090 \renewcommand*{\glsxtrfullformat}[2]{%
8091   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8092   \ifglsxtrinsertinside\else##2\fi
8093 }%
8094 \renewcommand*{\glsxtrfullplformat}[2]{%
8095   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8096   \ifglsxtrinsertinside\else##2\fi
8097 }%
8098 \renewcommand*{\Glsxtrfullformat}[2]{%
8099   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8100   \ifglsxtrinsertinside\else##2\fi
8101 }%
8102 \renewcommand*{\Glsxtrfullplformat}[2]{%
8103   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8104   \ifglsxtrinsertinside\else##2\fi
8105 }%
8106 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

8107 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```
8108 \newabbreviationstyle{long-noshort-desc-noreg}%
8109 {%
8110   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
      Unset the regular attribute if it has been set.
8111   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8112     \glshasattribute{\the\glslabeltok}{regular}%
8113     {%
8114       \glissetattribute{\the\glslabeltok}{regular}{false}%
8115     }%
8116     {}%
8117   }%
8118 }%
8119 {%
8120   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8121 }
```

longnoshortname

```
8122 \newcommand*{\glsxtrlongnoshortname}{%
8123   \protect\glsabbrvfont{\the\glsshorttok}%
8124 }
```

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.

```
8125 \newabbreviationstyle{long}%
8126 {%
8127   \renewcommand*{\CustomAbbreviationFields}{%
8128     name={\glsxtrlongnoshortname},
8129     sort={\the\glsshorttok},
8130     first={\protect\glsfirstlongfont{\the\glslongtok}},
8131     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8132     text={\glslongfont{\the\glslongtok}},
8133     plural={\glslongfont{\the\glslongpltok}},%
8134     description={\the\glslongtok}%
8135   }%
8136   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8137     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8138 }%
8139 {%
8140   \GlsXtrUseAbbrStyleFmts{long-desc}%
8141 }
```

long-noshort Provide a synonym that matches similar styles.

```
8142 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.


```

8143 \newabbreviationstyle{long-noshort-noreg}%
8144 {%
8145   \GlsXtrUseAbbrStyleSetup{long-noshort}%
      Unset the regular attribute if it has been set.
8146   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8147     \glshasattribute{\the\glslabeltok}{regular}%
8148     {%
8149       \glissetattribute{\the\glslabeltok}{regular}{false}%
8150     }%
8151     {}}%
8152   }%
8153 }%
8154 {%
8155   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8156 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

```

\glsxtrscfont   Maintained for backward-compatibility.
8157 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}

\glsabbrvscfont Added for consistent naming.
8158 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}

sxtfirstscfont  Maintained for backward-compatibility.
8159 \newcommand*{\sxtfirstscfont}[1]{\glsabbrvscfont{#1}}

irstabbrvscfont Added for consistent naming.
8160 \newcommand*{\irstabbrvscfont}{\sxtfirstscfont}

```

and for the default short form suffix:

```

\glsxtrscsuffix
8161 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}

long-short-sc
8162 \newabbreviationstyle{long-short-sc}%
8163 {%
8164   \renewcommand*{\CustomAbbreviationFields}{%
8165     name={\glsxtrlongshortname},
8166     sort={\the\glsshorttok},
8167     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8168       \protect\glsxtrfullsep{\the\glslabeltok}%
8169       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8170     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8171       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

8172     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8173     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
8174     description={\the\glslongtok}}%
8175 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8176     \glsattribute{\the\glslabeltok}{regular}%
8177     {%
8178         \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8179     }%
8180     {}%
8181 }%
8182 }%
8183 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8184 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8185 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8186 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

8187 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8188 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8189 \renewcommand*\glxtrfullformat[2]{%
8190     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8191     \ifglxtrinsertinside\else##2\fi
8192     \glxtrfullsep{##1}%
8193     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8194 }%
8195 \renewcommand*\glxtrfullplformat[2]{%
8196     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8197     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8198     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8199 }%
8200 \renewcommand*\Glsxtrfullformat[2]{%
8201     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8202     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8203     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8204 }%
8205 \renewcommand*\Glsxtrfullplformat[2]{%
8206     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8207     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8208     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8209 }%
8210 }

```

g-short-sc-desc

```

8211 \newabbreviationstyle{long-short-sc-desc}%
8212 {%
8213     \renewcommand*\CustomAbbreviationFields{%

```

```

8214   name={\glxtrlongshortdescname},
8215   sort={\glxtrlongshortdescsort},%
8216   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8217     \protect\glxtrfullsep{\the\glslabeltok}%
8218     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8219   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8220     \protect\glxtrfullsep{\the\glslabeltok}%
8221     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8222   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8223   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8224 }%

```

Unset the regular attribute if it has been set.

```

8225 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8226   \glshasattribute{\the\glslabeltok}{regular}%
8227   {%
8228     \glissetattribute{\the\glslabeltok}{regular}{false}%
8229   }%
8230   {}%
8231 }%
8232 }%
8233 {%

```

As long-short-sc style:

```

8234 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8235 }

```

Now the short (long) version

```

8236 \newabbreviationstyle{short-sc-long}%
8237 {%
8238   \renewcommand*{\CustomAbbreviationFields}{%
8239     name={\glxtrshortlongname},
8240     sort={\the\glsshorttok},
8241     description={\the\glslongtok},%
8242     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8243       \protect\glxtrfullsep{\the\glslabeltok}%
8244       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8245     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8246       \protect\glxtrfullsep{\the\glslabeltok}%
8247       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8248     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8249 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8250   \glshasattribute{\the\glslabeltok}{regular}%
8251   {%
8252     \glissetattribute{\the\glslabeltok}{regular}{false}%
8253   }%
8254   {}%
8255 }%
8256 }%

```

8257 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8258 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8259 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8260 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8261 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8262 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8263 \renewcommand*{\glxtrfullformat}[2]{%
8264   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8265   \ifglxtrininsertinside\else##2\fi
8266   \glxtrfullsep{##1}%
8267   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8268 }%
8269 \renewcommand*{\glxtrfullplformat}[2]{%
8270   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8271   \ifglxtrininsertinside\else##2\fi
8272   \glxtrfullsep{##1}%
8273   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8274 }%
8275 \renewcommand*{\Glsxtrfullformat}[2]{%
8276   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8277   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8278   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8279 }%
8280 \renewcommand*{\Glsxtrfullplformat}[2]{%
8281   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8282   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8283   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8284 }%
8285 }
```

As before but user provides description

```
8286 \newabbreviationstyle{short-sc-long-desc}%
8287 {%
8288   \renewcommand*{\CustomAbbreviationFields}{%
8289     name={\glxtrshortlongdescname},
8290     sort={\glxtrshortlongdescsort},
8291     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8292       \protect\glxtrfullsep{\the\glslabeltok}%
8293       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8294     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8295       \protect\glxtrfullsep{\the\glslabeltok}%
8296       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8297     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8298     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8299   }%
```

Unset the regular attribute if it has been set.

```

8300 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8301   \glshasattribute{\the\glslabeltok}{regular}}%
8302   {%
8303     \glissetattribute{\the\glslabeltok}{regular}{false}}%
8304   }%
8305   {}%
8306 }%
8307 }%
8308 {%

```

As short-sc-long style:

```

8309 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8310 }

```

short-sc

```

8311 \newabbreviationstyle{short-sc}%
8312 {%
8313   \renewcommand*{\CustomAbbreviationFields}{%
8314     name={\glxtrshortnolongname},
8315     sort={\the\glsshorttok},
8316     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8317     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8318     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8319     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8320     description={\the\glslongtok}}%
8321   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8322     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8323   }%
8324 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8325 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8326 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8327 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8328 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8329 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8330 \renewcommand*{\glxtrinlinefullformat}[2]{%
8331   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8332   \ifglxtrininsertinside##2\fi}%
8333   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8334   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8335 }%
8336 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8337   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8338   \ifglxtrininsertinside##2\fi}%
8339   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8340   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8341 }%

```

```

8342 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8343   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
8344     \ifglxtrinsertinside##2\fi}%
8345   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8346   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8347 }%
8348 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8349   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
8350     \ifglxtrinsertinside##2\fi}%
8351   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8352   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8353 }%

```

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.

```

8354 \renewcommand*{\glxtrfullformat}[2]{%
8355   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8356   \ifglxtrinsertinside\else##2\fi
8357 }%
8358 \renewcommand*{\glxtrfullplformat}[2]{%
8359   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8360   \ifglxtrinsertinside\else##2\fi
8361 }%
8362 \renewcommand*{\Glsxtrfullformat}[2]{%
8363   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8364   \ifglxtrinsertinside\else##2\fi
8365 }%
8366 \renewcommand*{\Glsxtrfullplformat}[2]{%
8367   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8368   \ifglxtrinsertinside\else##2\fi
8369 }%
8370 }

```

short-sc-nolong

```

8371 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8372 \newabbreviationstyle{short-sc-desc}%
8373 {%
8374   \renewcommand*{\CustomAbbreviationFields}{%
8375     name={\glxtrshortdesname},
8376     sort={\the\glsshorttok},
8377     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8378     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8379     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8380     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8381     description={\the\glslongtok}}%
8382 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8383   \glsetattribute{\the\glslabeltok}{regular}{true}}%

```

8384 }%
8385 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8386 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8387 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8388 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvscfont{##1}}%
8389 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8390 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8391 \renewcommand*{\glxtrinlinefullformat}[2]{%
8392   \glssfirstabbrvscfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8393   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8394   \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
8395 }%
8396 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8397   \glssfirstabbrvscfont{\glssaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8398   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8399   \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8400 }%
8401 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8402   \glssfirstabbrvscfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8403   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8404   \glxtrparen{\glssfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8405 }%
8406 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8407   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8408   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8409   \glxtrparen{\glssfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8410 }%
```

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.

```
8411 \renewcommand*{\glxtrfullformat}[2]{%
8412   \glssfirstabbrvscfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8413   \ifglxtrininsertinside\else##2\fi
8414 }%
8415 \renewcommand*{\glxtrfullplformat}[2]{%
8416   \glssfirstabbrvscfont{\glssaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8417   \ifglxtrininsertinside\else##2\fi
8418 }%
8419 \renewcommand*{\Glsxtrfullformat}[2]{%
8420   \glssfirstabbrvscfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8421   \ifglxtrininsertinside\else##2\fi
8422 }%
8423 \renewcommand*{\Glsxtrfullplformat}[2]{%
8424   \glssfirstabbrvscfont{\glssaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8425   \ifglxtrininsertinside\else##2\fi
8426 }%
8427 }
```

-sc-nolong-desc

```
8428 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
8429 \newabbreviationstyle{nolong-short-sc}%
8430 {%
8431   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8432 }%
8433 {%
8434   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8435 \renewcommand*{\glxtrinlinefullformat}[2]{%
8436   \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
8437     \ifglxtrininsertinside##2\fi}%
8438   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8439   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8440 }%
8441 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8442   \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
8443     \ifglxtrininsertinside##2\fi}%
8444   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8445   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8446 }%
8447 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8448   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8449     \ifglxtrininsertinside##2\fi}%
8450   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8451   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8452 }%
8453 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8454   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8455     \ifglxtrininsertinside##2\fi}%
8456   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8457   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8458 }%
8459 }
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```
8460 \newabbreviationstyle{long-noshort-sc}%
8461 {%
8462   \renewcommand*{\CustomAbbreviationFields}{%
8463     name={\glxtrlongnoshortname},
8464     sort={\the\glssshorttok},
8465     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8466     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8467     text={\protect\glslongdefaultfont{\the\glslongtok}},
8468     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
```



```

8469     description={\the\glslongtok}%
8470 }%
8471 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8472     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8473 }%
8474 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8475 \renewcommand*\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8476 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8477 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8478 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8479 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8480 \renewcommand*\glsxtrsubsequentfmt}[2]{%
8481     \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8482     \ifglsxtrininsertinside \else##2\fi
8483 }%
8484 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
8485     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8486     \ifglsxtrininsertinside \else##2\fi
8487 }%
8488 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8489     \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8490     \ifglsxtrininsertinside \else##2\fi
8491 }%
8492 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8493     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8494     \ifglsxtrininsertinside \else##2\fi
8495 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8496 \renewcommand*\glsxtrinlinefullformat}[2]{%
8497     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8498     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8499     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8500 }%
8501 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8502     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8503     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8504     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8505 }%
8506 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8507     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8508     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8509     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8510 }%
8511 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8512     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8513     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8514 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8515 }%

```

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.

```

8516 \renewcommand*{\glxtrfullformat}[2]{%
8517 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8518 \ifglxtrininsertinside\else##2\fi
8519 }%
8520 \renewcommand*{\glxtrfullplformat}[2]{%
8521 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8522 \ifglxtrininsertinside\else##2\fi
8523 }%
8524 \renewcommand*{\Glsxtrfullformat}[2]{%
8525 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8526 \ifglxtrininsertinside\else##2\fi
8527 }%
8528 \renewcommand*{\Glsxtrfullplformat}[2]{%
8529 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8530 \ifglxtrininsertinside\else##2\fi
8531 }%
8532 }

```

long-sc Backward compatibility:

```

8533 \@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`.

```

8534 \newabbreviationstyle{long-noshort-sc-desc}%
8535 {%
8536 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8537 }%
8538 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8539 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8540 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8541 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8542 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8543 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8544 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8545 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8546 \ifglxtrininsertinside \else##2\fi
8547 }%
8548 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8549 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8550 \ifglxtrininsertinside \else##2\fi
8551 }%

```

```

8552 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8553   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8554   \ifglxtrinsertinside \else##2\fi
8555 }%
8556 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8557   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8558   \ifglxtrinsertinside \else##2\fi
8559 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8560 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8561   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8562   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8563   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8564 }%
8565 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8566   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8567   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8568   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8569 }%
8570 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8571   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8572   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8573   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8574 }%
8575 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8576   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8577   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8578   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8579 }%

```

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.

```

8580 \renewcommand*{\Glsxtrfullformat}[2]{%
8581   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8582   \ifglxtrinsertinside\else##2\fi
8583 }%
8584 \renewcommand*{\Glsxtrfullplformat}[2]{%
8585   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8586   \ifglxtrinsertinside\else##2\fi
8587 }%
8588 \renewcommand*{\Glsxtrfullformat}[2]{%
8589   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8590   \ifglxtrinsertinside\else##2\fi
8591 }%
8592 \renewcommand*{\Glsxtrfullplformat}[2]{%
8593   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8594   \ifglxtrinsertinside\else##2\fi
8595 }%
8596 }

```

long-desc-sc Backward compatibility:

```
8597 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
8598 \newabbreviationstyle{short-sc-footnote}%
8599 {%
8600   \renewcommand*{\CustomAbbreviationFields}{%
8601     name={\glxtrfootnotename},
8602     sort={\the\glsshorttok},
8603     description={\the\glslongtok},%
8604     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8605       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8606       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8607     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8608       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8609       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8610     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8611   }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
8611   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8612     \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8613     \glsattribute{\the\glslabeltok}{regular}%
8614     {%
8615       \glssetattribute{\the\glslabeltok}{regular}{false}%
8616     }%
8617   }%
8618 }%
8619 }%
8620 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8621 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8622 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8623 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8624 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8625 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
8626 \renewcommand*{\glxtrfullformat}[2]{%
8627   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8628   \ifglxtrininsertinside\else##2\fi
8629   \protect\glxtrabbrvfootnote{##1}%
8630   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8631 }%
8632 \renewcommand*{\glxtrfullplformat}[2]{%
8633   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8634   \ifglxtrininsertinside\else##2\fi
8635   \protect\glxtrabbrvfootnote{##1}%
8636   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8637 }
```

```

8637 }%
8638 \renewcommand*{\Glsxtrfullformat}[2]{%
8639   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8640   \ifglxtrinsertinside\else##2\fi
8641   \protect\glxtrabbrvfootnote{##1}%
8642   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8643 }%
8644 \renewcommand*{\Glsxtrfullplformat}[2]{%
8645   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8646   \ifglxtrinsertinside\else##2\fi
8647   \protect\glxtrabbrvfootnote{##1}%
8648   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8649 }%

```

The first use full form and the inline full form use the short (long) style.

```

8650 \renewcommand*{\glxtrinlinefullformat}[2]{%
8651   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8652   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8653   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8654 }%
8655 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8656   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8657   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8658   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8659 }%
8660 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8661   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8662   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8663   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8664 }%
8665 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8666   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8667   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8668   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8669 }%
8670 }

```

footnote-sc Backward compatibility:

```

8671 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8672 \newabbreviationstyle{short-sc-postfootnote}%
8673 {%
8674   \renewcommand*{\CustomAbbreviationFields}{%
8675     name={\glxtrfootnotename},
8676     sort={\the\glsshorttok},
8677     description={\the\glslongtok},%
8678     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8679     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8680     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.

```
8681 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8682   \csdef{glxstrpostlink\glscategorylabel}{%
8683     \glxtrifwasfirstuse
8684     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
8685       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8686       {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
8687     }%
8688   }%
8689 }%
8690 \glshasattribute{\the\glslabeltok}{regular}%
8691 {%
8692   \glissetattribute{\the\glslabeltok}{regular}{false}%
8693   }%
8694   }%
8695 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8696 \renewcommand*{\glxtrsetupfulldefs}{%
8697   \let\glxtrifwasfirstuse\@secondoftwo
8698 }%
8699 }%
8700 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8701 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8702 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
8703 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8704 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8705 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8706 \renewcommand*{\glxtrfullformat}[2]{%
8707   \glsfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8708   \ifglxtrininsertinside\else##2\fi
8709 }%
8710 \renewcommand*{\glxtrfullplformat}[2]{%
8711   \glsfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8712   \ifglxtrininsertinside\else##2\fi
8713 }%
8714 \renewcommand*{\Glsxtrfullformat}[2]{%
8715   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8716   \ifglxtrininsertinside\else##2\fi
8717 }%
8718 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

8719 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8720 \ifglxtrinsertinside\else##2\fi
8721 }%

```

The first use full form and the inline full form use the short (long) style.

```

8722 \renewcommand*{\glxtrinlinefullformat}[2]{%
8723 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8724 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8725 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8726 }%
8727 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8728 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8729 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8730 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8731 }%
8732 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8733 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8734 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8735 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8736 }%
8737 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8738 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8739 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8740 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8741 }%
8742 }

```

postfootnote-sc Backward compatibility:

```

8743 \@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.
8744 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
8745 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8746 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
8747 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

```

and for the default short form suffix:

\glxtrsmsuffix

```
8748 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8749 \newabbreviationstyle{long-short-sm}%
8750 {%
8751   \renewcommand*{\CustomAbbreviationFields}{%
8752     name={\glxtrlongshortname},
8753     sort={\the\glsshorttok},
8754     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8755       \protect\glxtrfullsep{\the\glslabeltok}%
8756       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8757     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8758       \protect\glxtrfullsep{\the\glslabeltok}%
8759       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8760     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},%
8761     description={\the\glslongtok}}%
8762   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8763     \glshasattribute{\the\glslabeltok}{regular}%
8764     {%
8765       \glissetattribute{\the\glslabeltok}{regular}{false}%
8766     }%
8767   }%
8768 }%
8769 }%
8770 {%
8771   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8772   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8773   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8774 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8775 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8776 \renewcommand*{\glxtrfullformat}[2]{%
8777   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8778   \ifglxtrininsertinside\else##2\fi
8779   \glxtrfullsep{##1}%
8780   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8781 }%
8782 \renewcommand*{\glxtrfullplformat}[2]{%
8783   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8784   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8785   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8786 }%
8787 \renewcommand*{\Glsxtrfullformat}[2]{%
8788   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8789   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8790   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%

```



```

8791 }%
8792 \renewcommand*{\Glsxtrfullplformat}[2]{%
8793   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8794   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8795   \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8796 }%
8797 }

```

g-short-sm-desc

```

8798 \newabbreviationstyle{long-short-sm-desc}%
8799 {%
8800   \renewcommand*{\CustomAbbreviationFields}{%
8801     name={\glxtrlongshortdescname},
8802     sort={\glxtrlongshortdescsort},%
8803     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8804       \protect\glxtrfullsep{\the\glslabeltok}%
8805       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8806     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8807       \protect\glxtrfullsep{\the\glslabeltok}%
8808       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8809     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8810     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8811   }%

```

Unset the regular attribute if it has been set.

```

8812 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8813   \glshasattribute{\the\glslabeltok}{regular}%
8814   {%
8815     \glissetattribute{\the\glslabeltok}{regular}{false}%
8816   }%
8817   {}%
8818 }%
8819 }%
8820 {%

```

As long-short-sm style:

```

8821 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8822 }

```

short-sm-long Now the short (long) version

```

8823 \newabbreviationstyle{short-sm-long}%
8824 {%
8825   \renewcommand*{\CustomAbbreviationFields}{%
8826     name={\glxtrshortlongname},
8827     sort={\the\glsshorttok},
8828     description={\the\glslongtok},%
8829     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8830       \protect\glxtrfullsep{\the\glslabeltok}%
8831       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8832     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%

```

```

8833 \protect\glxtrfullsep{\the\glslabeltok}%
8834 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8835 plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8836 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8837 \glshasattribute{\the\glslabeltok}{regular}%
8838 {%
8839 \glissetattribute{\the\glslabeltok}{regular}{false}%
8840 }%
8841 {}%
8842 }%
8843 }%
8844 {%
8845 \renewcommand*\glssabbrvfont[1]{\glssabbrvsmfont{##1}}%
8846 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8847 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8848 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8849 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8850 \renewcommand*{\glxtrfullformat}[2]{%
8851 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8852 \ifglxtrininsertinside\else##2\fi
8853 \glxtrfullsep{##1}%
8854 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8855 }%
8856 \renewcommand*{\glxtrfullplformat}[2]{%
8857 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8858 \ifglxtrininsertinside\else##2\fi
8859 \glxtrfullsep{##1}%
8860 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8861 }%
8862 \renewcommand*{\Glsxtrfullformat}[2]{%
8863 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8864 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8865 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8866 }%
8867 \renewcommand*{\Glsxtrfullplformat}[2]{%
8868 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8869 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8870 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8871 }%
8872 }

```

rt-sm-long-desc As before but user provides description

```

8873 \newabbreviationstyle{short-sm-long-desc}%
8874 {%
8875 \renewcommand*{\CustomAbbreviationFields}{%
8876 name={\glxtrshortlongdescname},

```

```

8877     sort={\glxtrshortlongdescsort},
8878     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8879       \protect\glxtrfullsep{\the\glslabeltok}%
8880       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8881     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8882       \protect\glxtrfullsep{\the\glslabeltok}%
8883       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8884     text={\protect\glsabbrvsmfont{\the\glsshorttok}}%,
8885     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8886   }%

```

Unset the regular attribute if it has been set.

```

8887   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8888     \glshasattribute{\the\glslabeltok}{regular}%
8889     {%
8890       \glissetattribute{\the\glslabeltok}{regular}{false}%
8891     }%
8892   {}%
8893 }%
8894 }%
8895 {%

```

As short-sm-long style:

```

8896   \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8897 }

```

short-sm

```

8898 \newabbreviationstyle{short-sm}%
8899 {%
8900   \renewcommand*{\CustomAbbreviationFields}{%
8901     name={\glxtrshortnolongname},
8902     sort={\the\glsshorttok},
8903     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8904     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8905     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8906     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8907     description={\the\glslongtok}}%
8908   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8909     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8910 }%
8911 {%
8912   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8913   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8914   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8915   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8916   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8917   \renewcommand*\glxtrinlinfullformat[2]{%
8918     \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%

```

```

8919     \ifglxtrinsertinside##2\fi}%
8920     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8921     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8922 }%
8923 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8924     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
8925         \ifglxtrinsertinside##2\fi}%
8926     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8927     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8928 }%

8929 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8930     \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8931         \ifglxtrinsertinside##2\fi}%
8932     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8933     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8934 }%
8935 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8936     \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8937         \ifglxtrinsertinside##2\fi}%
8938     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8939     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8940 }%

```

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.

```

8941 \renewcommand*{\glxtrfullformat}[2]{%
8942     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8943     \ifglxtrinsertinside\else##2\fi
8944 }%
8945 \renewcommand*{\glxtrfullplformat}[2]{%
8946     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8947     \ifglxtrinsertinside\else##2\fi
8948 }%
8949 \renewcommand*{\Glsxtrfullformat}[2]{%
8950     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8951     \ifglxtrinsertinside\else##2\fi
8952 }%
8953 \renewcommand*{\Glsxtrfullplformat}[2]{%
8954     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8955     \ifglxtrinsertinside\else##2\fi
8956 }%
8957 }

```

short-sm-nolong

```
8958 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8959 \newabbreviationstyle{short-sm-desc}%

```

```

8960 {%
8961   \renewcommand*{\CustomAbbreviationFields}{%
8962     name={\glxtrshortdescname},
8963     sort={\the\glsshorttok},
8964     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8965     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8966     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8967     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8968     description={\the\glslongtok}}%
8969   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8970     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8971 }%
8972 {%
8973   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8974   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8975   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8976   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8977   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8978   \renewcommand*\glsxtrinlinefullformat}[2]{%
8979     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8980     \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8981     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8982   }%
8983   \renewcommand*\glsxtrinlinefullplformat}[2]{%
8984     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8985     \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8986     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8987   }%
8988   \renewcommand*\Glsxtrinlinefullformat}[2]{%
8989     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8990     \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8991     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8992   }%
8993   \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8994     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8995     \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8996     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8997   }%

```

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.

```

8998   \renewcommand*\glsxtrfullformat}[2]{%
8999     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9000     \ifglxtrininsertinside\else##2\fi
9001   }%
9002   \renewcommand*\glsxtrfullplformat}[2]{%
9003     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9004     \ifglxtrininsertinside\else##2\fi

```

```

9005 }%
9006 \renewcommand*{\Glsxtrfullformat}[2]{%
9007   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9008   \ifglxtrinsertinside\else##2\fi
9009 }%
9010 \renewcommand*{\Glsxtrfullplformat}[2]{%
9011   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9012   \ifglxtrinsertinside\else##2\fi
9013 }%
9014 }

```

-sm-nolong-desc

```

9015 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

9016 \newabbreviationstyle{nolong-short-sm}%
9017 {%
9018   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9019 }%
9020 {%
9021   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9022 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9023   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9024     \ifglxtrinsertinside##2\fi}%
9025   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9026   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9027 }%
9028 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9029   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9030     \ifglxtrinsertinside##2\fi}%
9031   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9032   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9033 }%
9034 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9035   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9036     \ifglxtrinsertinside##2\fi}%
9037   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9038   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9039 }%
9040 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9041   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9042     \ifglxtrinsertinside##2\fi}%
9043   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9044   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9045 }%
9046 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9047 \newabbreviationstyle{long-noshort-sm}%
9048 {%
9049   \renewcommand*{\CustomAbbreviationFields}{%
9050     name={\glsxtrlongnoshortname},
9051     sort={\the\glsshorttok},
9052     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9053     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9054     text={\protect\glslongdefaultfont{\the\glslongtok}},
9055     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9056     description={\the\glslongtok}%
9057   }%
9058   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9059     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9060 }%
9061 {%
9062   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
9063   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9064   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
9065   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9066   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9067 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9068   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9069   \ifglsxtrininsertinside \else##2\fi
9070 }%
9071 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9072   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9073   \ifglsxtrininsertinside \else##2\fi
9074 }%
9075 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9076   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9077   \ifglsxtrininsertinside \else##2\fi
9078 }%
9079 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9080   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9081   \ifglsxtrininsertinside \else##2\fi
9082 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9083 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9084   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9085   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9086   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
9087 }%
9088 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9089   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9090   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

9091 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9092 }%
9093 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9094 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9095 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9096 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9097 }%
9098 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9099 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9100 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9101 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9102 }%

```

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.

```

9103 \renewcommand*{\glxtrfullformat}[2]{%
9104 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9105 \ifglxtrinsertinside\else##2\fi
9106 }%
9107 \renewcommand*{\glxtrfullplformat}[2]{%
9108 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9109 \ifglxtrinsertinside\else##2\fi
9110 }%
9111 \renewcommand*{\Glsxtrfullformat}[2]{%
9112 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9113 \ifglxtrinsertinside\else##2\fi
9114 }%
9115 \renewcommand*{\Glsxtrfullplformat}[2]{%
9116 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9117 \ifglxtrinsertinside\else##2\fi
9118 }%
9119 }

```

long-sm Backward compatibility:

```

9120 \@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`.

```

9121 \newabbreviationstyle{long-noshort-sm-desc}%
9122 {%
9123 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9124 }%
9125 {%
9126 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9127 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9128 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9129 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9130 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).


```

9131 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9132   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9133   \ifglxtrinsertinside \else##2\fi
9134 }%
9135 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9136   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9137   \ifglxtrinsertinside \else##2\fi
9138 }%
9139 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9140   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9141   \ifglxtrinsertinside \else##2\fi
9142 }%
9143 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9144   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9145   \ifglxtrinsertinside \else##2\fi
9146 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9147 \renewcommand*{\glxtrinlinefullformat}[2]{%
9148   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9149   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9150   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9151 }%
9152 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9153   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9154   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9155   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9156 }%
9157 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9158   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9159   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9160   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9161 }%
9162 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9163   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9164   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9165   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9166 }%

```

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.

```

9167 \renewcommand*{\glxtrfullformat}[2]{%
9168   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9169   \ifglxtrinsertinside\else##2\fi
9170 }%
9171 \renewcommand*{\glxtrfullplformat}[2]{%
9172   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9173   \ifglxtrinsertinside\else##2\fi
9174 }%
9175 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9176 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9177 \ifglxtrinsertinside\else##2\fi
9178 }%
9179 \renewcommand*{\Glsxtrfullplformat}[2]{%
9180 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9181 \ifglxtrinsertinside\else##2\fi
9182 }%
9183 }

```

long-desc-sm Backward compatibility:

```

9184 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9185 \newabbreviationstyle{short-sm-footnote}%
9186 {%
9187 \renewcommand*{\CustomAbbreviationFields}{%
9188 name={\glxtrfootnotename},
9189 sort={\the\glsshorttok},
9190 description={\the\glslongtok},%
9191 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9192 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9193 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9194 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9195 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9196 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9197 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.

```

9198 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9199 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9200 \glsattribute{\the\glslabeltok}{regular}%
9201 {%
9202 \glssetattribute{\the\glslabeltok}{regular}{false}%
9203 }%
9204 {}%
9205 }%
9206 }%
9207 {%
9208 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9209 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9210 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
9211 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9212 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9213 \renewcommand*{\glxtrfullformat}[2]{%
9214 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9215 \ifglxtrinsertinside\else##2\fi
9216 \protect\glxtrabbrvfootnote{##1}%

```

```

9217     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9218 }%
9219 \renewcommand*{\glsxtrfullplformat}[2]{%
9220   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9221   \ifglsxtrininsertinside\else##2\fi
9222   \protect\glsxtrabbrvfootnote{##1}%
9223   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9224 }%
9225 \renewcommand*{\Glsxtrfullformat}[2]{%
9226   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9227   \ifglsxtrininsertinside\else##2\fi
9228   \protect\glsxtrabbrvfootnote{##1}%
9229   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9230 }%
9231 \renewcommand*{\Glsxtrfullplformat}[2]{%
9232   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9233   \ifglsxtrininsertinside\else##2\fi
9234   \protect\glsxtrabbrvfootnote{##1}%
9235   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9236 }%

```

The first use full form and the inline full form use the short (long) style.

```

9237 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9238   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9239   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9240   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9241 }%
9242 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9243   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9244   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9245   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9246 }%
9247 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9248   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9249   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9250   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9251 }%
9252 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9253   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9254   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9255   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9256 }%
9257 }

```

footnote-sm Backward compatibility:

```

9258 \@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

9259 \newabbreviationstyle{short-sm-postfootnote}%
9260 {}

```

```

9261 \renewcommand*{\CustomAbbreviationFields}{%
9262   name={\glxtrfootnotename},
9263   sort={\the\glsshorttok},
9264   description={\the\glslongtok},%
9265   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9266   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9267   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.

```

9268 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9269   \csdef{glxtrpostlink\glscategorylabel}{%
9270     \glxtrifwasfirstuse
9271     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9272       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
9273       {\glsfirstlongfootnotefont{\glstentrylong{\glslabel}}}}%
9274   }%
9275   {%
9276   }%
9277   \glshasattribute{\the\glslabeltok}{regular}%
9278   {%
9279     \glsetattribute{\the\glslabeltok}{regular}{false}%
9280   }%
9281   {%
9282   }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9283 \renewcommand*{\glxtrsetupfulldefs}{%
9284   \let\glxtrifwasfirstuse\@secondoftwo
9285   }%
9286 }%
9287 {%
9288 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9289 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9290 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9291 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9292 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9293 \renewcommand*{\glxtrfullformat}[2]{%
9294   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9295   \ifglxtrininsertinside\else##2\fi
9296 }%
9297 \renewcommand*{\glxtrfullplformat}[2]{%
9298   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9299   \ifglxtrininsertinside\else##2\fi
9300 }%

```

```

9301 \renewcommand*{\Glsxtrfullformat}[2]{%
9302   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9303   \ifglxtrinsertinside\else##2\fi
9304 }%
9305 \renewcommand*{\Glsxtrfullplformat}[2]{%
9306   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9307   \ifglxtrinsertinside\else##2\fi
9308 }%

```

The first use full form and the inline full form use the short (long) style.

```

9309 \renewcommand*{\glxtrinlinefullformat}[2]{%
9310   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9311   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9312   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9313 }%
9314 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9315   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9316   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9317   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9318 }%
9319 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9320   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9321   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9322   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9323 }%
9324 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9325   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9326   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9327   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9328 }%
9329 }

```

postfootnote-sm Backward compatibility:

```

9330 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9331 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

9332 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

9333 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

firstlongemfont Only used by the “long-em” styles.

```
9334 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%
```

\glslongemfont Only used by the “long-em” styles.

```
9335 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

long-short-em The long form is just set in the default long font.

```
9336 \newabbreviationstyle{long-short-em}%
9337 {%
9338   \renewcommand*{\CustomAbbreviationFields}{%
9339     name={\glsxtrlongshortname},
9340     sort={\the\glsshorttok},
9341     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
9342       \protect\glsxtrfullsep{\the\glslabeltok}%
9343       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9344     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
9345       \protect\glsxtrfullsep{\the\glslabeltok}%
9346       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9347     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9348     description={\the\glslongtok}}%
9349   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9350     \glshasattribute{\the\glslabeltok}{regular}}%
9351   {%
9352     \glissetattribute{\the\glslabeltok}{regular}{false}%
9353   }%
9354   {}%
9355 }%
9356 }%
9357 {%
9358   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9359   \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvemfont{##1}}%
9360   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrremsuffix}%

```

Use the default long fonts.

```
9361 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9362 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9363 \renewcommand*{\glsxtrfullformat}[2]{%
9364   \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9365   \ifglxtrinsertinside\else##2\fi
9366   \glsxtrfullsep{##1}%
9367   \glsxtrparen{\glfirstabbrvemfont{\glaccessshort{##1}}}%
9368 }%
9369 \renewcommand*{\glsxtrfullplformat}[2]{%
9370   \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9371   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9372   \glsxtrparen{\glfirstabbrvemfont{\glaccessshortpl{##1}}}%
9373 }%
9374 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9375 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9376 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9377 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9378 }%
9379 \renewcommand*{\Glsxtrfullplformat}[2]{%
9380 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9381 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9382 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9383 }%
9384 }

```

g-short-em-desc

```

9385 \newabbreviationstyle{long-short-em-desc}%
9386 {%
9387 \renewcommand*{\CustomAbbreviationFields}{%
9388 name={\glxtrlongshortdescname},
9389 sort={\glxtrlongshortdescsort},%
9390 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9391 \protect\glxtrfullsep{\the\glslabeltok}%
9392 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9393 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9394 \protect\glxtrfullsep{\the\glslabeltok}%
9395 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9396 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9397 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9398 }%

```

Unset the regular attribute if it has been set.

```

9399 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9400 \glshasattribute{\the\glslabeltok}{regular}%
9401 {%
9402 \glissetattribute{\the\glslabeltok}{regular}{false}%
9403 }%
9404 {}%
9405 }%
9406 }%
9407 {%

```

As long-short-em style:

```

9408 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9409 }

```

ong-em-short-em

```

9410 \newabbreviationstyle{long-em-short-em}%
9411 {%
9412 \glslongemfont is used in the description since \glsdesc doesn't set the style.
9412 \renewcommand*{\CustomAbbreviationFields}{%
9413 name={\glxtrlongshortname},
9414 sort={\the\glsshorttok},

```

```

9415 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9416 \protect\glsxtrfullsep{\the\glslabeltok}%
9417 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9418 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9419 \protect\glsxtrfullsep{\the\glslabeltok}%
9420 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9421 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9422 description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9423 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9424 \glsattribute{\the\glslabeltok}{regular}%
9425 {%
9426 \glssetattribute{\the\glslabeltok}{regular}{false}%
9427 }%
9428 {}%
9429 }%
9430}%
9431{%
9432 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9433 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9434 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9435 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9436 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9437 \renewcommand*{\glsxtrfullformat}[2]{%
9438 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9439 \ifglsxtrininsertinside\else##2\fi
9440 \glsxtrfullsep{##1}%
9441 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9442}%
9443 \renewcommand*{\glsxtrfullplformat}[2]{%
9444 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9445 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9446 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9447}%
9448 \renewcommand*{\Glsxtrfullformat}[2]{%
9449 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9450 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9451 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9452}%
9453 \renewcommand*{\Glsxtrfullplformat}[2]{%
9454 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9455 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9456 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9457}%
9458}

```


m-short-em-desc

```
9459 \newabbreviationstyle{long-em-short-em-desc}%
9460 {%
9461   \renewcommand*{\CustomAbbreviationFields}{%
9462     name={\glxtrlongshortdescname},
9463     sort={\glxtrlongshortdescsort},%
9464     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9465       \protect\glxtrfullsep{\the\glslabeltok}%
9466       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9467     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9468       \protect\glxtrfullsep{\the\glslabeltok}%
9469       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9470     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9471     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9472   }%
```

Unset the regular attribute if it has been set.

```
9473   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9474     \glshasattribute{\the\glslabeltok}{regular}%
9475     {%
9476       \glissetattribute{\the\glslabeltok}{regular}{false}%
9477     }%
9478   }%
9479 }%
9480 }%
9481 {%
9482   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9483 }
```

short-em-long Now the short (long) version

```
9484 \newabbreviationstyle{short-em-long}%
9485 {%
9486   \renewcommand*{\CustomAbbreviationFields}{%
9487     name={\glxtrshortlongname},
9488     sort={\the\glsshorttok},
9489     description={\the\glslongtok},%
9490     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9491       \protect\glxtrfullsep{\the\glslabeltok}%
9492       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9493     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9494       \protect\glxtrfullsep{\the\glslabeltok}%
9495       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9496     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9497   }%
```

Unset the regular attribute if it has been set.

```
9497   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9498     \glshasattribute{\the\glslabeltok}{regular}%
9499     {%
9500       \glissetattribute{\the\glslabeltok}{regular}{false}%
9501     }%
```

```

9502   {}%
9503   }%
9504 }%
9505 {%

```

Mostly as short-long style:

```

9506 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9507 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9508 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9509 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9510 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9511 \renewcommand*{\glssxtrfullformat}[2]{%
9512   \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9513   \ifglssxtrininsertinside\else##2\fi
9514   \glssxtrfullsep{##1}%
9515   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9516 }%
9517 \renewcommand*{\glssxtrfullplformat}[2]{%
9518   \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9519   \ifglssxtrininsertinside\else##2\fi
9520   \glssxtrfullsep{##1}%
9521   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9522 }%
9523 \renewcommand*{\Glsxtrfullformat}[2]{%
9524   \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9525   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9526   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9527 }%
9528 \renewcommand*{\Glsxtrfullplformat}[2]{%
9529   \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9530   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9531   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9532 }%
9533 }

```

rt-em-long-desc As before but user provides description

```

9534 \newabbreviationstyle{short-em-long-desc}%
9535 {%
9536   \renewcommand*{\CustomAbbreviationFields}{%
9537     name={\glssxtrshortlongdescname},
9538     sort={\glssxtrshortlongdescsort},
9539     first={\protect\glssfirstabbrvemfont{\the\glssshorttok}}%
9540     \protect\glssxtrfullsep{\the\glsslabeltok}%
9541     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongtok}}},%
9542     firstplural={\protect\glssfirstabbrvemfont{\the\glssshortpltok}}%
9543     \protect\glssxtrfullsep{\the\glsslabeltok}%
9544     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongpltok}}},%
9545     text={\protect\glssabbrvemfont{\the\glssshorttok}},%

```

```

9546 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9547 }%

```

Unset the regular attribute if it has been set.

```

9548 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9549   \glshasattribute{\the\glslabeltok}{regular}%
9550   {%
9551     \glssetattribute{\the\glslabeltok}{regular}{false}%
9552   }%
9553   {}%
9554 }%
9555 }%
9556 {%
9557   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9558 }

```

hort-em-long-em

```

9559 \newabbreviationstyle{short-em-long-em}%
9560 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9561 \renewcommand*{\CustomAbbreviationFields}{%
9562   name={\glstrshortlongname},
9563   sort={\the\glsshorttok},
9564   description={\protect\glslongemfont{\the\glslongtok}},%
9565   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9566   \protect\glstrfullsep{\the\glslabeltok}%
9567   \glstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9568   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9569   \protect\glstrfullsep{\the\glslabeltok}%
9570   \glstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9571   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9572 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9573   \glshasattribute{\the\glslabeltok}{regular}%
9574   {%
9575     \glssetattribute{\the\glslabeltok}{regular}{false}%
9576   }%
9577   {}%
9578 }%
9579 }%
9580 {%
9581   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9582   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9583   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9584   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9585   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

9586 \renewcommand*{\glxtrfullformat}[2]{%
9587   \glsfirstabbrvemfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9588   \ifglxtrinsertinside\else##2\fi
9589   \glxtrfullsep{##1}%
9590   \glxtrparen{\glsfirstlongemfont{\glaccesslong{##1}}}%
9591 }%
9592 \renewcommand*{\glxtrfullplformat}[2]{%
9593   \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9594   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9595   \glxtrfullsep{##1}%
9596   \glxtrparen{\glsfirstlongemfont{\glaccesslongpl{##1}}}%
9597 }%
9598 \renewcommand*{\Glsxtrfullformat}[2]{%
9599   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9600   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9601   \glxtrparen{\glsfirstlongemfont{\glaccesslong{##1}}}%
9602 }%
9603 \renewcommand*{\Glsxtrfullplformat}[2]{%
9604   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9605   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9606   \glxtrparen{\glsfirstlongemfont{\glaccesslongpl{##1}}}%
9607 }%
9608 }

```

em-long-em-desc

```

9609 \newabbreviationstyle{short-em-long-em-desc}%
9610 {%
9611   \renewcommand*{\CustomAbbreviationFields}{%
9612     name={\glxtrshortlongdescname},%
9613     sort={\glxtrshortlongdescsort},%
9614     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9615     \protect\glxtrfullsep{\the\glslabeltok}}%
9616     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9617     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9618     \protect\glxtrfullsep{\the\glslabeltok}}%
9619     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9620     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9621     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9622 }%

```

Unset the regular attribute if it has been set.

```

9623 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9624   \glshasattribute{\the\glslabeltok}{regular}%
9625   {%
9626     \glissetattribute{\the\glslabeltok}{regular}{false}%
9627   }%
9628   {}%
9629 }%

```

```

9630 }%
9631 {%
9632   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9633 }

```

short-em

```

9634 \newabbreviationstyle{short-em}%
9635 {%
9636   \renewcommand*{\CustomAbbreviationFields}{%
9637     name={\glstrshortnolongname},
9638     sort={\the\glsshorttok},
9639     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9640     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9641     text={\protect\glabbrvemfont{\the\glsshorttok}},
9642     plural={\protect\glabbrvemfont{\the\glsshortpltok}},
9643     description={\the\glslongtok}}%
9644   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9645     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9646 }%
9647 {%
9648   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9649   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9650   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9651   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9652   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9653   \renewcommand*{\glxtrinlinefullformat}[2]{%
9654     \protect\glfirstabbrvemfont{\glaccessshort{##1}}%
9655     \ifglxtrininsertinside##2\fi}%
9656   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9657   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9658 }%
9659   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9660     \protect\glfirstabbrvemfont{\glaccessshortpl{##1}}%
9661     \ifglxtrininsertinside##2\fi}%
9662   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9663   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9664 }%

9665   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9666     \protect\glfirstabbrvemfont{\Glsaccessshort{##1}}%
9667     \ifglxtrininsertinside##2\fi}%
9668   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9669   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9670 }%
9671   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9672     \protect\glfirstabbrvemfont{\Glsaccessshortpl{##1}}%
9673     \ifglxtrininsertinside##2\fi}%
9674   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9675 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9676 }%

```

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.

```

9677 \renewcommand*{\glxtrfullformat}[2]{%
9678   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9679   \ifglxtrininsertinside\else##2\fi
9680 }%
9681 \renewcommand*{\glxtrfullplformat}[2]{%
9682   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9683   \ifglxtrininsertinside\else##2\fi
9684 }%
9685 \renewcommand*{\Glsxtrfullformat}[2]{%
9686   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9687   \ifglxtrininsertinside\else##2\fi
9688 }%
9689 \renewcommand*{\Glsxtrfullplformat}[2]{%
9690   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9691   \ifglxtrininsertinside\else##2\fi
9692 }%
9693 }

```

short-em-nolong

```

9694 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9695 \newabbreviationstyle{short-em-desc}%
9696 {%
9697   \renewcommand*{\CustomAbbreviationFields}{%
9698     name={\glxtrshortdescname},
9699     sort={\the\glssshorttok},
9700     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9701     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9702     text={\protect\glsabbrvemfont{\the\glssshorttok}},
9703     plural={\protect\glsabbrvemfont{\the\glssshortpltok}},
9704     description={\the\glslongtok}}%
9705   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9706     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9707 }%
9708 {%
9709   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9710   \renewcommand*{\glsabbrvfnt}[1]{\glsabbrvemfont{##1}}%
9711   \renewcommand*{\glsfirstabbrvfnt}[1]{\glsfirstabbrvemfont{##1}}%
9712   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9713   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9714 \renewcommand*{\glxtrinlinefullformat}[2]{%
9715   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```

```

9716     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9717     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9718 }%
9719 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9720     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9721     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9722     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9723 }%
9724 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9725     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9726     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9727     \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9728 }%
9729 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9730     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9731     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9732     \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9733 }%

```

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.

```

9734 \renewcommand*{\glxtrfullformat}[2]{%
9735     \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9736     \ifglxtrinsertinside\else##2\fi
9737 }%
9738 \renewcommand*{\glxtrfullplformat}[2]{%
9739     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9740     \ifglxtrinsertinside\else##2\fi
9741 }%
9742 \renewcommand*{\Glsxtrfullformat}[2]{%
9743     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9744     \ifglxtrinsertinside\else##2\fi
9745 }%
9746 \renewcommand*{\Glsxtrfullplformat}[2]{%
9747     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9748     \ifglxtrinsertinside\else##2\fi
9749 }%
9750 }

```

-em-nolong-desc

```

9751 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9752 \newabbreviationstyle{nolong-short-em}%
9753 {%
9754     \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9755 }%
9756 {%
9757     \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9758 \renewcommand*{\glxtrinlinefullformat}[2]{%
9759   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9760     \ifglxtrinsertinside##2\fi}%
9761   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9762   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9763 }%
9764 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9765   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9766     \ifglxtrinsertinside##2\fi}%
9767   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9768   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9769 }%
9770 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9771   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9772     \ifglxtrinsertinside##2\fi}%
9773   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9774   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9775 }%
9776 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9777   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9778     \ifglxtrinsertinside##2\fi}%
9779   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9780   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9781 }%
9782 }

```

`long-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```

9783 \newabbreviationstyle{long-noshort-em}%
9784 {%
9785   \renewcommand*{\CustomAbbreviationFields}{%
9786     name={\glxtrlongnoshortname},
9787     sort={\the\glsshorttok},
9788     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9789     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9790     text={\protect\glslongdefaultfont{\the\glslongtok}},
9791     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9792     description={\the\glslongtok}%
9793   }%
9794   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9795     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9796 }%
9797 {%
9798   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9799   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9800   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9801   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9802   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).


```

9803 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9804   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9805   \ifglxtrinsertinside \else##2\fi
9806 }%
9807 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9808   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9809   \ifglxtrinsertinside \else##2\fi
9810 }%
9811 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9812   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9813   \ifglxtrinsertinside \else##2\fi
9814 }%
9815 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9816   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9817   \ifglxtrinsertinside \else##2\fi
9818 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9819 \renewcommand*{\glxtrinlinefullformat}[2]{%
9820   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9821   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9822   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9823 }%
9824 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9825   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9826   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9827   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9828 }%
9829 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9830   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9831   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9832   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9833 }%
9834 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9835   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9836   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9837   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9838 }%

```

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.

```

9839 \renewcommand*{\glxtrfullformat}[2]{%
9840   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9841   \ifglxtrinsertinside\else##2\fi
9842 }%
9843 \renewcommand*{\glxtrfullplformat}[2]{%
9844   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9845   \ifglxtrinsertinside\else##2\fi
9846 }%
9847 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9848 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9849 \ifglxtrinsertinside\else##2\fi
9850 }%
9851 \renewcommand*{\Glsxtrfullplformat}[2]{%
9852 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9853 \ifglxtrinsertinside\else##2\fi
9854 }%
9855 }

```

long-em Backward compatibility:

```

9856 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9857 \newabbreviationstyle{long-em-noshort-em}%
9858 {%
9859 \renewcommand*{\CustomAbbreviationFields}{%
9860 name={\glxtrlongnoshortname},
9861 sort={\the\glsshorttok},
9862 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9863 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9864 text={\protect\glslongemfont{\the\glslongtok}},
9865 plural={\protect\glslongemfont{\the\glslongpltok}},%
9866 description={\protect\glslongemfont{\the\glslongtok}}%
9867 }%
9868 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9869 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9870 }%
9871 {%
9872 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9873 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9874 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9875 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9876 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9877 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9878 \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9879 \ifglxtrinsertinside \else##2\fi
9880 }%
9881 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9882 \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9883 \ifglxtrinsertinside \else##2\fi
9884 }%
9885 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9886 \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9887 \ifglxtrinsertinside \else##2\fi
9888 }%
9889 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9890 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9891 \ifglxtrinsertinside \else##2\fi

```

9892 }%

The inline full form displays the long format followed by the short form in parentheses.

```
9893 \renewcommand*{\glxstrinlinefullformat}[2]{%
9894   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9895   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9896   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9897 }%
9898 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9899   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9900   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9901   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9902 }%
9903 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9904   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9905   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9906   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9907 }%
9908 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9909   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9910   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9911   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9912 }%
```

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.

```
9913 \renewcommand*{\glxstrfullformat}[2]{%
9914   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9915   \ifglxstrinsertinside\else##2\fi
9916 }%
9917 \renewcommand*{\glxstrfullplformat}[2]{%
9918   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9919   \ifglxstrinsertinside\else##2\fi
9920 }%
9921 \renewcommand*{\Glsxtrfullformat}[2]{%
9922   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9923   \ifglxstrinsertinside\else##2\fi
9924 }%
9925 \renewcommand*{\Glsxtrfullplformat}[2]{%
9926   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9927   \ifglxstrinsertinside\else##2\fi
9928 }%
9929 }
```

oshort-em-noreg Like long-em-noshort-em but doesn't set the regular attribute.

```
9930 \newabbreviationstyle{long-em-noshort-em-noreg}%
9931 {%
9932   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```

9933 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9934   \glshasattribute{\the\glslabeltok}{regular}}%
9935   {%
9936     \glissetattribute{\the\glslabeltok}{regular}{false}}%
9937   }%
9938   {}%
9939 }%
9940 }%
9941 {%
9942   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}}%
9943 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9944 \newabbreviationstyle{long-noshort-em-desc}%
9945 {%
9946   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}}%
9947 }%
9948 {%
9949   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9950   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9951   \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9952   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9953   \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9954 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9955   \glsslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9956   \ifglsxtrininsertinside \else##2\fi
9957 }%
9958 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9959   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9960   \ifglsxtrininsertinside \else##2\fi
9961 }%
9962 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9963   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9964   \ifglsxtrininsertinside \else##2\fi
9965 }%
9966 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9967   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9968   \ifglsxtrininsertinside \else##2\fi
9969 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9970 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9971   \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9972   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9973   \glsxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9974 }%
9975 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

```

```

9976 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9977 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9978 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9979 }%
9980 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9981 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9982 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9983 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9984 }%
9985 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9986 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9987 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9988 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9989 }%

```

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.

```

9990 \renewcommand*{\glxtrfullformat}[2]{%
9991 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9992 \ifglxtrinsertinside\else##2\fi
9993 }%
9994 \renewcommand*{\glxtrfullplformat}[2]{%
9995 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9996 \ifglxtrinsertinside\else##2\fi
9997 }%
9998 \renewcommand*{\Glsxtrfullformat}[2]{%
9999 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10000 \ifglxtrinsertinside\else##2\fi
10001 }%
10002 \renewcommand*{\Glsxtrfullplformat}[2]{%
10003 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10004 \ifglxtrinsertinside\else##2\fi
10005 }%
10006 }

```

long-desc-em Backward compatibility:

```
10007 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```

10008 \newabbreviationstyle{long-em-noshort-em-desc}%
10009 {%
10010 \renewcommand*{\CustomAbbreviationFields}{%
10011 name={\glxtrlongnoshortdescname},
10012 sort={\the\glslongtok},
10013 first={\protect\glsfirstlongemfont{\the\glslongtok}},
10014 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10015 text={\glslongemfont{\the\glslongtok}},
10016 plural={\glslongemfont{\the\glslongpltok}}}%

```

```

10017 }%
10018 \renewcommand*\GlsXtrPostNewAbbreviation}{%
10019   \glssetattribute{\the\glslabeltok}{regular}{true}}%
10020 }%
10021 {%
10022   \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
10023   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
10024   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
10025   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
10026   \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10027 \renewcommand*\glsxtrsubsequentfmt}[2]{%
10028   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10029   \ifglsxtrininsertinside \else##2\fi
10030 }%
10031 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
10032   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10033   \ifglsxtrininsertinside \else##2\fi
10034 }%
10035 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
10036   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10037   \ifglsxtrininsertinside \else##2\fi
10038 }%
10039 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
10040   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10041   \ifglsxtrininsertinside \else##2\fi
10042 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10043 \renewcommand*\glsxtrinlinefullformat}[2]{%
10044   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10045   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10046   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10047 }%
10048 \renewcommand*\glsxtrinlinefullplformat}[2]{%
10049   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10050   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10051   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10052 }%
10053 \renewcommand*\Glsxtrinlinefullformat}[2]{%
10054   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10055   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10056   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10057 }%
10058 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
10059   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10060   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10061   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10062 }%

```

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.

```

10063 \renewcommand*{\glxtrfullformat}[2]{%
10064   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10065   \ifglxtrinsertinside\else##2\fi
10066 }%
10067 \renewcommand*{\glxtrfullplformat}[2]{%
10068   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10069   \ifglxtrinsertinside\else##2\fi
10070 }%
10071 \renewcommand*{\Glsxtrfullformat}[2]{%
10072   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10073   \ifglxtrinsertinside\else##2\fi
10074 }%
10075 \renewcommand*{\Glsxtrfullplformat}[2]{%
10076   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10077   \ifglxtrinsertinside\else##2\fi
10078 }%
10079 }

```

`t-em-desc-noreg` Like `long-em-noshort-em-desc` but doesn't set the regular attribute.

```

10080 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10081 {%
10082   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

10083 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10084   \glshasattribute{\the\glslabeltok}{regular}%
10085   {%
10086     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10087   }%
10088   {}}%
10089 }%
10090 }%
10091 {%
10092   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10093 }

```

`ort-em-footnote`

```

10094 \newabbreviationstyle{short-em-footnote}%
10095 {%
10096   \renewcommand*{\CustomAbbreviationFields}{%
10097     name={\glxtrfootnotename},
10098     sort={\the\glsshorttok},
10099     description={\the\glslongtok},%
10100     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10101       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10102       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
10103     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%

```

```

10104 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10105 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10106 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

10107 \renewcommand*\GlsXtrPostNewAbbreviation{%
10108 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
10109 \glsattribute{\the\glslabeltok}{regular}%
10110 {%
10111 \glssetattribute{\the\glslabeltok}{regular}{false}%
10112 }%
10113 {}%
10114 }%
10115 }%
10116 {%
10117 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
10118 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
10119 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
10120 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
10121 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10122 \renewcommand*\glxtrfullformat[2]{%
10123 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10124 \ifglxtrininsertinside\else##2\fi
10125 \protect\glxtrabbrvfootnote{##1}%
10126 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10127 }%
10128 \renewcommand*\glxtrfullplformat[2]{%
10129 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10130 \ifglxtrininsertinside\else##2\fi
10131 \protect\glxtrabbrvfootnote{##1}%
10132 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10133 }%
10134 \renewcommand*\Glsxtrfullformat[2]{%
10135 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10136 \ifglxtrininsertinside\else##2\fi
10137 \protect\glxtrabbrvfootnote{##1}%
10138 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10139 }%
10140 \renewcommand*\Glsxtrfullplformat[2]{%
10141 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10142 \ifglxtrininsertinside\else##2\fi
10143 \protect\glxtrabbrvfootnote{##1}%
10144 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10145 }%

```

The first use full form and the inline full form use the short (long) style.

```

10146 \renewcommand*\glxtrininlinefullformat[2]{%
10147 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```



```

10148     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10149     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10150 }%
10151 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10152     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10153     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10154     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10155 }%
10156 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10157     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10158     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10159     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10160 }%
10161 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10162     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10163     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10164     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10165 }%
10166 }

```

footnote-em Backward compatibility:

```

10167 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

10168 \newabbreviationstyle{short-em-postfootnote}%
10169 {%
10170     \renewcommand*{\CustomAbbreviationFields}{%
10171         name={\glxtrfootnotename},
10172         sort={\the\glssshorttok},
10173         description={\the\glslongtok},%
10174         first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},%
10175         firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},%
10176         plural={\protect\glssabbrvemfont{\the\glssshortpltok}}}%

```

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.

```

10177 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10178     \csdef{glxtrpostlink\glscategorylabel}{%
10179         \glxtrifwasfirstuse
10180         {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

10181         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10182             {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
10183         }%
10184     }%
10185 }%
10186 \glshasattribute{\the\glslabeltok}{regular}%
10187 {%

```

```

10188 \glsssetattribute{\the\glslabelltok}{regular}{false}%
10189 }%
10190 {}%
10191 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

10192 \renewcommand*\glxtrsetupfulldefs{%
10193 \let\glxtrifwasfirstuse\@secondoftwo
10194 }%
10195 }%
10196 {%
10197 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
10198 \renewcommand*\glssabbrvfont[1]{\glssabbrvemfont{##1}}%
10199 \renewcommand*\glssfirstabbrvfont[1]{\glssfirstabbrvemfont{##1}}%
10200 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongfootnotefont{##1}}%
10201 \renewcommand*\glsslongfont[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

10202 \renewcommand*\glxtrfullformat[2]{%
10203 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10204 \ifglxtrininsertinside\else##2\fi
10205 }%
10206 \renewcommand*\glxtrfullplformat[2]{%
10207 \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10208 \ifglxtrininsertinside\else##2\fi
10209 }%
10210 \renewcommand*\Glsxtrfullformat[2]{%
10211 \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10212 \ifglxtrininsertinside\else##2\fi
10213 }%
10214 \renewcommand*\Glsxtrfullplformat[2]{%
10215 \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10216 \ifglxtrininsertinside\else##2\fi
10217 }%

```

The first use full form and the inline full form use the short (long) style.

```

10218 \renewcommand*\glxtrininlinefullformat[2]{%
10219 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10220 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10221 \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10222 }%
10223 \renewcommand*\glxtrininlinefullplformat[2]{%
10224 \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10225 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10226 \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10227 }%
10228 \renewcommand*\Glsxtrininlinefullformat[2]{%
10229 \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10230 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10231 \glxtrparen{\glssfirstlongfootnotefont{\Glsaccesslong{##1}}}%

```

```

10232 }%
10233 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10234   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10235   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10236   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10237 }%
10238 }

```

postfootnote-em Backward compatibility:

```

10239 \@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.

```

10240 \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.

```

10241 \ifdef\glscurrentfieldvalue
10242 {
10243   \newcommand*{\glxtruserparen}[2]{%
10244     \glsxtrfullsep{#2}%
10245     \glsxtrparen
10246       {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}}}%
10247   }
10248 }
10249 {
10250   \newcommand*{\glxtruserparen}[2]{%
10251     \glsxtrfullsep{#2}%
10252     \glsxtrparen
10253       {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}}}%
10254   }
10255 }

```

Font used for short form:

lsabbrvuserfont

```

10256 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

10257 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
10258 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10259 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10260 \newcommand*{\lsxtrusersuffix}{\lsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10261 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10262 \newabbreviationstyle{long-short-user}%
10263 {%
10264   \renewcommand*{\CustomAbbreviationFields}{%
10265     name={\glstrlongshortname},
10266     sort={\the\glsshorttok},
10267     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10268       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10269     {\the\glslabeltok}},%
10270     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10271       \protect\glxtruserparen
10272       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10273     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10274     description={\protect\glsuserdescription{\the\glslongtok}%
10275       {\the\glslabeltok}}}%
10276   }
```

Unset the regular attribute if it has been set.

```
10276   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10277     \glshasattribute{\the\glslabeltok}{regular}%
10278     {%
10279       \glissetattribute{\the\glslabeltok}{regular}{false}%
10280     }%
10281   }%
10282 }%
10283 }%
10284 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10285 \renewcommand*{\abbrvpluralsuffix}{\lsxtrusersuffix}%
10286 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10287 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10288 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10289 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline form are the same for this style.

```

10290 \renewcommand*{\glxtrfullformat}[2]{%
10291   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10292   \ifglxtrinsertinside\else##2\fi
10293   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10294 }%
10295 \renewcommand*{\glxtrfullplformat}[2]{%
10296   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10297   \ifglxtrinsertinside\else##2\fi
10298   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10299 }%
10300 \renewcommand*{\Glsxtrfullformat}[2]{%
10301   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10302   \ifglxtrinsertinside\else##2\fi
10303   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10304 }%
10305 \renewcommand*{\Glsxtrfullplformat}[2]{%
10306   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10307   \ifglxtrinsertinside\else##2\fi
10308   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10309 }%
10310 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10311 \newabbreviationstyle{long-postshort-user}%
10312 {%
10313   \renewcommand*{\CustomAbbreviationFields}{%
10314     name={\glxtrlongshortname},
10315     sort={\the\glsshorttok},
10316     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10317     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10318     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10319     description={\protect\glsuserdescription{\the\glslongtok}%
10320       {\the\glslabeltok}}}%
10321   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10322     \csdef{glxtrpostlink\glscategorylabel}{%
10323       \glxtrifwasfirstuse
10324       {%
10325         \glxtruserparen
10326         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10327         {\glslabel}%
10328       }%
10329     }%
10330   }%
10331   \glsasattribute{\the\glslabeltok}{regular}%
10332   {%
10333     \glssetattribute{\the\glslabeltok}{regular}{false}%
10334   }%

```

```

10335     {}%
10336   }%
10337 }%
10338 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10339 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10340 \renewcommand*{\glsabrvfont}[1]{\glsabrvuserfont{##1}}%
10341 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10342 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10343 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10344 \renewcommand*{\glxtrfullformat}[2]{%
10345   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10346   \ifglxtrininsertinside\else##2\fi
10347 }%
10348 \renewcommand*{\glxtrfullplformat}[2]{%
10349   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10350   \ifglxtrininsertinside\else##2\fi
10351 }%
10352 \renewcommand*{\Glsxtrfullformat}[2]{%
10353   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10354   \ifglxtrininsertinside\else##2\fi
10355 }%
10356 \renewcommand*{\Glsxtrfullplformat}[2]{%
10357   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10358   \ifglxtrininsertinside\else##2\fi
10359 }%

```

In-line format:

```

10360 \renewcommand*{\glxtrinlinefullformat}[2]{%
10361   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10362   \ifglxtrininsertinside\else##2\fi
10363   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10364 }%
10365 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10366   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10367   \ifglxtrininsertinside\else##2\fi
10368   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10369 }%
10370 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10371   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10372   \ifglxtrininsertinside\else##2\fi
10373   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10374 }%
10375 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10376   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10377   \ifglxtrininsertinside\else##2\fi
10378   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%

```

```

10379 }%
10380 }

```

ortuserdescname

```

10381 \newcommand*{\glxtrlongshortuserdescname}{%
10382   \protect\glslonguserfont{\the\glslongtok}%
10383   \protect\glxtruserparen
10384   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10385 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10386 \newabbreviationstyle{long-postshort-user-desc}%
10387 {%
10388   \renewcommand*{\CustomAbbreviationFields}{%
10389     name={\glxtrlongshortuserdescname},
10390     sort={\the\glslongtok},
10391     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10392     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10393     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10394     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10395 }%
10396 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10397   \csdef{glxtrpostlink\glscategorylabel}{%
10398     \glxtrifwasfirstuse
10399     {%
10400       \glxtruserparen
10401       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10402       {\glslabel}%
10403     }%
10404     {}%
10405   }%
10406   \glshasattribute{\the\glslabeltok}{regular}%
10407   {%
10408     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10409   }%
10410   {}%
10411 }%
10412 }%
10413 {%
10414   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10415 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10416 \newabbreviationstyle{short-postlong-user}%
10417 {%
10418   \renewcommand*{\CustomAbbreviationFields}{%
10419     name={\glxtrshortlongname},
10420     sort={\the\glsshorttok},

```

```

10421 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10422 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10423 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10424 description={\protect\glsuserdescription{\the\glslongtok}%
10425 {\the\glslabeltok}}}%
10426 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10427 \csdef{glsxtrpostlink\glscategorylabel}{%
10428 \glsxtrifwasfirstuse
10429 {%
10430 \glsxtruserparen
10431 {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10432 {\glslabel}}%
10433 }%
10434 {}%
10435 }%
10436 \glsattribute{\the\glslabeltok}{regular}%
10437 {%
10438 \glssetattribute{\the\glslabeltok}{regular}{false}%
10439 }%
10440 {}%
10441 }%
10442 }%
10443 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10444 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10445 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10446 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10447 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10448 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10449 \renewcommand*{\glsxtrfullformat}[2]{%
10450 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10451 \ifglsxtrininsertinside\else##2\fi
10452 }%
10453 \renewcommand*{\glsxtrfullplformat}[2]{%
10454 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10455 \ifglsxtrininsertinside\else##2\fi
10456 }%
10457 \renewcommand*{\Glsxtrfullformat}[2]{%
10458 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10459 \ifglsxtrininsertinside\else##2\fi
10460 }%
10461 \renewcommand*{\Glsxtrfullplformat}[2]{%
10462 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10463 \ifglsxtrininsertinside\else##2\fi
10464 }%

```

In-line format:


```

10465 \renewcommand*{\glxtrinlinefullformat}[2]{%
10466   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10467   \ifglxtrininsertinside\else##2\fi
10468   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10469 }%
10470 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10471   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10472   \ifglxtrininsertinside\else##2\fi
10473   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10474 }%
10475 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10476   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10477   \ifglxtrininsertinside\else##2\fi
10478   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10479 }%
10480 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10481   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10482   \ifglxtrininsertinside\else##2\fi
10483   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10484 }%
10485 }

```

onguserdesname

```

10486 \newcommand*{\glxtrshortlonguserdesname}{%
10487   \protect\glssabbrvuserfont{\the\glssshorttok}%
10488   \protect\glxtruserparen
10489     {\protect\glslonguserfont{\the\glslongpltok}}}%
10490     {\the\glslabelltok}%
10491 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10492 \newabbreviationstyle{short-postlong-user-desc}%
10493 {%
10494   \renewcommand*{\CustomAbbreviationFields}{%
10495     name={\glxtrshortlonguserdesname},
10496     sort={\the\glssshorttok},
10497     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10498     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10499     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10500     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10501 }%
10502 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10503   \csdef{glxtrpostlink\glscategorylabel}{%
10504     \glxtrifwasfirstuse
10505     {%
10506       \glxtruserparen
10507         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10508         {\glslabel}%

```

```

10509     }%
10510     {}%
10511     }%
10512     \glshasattribute{\the\glslabeltok}{regular}%
10513     {%
10514         \glissetattribute{\the\glslabeltok}{regular}{false}%
10515     }%
10516     {}%
10517 }%
10518 }%
10519 {%
10520     \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10521 }

```

short-user-desc

```

10522 \newabbreviationstyle{long-short-user-desc}%
10523 {%
10524     \renewcommand*{\CustomAbbreviationFields}{%
10525         name={\glsxtrlongshortuserdescname},
10526         sort={\glsxtrlongshortdescsort},%

10527         first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10528             \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10529             {\the\glslabeltok}},%
10530         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10531             \protect\glsxtruserparen
10532             {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10533         text={\protect\glsabbrvfont{\the\glsshorttok}},%
10534         plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10535     }%

```

Unset the regular attribute if it has been set.

```

10536     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10537         \glshasattribute{\the\glslabeltok}{regular}%
10538         {%
10539             \glissetattribute{\the\glslabeltok}{regular}{false}%
10540         }%
10541         {}%
10542     }%
10543 }%
10544 {%
10545     \GlsXtrUseAbbrStyleFmts{long-short-user}%
10546 }

```

short-long-user

```

10547 \newabbreviationstyle{short-long-user}%
10548 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10549 \renewcommand*{\CustomAbbreviationFields}{%
10550     name={\glxtrshortlongname},
10551     sort={\the\glsshorttok},
10552     description={\protect\gluserdescription{\the\glslongtok}%
10553         {\the\glslabeltok}},%
10554     first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10555         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}%
10556         {\the\glslabeltok}},%
10557     firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10558         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}%
10559         {\the\glslabeltok}},%
10560     plural={\protect\glabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10561 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10562     \glshasattribute{\the\glslabeltok}{regular}%
10563     {%
10564         \glissetattribute{\the\glslabeltok}{regular}{false}%
10565     }%
10566     {}%
10567 }%
10568 }%
10569 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10570 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10571 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
10572 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvuserfont{##1}}%
10573 \renewcommand*{\glfirstlongfont}[1]{\glfirstlonguserfont{##1}}%
10574 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10575 \renewcommand*{\glxtrfullformat}[2]{%
10576     \glfirstabbrvuserfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10577     \ifglxtrininsertinside\else##2\fi
10578     \glxtruserparen{\glfirstlonguserfont{\glaccesslong{##1}}}{##1}%
10579 }%
10580 \renewcommand*{\glxtrfullplformat}[2]{%
10581     \glfirstabbrvuserfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10582     \ifglxtrininsertinside\else##2\fi
10583     \glxtruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10584 }%
10585 \renewcommand*{\Glsxtrfullformat}[2]{%
10586     \glfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10587     \ifglxtrininsertinside\else##2\fi
10588     \glxtruserparen{\glfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10589 }%
10590 \renewcommand*{\Glsxtrfullplformat}[2]{%
10591     \glfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10592     \ifglxtrininsertinside\else##2\fi

```

```

10593   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10594 }%
10595 }

```

-long-user-desc

```

10596 \newabbreviationstyle{short-long-user-desc}%
10597 {%
10598   \renewcommand*{\CustomAbbreviationFields}{%
10599     name={\glxtrshortlonguserdescname},
10600     sort={\glxtrshortlongdescsort},%

10601     first={\protect\glsfirstabbrvuserfont{\the\glssshorttok}%
10602       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10603       {\the\glslabeltok}},%
10604     firstplural={\protect\glsfirstabbrvuserfont{\the\glssshortpltok}%
10605       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10606       {\the\glslabeltok}},%
10607     text={\protect\glsabbrvfont{\the\glssshorttok}},%
10608     plural={\protect\glsabbrvfont{\the\glssshortpltok}}}%
10609 }%

```

Unset the regular attribute if it has been set.

```

10610 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10611   \glshasattribute{\the\glslabeltok}{regular}%
10612   {%
10613     \glissetattribute{\the\glslabeltok}{regular}{false}%
10614   }%
10615   {}%
10616 }%
10617 }%
10618 {%
10619   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10620 }

```

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.

```

10621 \newrobustcmd*{\glxtrifhyphenstart}[3]{%
10622   \ifx\glsinsert#1\relax
10623     \expandafter\@glxtrifhyphenstart#1\relax\relax
10624     \@end@glxtrifhyphenstart{#2}{#3}%
10625   \else
10626     \@glxtrifhyphenstart#1\relax\relax\@end@glxtrifhyphenstart{#2}{#3}%

```

```
10627 \fi
10628 }
```

trifhyphenstart

```
10629 \def\@glxtrifhyphenstart#1#2\end@glxtrifhyphenstart#3#4{%
10630 \ifx-#1\relax#3\else #4\fi
10631 }
```

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.

```
10632 \newcommand*\glxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10633 {%
```

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.

```
10634 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
10635 \glxfirstlonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10636 \ifglxtrininsertinside\else{#4}\fi
10637 \glxtrfullsep{#1}%
10638 \glxtrparen{\glxfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10639 \ifglxtrininsertinside\else{#4}\fi}%
10640 }%
10641 }
```

abbrvhyphenfont

```
10642 \newcommand*\glxabbrvhyphenfont{\glxabbrvdefaultfont}%
```

abbrvhyphenfont

```
10643 \newcommand*\glxfirstabbrvhyphenfont{\glxabbrvhyphenfont}%
```

slonghyphenfont

```
10644 \newcommand*\glxslonghyphenfont{\glxslongdefaultfont}%
```

tlonghyphenfont

```
10645 \newcommand*\glxfirstlonghyphenfont{\glxslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10646 \newcommand*\glxtrhyphensuffix{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the keywords attribute.

```

10647 \newabbreviationstyle{long-hyphen-short-hyphen}%
10648 {%
10649   \renewcommand*{\CustomAbbreviationFields}{%
10650     name={\glxtrlongshortname},
10651     sort={\the\glsshorttok},
10652     first={\protect\glxtrfirstlonghyphenfont{\the\glslongtok}%
10653       \protect\glxtrfullsep{\the\glslabeltok}%
10654       \glxtrparen{\protect\glxtrfirstabbrvhyphenfont{\the\glsshorttok}}},%
10655     firstplural={\protect\glxtrfirstlonghyphenfont{\the\glslongpltok}%
10656       \protect\glxtrfullsep{\the\glslabeltok}%
10657       \glxtrparen{\protect\glxtrfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10658     plural={\protect\glxtrabbrvhyphenfont{\the\glsshortpltok}}},%
10659     description={\protect\glxtrlonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10660 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10661   \glshasattribute{\the\glslabeltok}{regular}%
10662   {%
10663     \glissetattribute{\the\glslabeltok}{regular}{false}%
10664   }%
10665   {}%
10666 }%
10667 }%
10668 {%
10669   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10670   \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvhyphenfont{##1}}%
10671   \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvhyphenfont{##1}}%
10672   \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlonghyphenfont{##1}}%
10673   \renewcommand*{\glxtrlongfont}[1]{\glxtrlonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10674 \renewcommand*{\glxtrfullformat}[2]{%
10675   \glxtrlonghyphenshort{##1}{\glxtraccesslong{##1}}{\glxtraccessshort{##1}}{##2}%
10676 }%
10677 \renewcommand*{\glxtrfullplformat}[2]{%
10678   \glxtrlonghyphenshort{##1}{\glxtraccesslongpl{##1}}%
10679   {\glxtraccessshortpl{##1}}{##2}%
10680 }%
10681 \renewcommand*{\Glsxtrfullformat}[2]{%
10682   \glxtrlonghyphenshort{##1}{\glxtraccesslong{##1}}{\glxtraccessshort{##1}}{##2}%
10683 }%
10684 \renewcommand*{\Glsxtrfullplformat}[2]{%
10685   \glxtrlonghyphenshort{##1}{\glxtraccesslongpl{##1}}%
10686   {\glxtraccessshortpl{##1}}{##2}%
10687 }%
10688 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10689 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%

```

```

10690 {%
10691   \renewcommand*{\CustomAbbreviationFields}{%
10692     name={\glxtrlongshortdescname},
10693     sort={\glxtrlongshortdescsort},
10694     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10695       \protect\glxtrfullsep{\the\glslabeltok}%
10696       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10697     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10698       \protect\glxtrfullsep{\the\glslabeltok}%
10699       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10700     text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
10701     plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
10702   }%

```

Unset the regular attribute if it has been set.

```

10703   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10704     \glshasattribute{\the\glslabeltok}{regular}%
10705     {%
10706       \glissetattribute{\the\glslabeltok}{regular}{false}%
10707     }%
10708   }%
10709 }%
10710 }%
10711 {%
10712   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10713 }

```

onghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```

10714 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10715 {%

```

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.

```

10716   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10717     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10718     \ifglxtrininsertinsideelse{#3}\fi
10719   }%
10720 }

```

short-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.

```

10721 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10722 {%
10723   \renewcommand*{\CustomAbbreviationFields}{%
10724     name={\glxtrlongnoshortdescname},
10725     sort={\expandonce\glxtrorglong},
10726     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10727     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10728     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10729   }%

```

Unset the regular attribute if it has been set.

```

10730 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10731   \glshasattribute{\the\glslabeltok}{regular}%
10732   {%
10733     \glissetattribute{\the\glslabeltok}{regular}{false}%
10734   }%
10735   {}}%
10736 }%
10737 }%
10738 {%
10739   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10740 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10741 \renewcommand*{\glssabrvfont[1]}{\glssabrvdefaultfont{##1}}%
10742 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10743 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10744 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10745 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10746   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10747 }%
10748 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10749   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10750 }%
10751 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10752   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10753 }%
10754 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10755   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10756 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10757 \renewcommand*{\glxtrininlinefullformat}[2]{%
10758   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10759   \glxtrfullsep{##1}%
10760   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10761 }%
10762 \renewcommand*{\glxtrininlinefullplformat}[2]{%
10763   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%

```



```

10764 \glstrfullsep{##1}%
10765 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10766 }%
10767 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10768 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10769 \glstrfullsep{##1}%
10770 \glstrparen{\protect\glfirstabbrvfont{\glaccessshort{##1}}}%
10771 }%
10772 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10773 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10774 \glstrfullsep{##1}%
10775 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10776 }%

```

The first use full form only displays the long form.

```

10777 \renewcommand*{\glstrfullformat}[2]{%
10778 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10779 }%
10780 \renewcommand*{\glstrfullplformat}[2]{%
10781 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10782 }%
10783 \renewcommand*{\Glsxtrfullformat}[2]{%
10784 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10785 }%
10786 \renewcommand*{\Glsxtrfullplformat}[2]{%
10787 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10788 }%
10789 }

```

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.

```

10790 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10791 {%
10792 \renewcommand*{\CustomAbbreviationFields}{%
10793 name={\glstrlongnoshortname},
10794 sort={\the\glsshorttok},
10795 first={\protect\glfirstlonghyphenfont{\the\glslongtok}},%
10796 firstplural={\protect\glfirstlonghyphenfont{\the\glslongpltok}},%
10797 text={\protect\glslonghyphenfont{\the\glslongtok}},%
10798 plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10799 description={\the\glslongtok}%
10800 }%

```

Unset the regular attribute if it has been set.

```

10801 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10802 \glshasattribute{\the\glslabeltok}{regular}%
10803 {%
10804 \glissetattribute{\the\glslabeltok}{regular}{false}%
10805 }%

```

```

10806     {}%
10807   }%
10808 }%
10809 {%
10810   \GlsXtrUseAbbrStyleFmts{long-desc}%
10811 }

```

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.

```

10812 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10813   {%
10814     \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10815     \glsfirstlonghyphenfont{#1}%
10816   }%
10817 }

```

rposthyphenshort `\glsxtrposthyphenshort{<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.

```

10818 \newcommand*{\glsxtrposthyphenshort}[2]{%
10819   {%
10820     \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10821     \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10822     \glsxtrfullsep{#1}%
10823     \glsxtrparen
10824     {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10825     \ifglsxtrininsertinside\else{#2}\fi
10826   }%
10827 }%
10828 }

```

hyphensequent `\glsxtrposthyphensequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10829 \newcommand*{\glsxtrposthyphensequent}[2]{%
10830   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%

```

```

10831 \ifglxtrinsertinside \else{#2}\fi
10832 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10833 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10834 {%
10835   \renewcommand*{\CustomAbbreviationFields}{%
10836     name={\glxtrlongshortname},
10837     sort={\the\glsshorttok},
10838     first={\protect\glxtrfirstlonghyphenfont{\the\glslongtok}},%
10839     firstplural={\protect\glxtrfirstlonghyphenfont{\the\glslongpltok}},%
10840     plural={\protect\glxtrabbrvhyphenfont{\the\glsshortpltok}},%
10841     description={\protect\glxtrlonghyphenfont{\the\glslongtok}}}%
10842   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10843     \csdef{glxtrpostlink\glscategorylabel}{%
10844       \glxtrifwasfirstuse
10845       {%
10846         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10847       }%
10848     }%

```

Put the insertion into the post-link:

```

10849       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10850     }%
10851   }%
10852   \glshasattribute{\the\glslabeltok}{regular}%
10853   {%
10854     \glissetattribute{\the\glslabeltok}{regular}{false}%
10855   }%
10856   {}%
10857 }%
10858 }%
10859 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10860 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10861 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10862 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10863 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10864 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10865 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10866   \glsabbrvfont{\glsaccessshort{##1}}%
10867 }%
10868 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10869   \glsabbrvfont{\glsaccessshortpl{##1}}%
10870 }%
10871 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

10872 \glsabbrvfont{\Glsaccessshort{##1}}%
10873 }%
10874 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10875 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10876 }%

First use full form:
10877 \renewcommand*{\glsxtrfullformat}[2]{%
10878 \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10879 }%
10880 \renewcommand*{\glsxtrfullplformat}[2]{%
10881 \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10882 }%
10883 \renewcommand*{\Glsxtrfullformat}[2]{%
10884 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10885 }%
10886 \renewcommand*{\Glsxtrfullplformat}[2]{%
10887 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10888 }%

```

In-line format.

```

10889 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10890 \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10891 \ifglsxtrininsertinside{##2}\fi}%
10892 \ifglsxtrininsertinside \else{##2}\fi
10893 }%
10894 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10895 \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10896 \ifglsxtrininsertinside{##2}\fi}%
10897 \ifglsxtrininsertinside \else{##2}\fi
10898 }%
10899 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10900 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10901 \ifglsxtrininsertinside{##2}\fi}%
10902 \ifglsxtrininsertinside \else{##2}\fi
10903 }%
10904 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10905 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10906 \ifglsxtrininsertinside{##2}\fi}%
10907 \ifglsxtrininsertinside \else{##2}\fi
10908 }%
10909 }

```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10910 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10911 {%
10912 \renewcommand*{\CustomAbbreviationFields}{%
10913 name={\glsxtrlongshortdescname},
10914 sort={\glsxtrlongshortdescsort},%
10915 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%

```

```

10916 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10917 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10918 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10919 }%
10920 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10921 \csdef{glxtrpostlink\glscategorylabel}{%
10922 \glxtrifwasfirstuse
10923 {%
10924 \glxtrposthyphenshort{\glslabel}{\glsinsert}}%
10925 }%
10926 {%

```

Put the insertion into the post-link:

```

10927 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
10928 }%
10929 }%
10930 \glshasattribute{\the\glslabeltok}{regular}%
10931 {%
10932 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10933 }%
10934 {}%
10935 }%
10936 }%
10937 {%
10938 \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10939 }

```

rshorthyphenlong

```
\glxtrshorthyphenlong{<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.

```
10940 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10941 {%
```

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.)

```

10942 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}}%
10943 \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10944 \ifglxtrininsertinside\else{#4}\fi
10945 \glxtrfullsep{#1}%
10946 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10947 \ifglxtrininsertinside\else{#4}\fi}%
10948 }%
10949 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```
10950 \newabbreviationstyle{short-hyphen-long-hyphen}%
10951 {%
10952   \renewcommand*{\CustomAbbreviationFields}{%
10953     name={\glxtrshortlongname},
10954     sort={\the\glsshorttok},
10955     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10956       \protect\glxtrfullsep{\the\glslabeltok}%
10957       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10958     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10959       \protect\glxtrfullsep{\the\glslabeltok}%
10960       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10961     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10962     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10963 }
```

Unset the regular attribute if it has been set.

```
10963 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10964   \glshasattribute{\the\glslabeltok}{regular}%
10965   {%
10966     \glissetattribute{\the\glslabeltok}{regular}{false}%
10967   }%
10968 }%
10969 }%
10970 }%
10971 {%
10972   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10973   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10974   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10975   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10976   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
10977 }
```

The first use full form and the inline full form are the same for this style.

```
10977 \renewcommand*{\glxtrfullformat}[2]{%
10978   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10979 }%
10980 \renewcommand*{\glxtrfullplformat}[2]{%
10981   \glxtrshorthyphenlong{##1}%
10982   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10983 }%
10984 \renewcommand*{\Glsxtrfullformat}[2]{%
10985   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10986 }%
10987 \renewcommand*{\Glsxtrfullplformat}[2]{%
10988   \glxtrshorthyphenlong{##1}%
10989   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10990 }%
10991 }
```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```
10992 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10993 }
```

```

10993 {%
10994   \renewcommand*{\CustomAbbreviationFields}{%
10995     name={\glxtrshortlongdescname},
10996     sort={\glxtrshortlongdescsort},
10997     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10998       \protect\glxtrfullsep{\the\glslabeltok}%
10999       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11000     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
11001       \protect\glxtrfullsep{\the\glslabeltok}%
11002       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11003     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11004     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
11005   }%

```

Unset the regular attribute if it has been set.

```

11006   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11007     \glshasattribute{\the\glslabeltok}{regular}%
11008     {%
11009       \glissetattribute{\the\glslabeltok}{regular}{false}%
11010     }%
11011   }%
11012 }%
11013 }%
11014 {%
11015   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11016 }

```

`\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.

```
11017 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

11018 {%
11019   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}}%
11020   \glsfirstabbrvhyphenfont{#1}%
11021 }%
11022 }

```

`\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.

```

11023 \newcommand*{\glxtrposthyphenlong}[2]{%
11024 {%
11025   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11026   \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11027   \glxtrfullsep{#1}%
11028   \glxtrparen
11029   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
11030   \ifglxtrininsertinside\else{#2}\fi
11031 }%
11032 }%
11033 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

11034 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11035 {%
11036   \renewcommand*{\CustomAbbreviationFields}{%
11037     name={\glxtrshortlongname},
11038     sort={\the\glsshorttok},
11039     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11040     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11041     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
11042     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11043   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11044     \csdef{glxtrpostlink\glscategorylabel}{%
11045       \glxtrifwasfirstuse
11046       {%
11047         \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11048       }%
11049       {%

```

Put the insertion into the post-link:

```

11050       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11051     }%
11052   }%
11053   \glshasattribute{\the\glslabeltok}{regular}%
11054   {%
11055     \glissetattribute{\the\glslabeltok}{regular}{false}%
11056   }%
11057   {}%
11058 }%
11059 }%
11060 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11061 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11062 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
11063 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11064 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11065 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```


Subsequent use needs to omit the insertion:

```

11066 \renewcommand*{\glxtrsubsequentfmt}[2]{%
11067   \glssabrvfont{\glssaccessshort{##1}}%
11068 }%
11069 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11070   \glssabrvfont{\glssaccessshortpl{##1}}%
11071 }%
11072 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11073   \glssabrvfont{\Glsaccessshort{##1}}%
11074 }%
11075 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11076   \glssabrvfont{\Glsaccessshortpl{##1}}%
11077 }%

```

First use full form:

```

11078 \renewcommand*{\glxtrfullformat}[2]{%
11079   \glxtrshorthyphen{\glssaccessshort{##1}}{##1}{##2}%
11080 }%
11081 \renewcommand*{\glxtrfullplformat}[2]{%
11082   \glxtrshorthyphen{\glssaccessshortpl{##1}}{##1}{##2}%
11083 }%
11084 \renewcommand*{\Glsxtrfullformat}[2]{%
11085   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
11086 }%
11087 \renewcommand*{\Glsxtrfullplformat}[2]{%
11088   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
11089 }%

```

In-line format. Commands like `\glxtrfull` set `\glinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

11090 \renewcommand*{\glxtrinlinefullformat}[2]{%
11091   \glsfirstabrvhyphenfont{\glssaccessshort{##1}}%
11092   \ifglxtrininsertinside{##2}\fi}%
11093   \ifglxtrininsertinside \else{##2}\fi
11094 }%
11095 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11096   \glsfirstabrvhyphenfont{\glssaccessshortpl{##1}}%
11097   \ifglxtrininsertinside{##2}\fi}%
11098   \ifglxtrininsertinside \else{##2}\fi
11099 }%
11100 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11101   \glsfirstabrvhyphenfont{\Glsaccessshort{##1}}%
11102   \ifglxtrininsertinside{##2}\fi}%
11103   \ifglxtrininsertinside \else{##2}\fi
11104 }%
11105 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11106   \glsfirstabrvhyphenfont{\Glsaccessshortpl{##1}}%
11107   \ifglxtrininsertinside{##2}\fi}%
11108   \ifglxtrininsertinside \else{##2}\fi
11109 }%

```

```
11110 }
```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
11111 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
11112 {%
11113   \renewcommand*{\CustomAbbreviationFields}{%
11114     name={\glstrshortlongdescname},
11115     sort={\glstrshortlongdescsort},%
11116     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11117     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11118     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11119     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11120 }%
11121 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11122   \csdef{glstrpostlink\glscategorylabel}{%
11123     \glstrifwasfirstuse
11124     {%
11125       \glstrposthyphenlong{\glslabel}{\glsinsert}%
11126     }%
11127     {%
```

Put the insertion into the post-link:

```
11128       \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
11129     }%
11130   }%
11131   \glshasattribute{\the\glslabeltok}{regular}%
11132   {%
11133     \glissetattribute{\the\glslabeltok}{regular}{false}%
11134   }%
11135   {}%
11136 }%
11137 }%
11138 {%
11139   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
11140 }
```

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

```
11141 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
11142 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
11143 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
11144 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
11145 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
11146 \newcommand*{\glsxtronlyname}{%
```

```
11147 \protect\glabbrvonlyfont{\the\glsshorttok}}%
```

```
11148 }
```

only-short-only

```
11149 \newabbreviationstyle{long-only-short-only}%
```

```
11150 {%
```

```
11151 \renewcommand*{\CustomAbbreviationFields}{%
```

```
11152 name={\glsxtronlyname},
```

```
11153 sort={\the\glsshorttok},
```

```
11154 first={\protect\glfirstlongonlyfont{\the\glslongtok}},%
```

```
11155 firstplural={\protect\glfirstlongonlyfont{\the\glslongpltok}},%
```

```
11156 plural={\protect\glabbrvonlyfont{\the\glsshortpltok}},%
```

```
11157 description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
11158 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
11159 \glshasattribute{\the\glslabeltok}{regular}}%
```

```
11160 {%
```

```
11161 \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
11162 }%
```

```
11163 {}%
```

```
11164 }%
```

```
11165 }%
```

```
11166 {%
```

```
11167 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
```

```
11168 \renewcommand*{\glabbrvfont}[1]{\glabbrvonlyfont{##1}}%
```

```
11169 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvonlyfont{##1}}%
```

```
11170 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongonlyfont{##1}}%
```

```
11171 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
11172 \renewcommand*{\glsxtrfullformat}[2]{%
```

```
11173 \glfirstlongonlyfont{\glaccesslong{##1}\ifglsxtrinsertinside##2\fi}}%
```

```
11174 \ifglsxtrinsertinside\else##2\fi
```

```
11175 }%
```

```
11176 \renewcommand*{\glsxtrfullplformat}[2]{%
```

```
11177 \glfirstlongonlyfont{\glaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}}%
```

```
11178 \ifglsxtrinsertinside\else##2\fi
```

```
11179 }%
```

```
11180 \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

11181 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11182 \ifglxtrinsertinside\else##2\fi
11183 }%
11184 \renewcommand*{\Glsxtrfullplformat}[2]{%
11185 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11186 \ifglxtrinsertinside\else##2\fi
11187 }%

```

The inline full form does show the short form.

```

11188 \renewcommand*{\glxtrinlinefullformat}[2]{%
11189 \glsfirstlongonlyfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11190 \ifglxtrinsertinside\else##2\fi
11191 \glxtrfullsep{##1}%
11192 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshort{##1}}}%
11193 }%
11194 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11195 \glsfirstlongonlyfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11196 \ifglxtrinsertinside\else##2\fi
11197 \glxtrfullsep{##1}%
11198 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
11199 }%
11200 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11201 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11202 \ifglxtrinsertinside\else##2\fi
11203 \glxtrfullsep{##1}%
11204 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
11205 }%
11206 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11207 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11208 \ifglxtrinsertinside\else##2\fi
11209 \glxtrfullsep{##1}%
11210 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11211 }%
11212 }

```

xtonlydescsort

```

11213 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtonlydescname

```

11214 \newcommand*{\glxtronlydescname}{%
11215 \protect\glslongfont{\the\glslongtok}%
11216 }

```

short-only-desc

```

11217 \newabbreviationstyle{long-only-short-only-desc}%
11218 {%
11219 \renewcommand*{\CustomAbbreviationFields}{%
11220 name={\glxtronlydescname},
11221 sort={\glxtronlydescsort},%

```

```

11222   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11223   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11224   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
11225   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
11226 }%

```

Unset the regular attribute if it has been set.

```

11227 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11228   \glshasattribute{\the\glslabeltok}{regular}}%
11229 {%
11230   \glssetattribute{\the\glslabeltok}{regular}{false}}%
11231 }%
11232 {}%
11233 }%
11234 }%
11235 {%
11236   \GlsXtrUseAbbrStyleFmts{long-only-short-only}}%
11237 }

```

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:

```

11238 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```
11239 \renewcommand*{\markright}[1]{%
11240   \glxtrmarkhook
11241   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
11242   \glxtrrestoremarkhook
11243 }
```

\markboth Save original definition:

```
11244 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
11245 \renewcommand*{\markboth}[2]{%
11246   \glxtrmarkhook
11247   \@glxtr@org@markboth
11248   {\@glxtrinmark#1\@glxtrnotinmark}%
11249   {\@glxtrinmark#2\@glxtrnotinmark}%
11250   \glxtrrestoremarkhook
11251 }
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
11252 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
11253 \renewcommand*{\@starttoc}[1]{%
11254   \glxtrmarkhook
11255   \@glxtrinmark
11256   \@glxtr@org@@starttoc{#1}%
11257   \@glxtrnotinmark
11258   \glxtrrestoremarkhook
11259 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

glxtrRevertMarks

```
11260 \newcommand*{\glxtrRevertMarks}{%
11261   \let\markright\@glxtr@org@markright
11262   \let\markboth\@glxtr@org@markboth
11263   \let\@starttoc\@glxtr@org@@starttoc
11264 }
```

glxtrRevertTocMarks Just restores \@starttoc.

```
11265 \newcommand*{\glxtrRevertTocMarks}{%
11266   \let\@starttoc\@glxtr@org@@starttoc
11267 }
```

\glxtrifinmark

```
11268 \newcommand*{\glxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
11269 \newrobustcmd*{\@glsxtrinmark}{%
11270   \let\glsxtrifinmark\@firstoftwo
11271 }
```

glsxtrnotinmark

```
11272 \newrobustcmd*{\@glsxtrnotinmark}{%
11273   \let\glsxtrifinmark\@secondoftwo
11274 }
```

eorpdforheading

```
11275 \ifdef\texorpdfstring
11276 {
11277   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11278 }
11279 {
11280   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
11281 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11282 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
11283   \let\@glsxtr@org@MakeUppercase\MakeUppercase
11284   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11285   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11286   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11287   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11288   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11289   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11290   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11291   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11292   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11293   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11294   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11295   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11296   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11297   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11298   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11299   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11300   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11301   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11302   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11303   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11304   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11305   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11306   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```

11307 \let\glsxtrifinmark\@firstoftwo
11308 \let\MakeUppercase\MakeTextUppercase
11309 \let\glsxtrtitleorpdforheading\@thirdofthree
11310 \let\glsxtrtitleshort\glsxtrheadshort
11311 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11312 \let\Glsxtrtitleshort\Glsxtrheadshort
11313 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11314 \let\glsxtrtitlename\glsxtrheadname
11315 \let\Glsxtrtitlename\Glsxtrheadname
11316 \let\glsxtrtitletext\glsxtrheadtext
11317 \let\Glsxtrtitletext\Glsxtrheadtext
11318 \let\glsxtrtitleplural\glsxtrheadplural
11319 \let\Glsxtrtitleplural\Glsxtrheadplural
11320 \let\glsxtrtitlefirst\glsxtrheadfirst
11321 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11322 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11323 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11324 \let\glsxtrtitlelong\glsxtrheadlong
11325 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11326 \let\Glsxtrtitlelong\Glsxtrheadlong
11327 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11328 \let\glsxtrtitlefull\glsxtrheadfull
11329 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11330 \let\Glsxtrtitlefull\Glsxtrheadfull
11331 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11332 }

```

`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.)

```

11333 \newcommand*{\glsxtrrestoremarkhook}{%
11334 \let\glsxtrifinmark\@secondoftwo
11335 \let\MakeUppercase\@glsxtr@org@MakeUppercase
11336 \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
11337 \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
11338 \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
11339 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
11340 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
11341 \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
11342 \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
11343 \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
11344 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
11345 \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
11346 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
11347 \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
11348 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
11349 \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural

```



```

11350 \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
11351 \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
11352 \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
11353 \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
11354 \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
11355 \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
11356 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
11357 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
11358 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
11359 }

```

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.

```

11360 \newcommand*{\glsxtrheadshort}[1]{%
11361   \protect\NoCaseChange
11362   {%
11363     \glsifattribute{#1}{headuc}{true}%
11364     {%
11365       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11366     }%
11367     {%
11368       \glsxtrshort[noindex,hyper=false]{#1}[]%
11369     }%
11370   }%
11371 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11372 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11373   \glsxtrshort[noindex,hyper=false]{#1}[]%
11374 }

```

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.

```

11375 \newcommand*{\glsxtrheadshortpl}[1]{%
11376   \protect\NoCaseChange
11377   {%
11378     \glsifattribute{#1}{headuc}{true}%
11379     {%
11380       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11381     }%
11382     {%
11383       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11384     }%
11385   }%
11386 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11387 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11388   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11389 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11390 \newcommand*{\Glsxtrheadshort}[1]{%
11391   \protect\NoCaseChange
11392   {%
11393     \glsifattribute{#1}{headuc}{true}%
11394     {%
11395       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11396     }%
11397     {%
11398       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11399     }%
11400   }%
11401 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11402 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11403   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11404 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11405 \newcommand*{\Glsxtrheadshortpl}[1]{%
11406   \protect\NoCaseChange
11407   {%
11408     \glsifattribute{#1}{headuc}{true}%
11409     {%
11410       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11411     }%
11412     {%
11413       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11414     }%
11415   }%
11416 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11417 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11418   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11419 }
```

`\glsxtrheadname` As above but for the name value.

```

11420 \newcommand*{\glxtrheadname}[1]{%
11421   \protect\NoCaseChange
11422   {%
11423     \gl@ifattribute{#1}{headuc}{true}%
11424     {%
11425       \GLSname[noindex,hyper=false]{#1}[]%
11426     }%
11427     {%
11428       \glSname[noindex,hyper=false]{#1}[]%
11429     }%
11430   }%
11431 }

```

glxtrtitlename Command to display name value in section title and table of contents.

```

11432 \newrobustcmd*{\glxtrtitlename}[1]{%
11433   \glSname[noindex,hyper=false]{#1}[]%
11434 }

```

\Glsxtrheadname First letter converted to upper case

```

11435 \newcommand*{\Glsxtrheadname}[1]{%
11436   \protect\NoCaseChange
11437   {%
11438     \gl@ifattribute{#1}{headuc}{true}%
11439     {%
11440       \GLSname[noindex,hyper=false]{#1}[]%
11441     }%
11442     {%
11443       \Glsname[noindex,hyper=false]{#1}[]%
11444     }%
11445   }%
11446 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11447 %\changes{1.21}{2017-11-03}{new}
11448 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11449   \Glsname[noindex,hyper=false]{#1}[]%
11450 }

```

\glxtrheadtext As above but for the text value.

```

11451 \newcommand*{\glxtrheadtext}[1]{%
11452   \protect\NoCaseChange
11453   {%
11454     \gl@ifattribute{#1}{headuc}{true}%
11455     {%
11456       \GLStext[noindex,hyper=false]{#1}[]%
11457     }%
11458     {%
11459       \glstext[noindex,hyper=false]{#1}[]%

```

```

11460 }%
11461 }%
11462 }

```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```

11463 \newrobustcmd*{\glsxtrtitletext}[1]{%
11464   \glstext[noindex,hyper=false]{#1}[]%
11465 }

```

`\Glsxtrheadtext` First letter converted to upper case

```

11466 \newcommand*{\Glsxtrheadtext}[1]{%
11467   \protect\NoCaseChange
11468   {%
11469     \glsifattribute{#1}{headuc}{true}%
11470     {%
11471       \GLStext[noindex,hyper=false]{#1}[]%
11472     }%
11473     {%
11474       \Glstext[noindex,hyper=false]{#1}[]%
11475     }%
11476   }%
11477 }

```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```

11478 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11479   \Glstext[noindex,hyper=false]{#1}[]%
11480 }

```

`lsxtrheadplural` As above but for the plural value.

```

11481 \newcommand*{\lsxtrheadplural}[1]{%
11482   \protect\NoCaseChange
11483   {%
11484     \glsifattribute{#1}{headuc}{true}%
11485     {%
11486       \GLSplural[noindex,hyper=false]{#1}[]%
11487     }%
11488     {%
11489       \glsplural[noindex,hyper=false]{#1}[]%
11490     }%
11491   }%
11492 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```

11493 \newrobustcmd*{\sxtrtitleplural}[1]{%
11494   \glsplural[noindex,hyper=false]{#1}[]%
11495 }

```

lsxtrheadplural Convert first letter to upper case.

```
11496 \newcommand*{\Glsxtrheadplural}[1]{%
11497   \protect\NoCaseChange
11498   {%
11499     \glsifattribute{#1}{headuc}{true}%
11500     {%
11501       \GLSplural[noindex,hyper=false]{#1}[]%
11502     }%
11503     {%
11504       \Glsplural[noindex,hyper=false]{#1}[]%
11505     }%
11506   }%
11507 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11508 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11509   \Glsplural[noindex,hyper=false]{#1}[]%
11510 }
```

glxtrheadfirst As above but for the first value.

```
11511 \newcommand*{\glxtrheadfirst}[1]{%
11512   \protect\NoCaseChange
11513   {%
11514     \glsifattribute{#1}{headuc}{true}%
11515     {%
11516       \GLSfirst[noindex,hyper=false]{#1}[]%
11517     }%
11518     {%
11519       \glsfirst[noindex,hyper=false]{#1}[]%
11520     }%
11521   }%
11522 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
11523 \newrobustcmd*{\glxtrtitlefirst}[1]{%
11524   \glsfirst[noindex,hyper=false]{#1}[]%
11525 }
```

Glsxtrheadfirst First letter converted to upper case

```
11526 \newcommand*{\Glsxtrheadfirst}[1]{%
11527   \protect\NoCaseChange
11528   {%
11529     \glsifattribute{#1}{headuc}{true}%
11530     {%
11531       \GLSfirst[noindex,hyper=false]{#1}[]%
11532     }%
11533     {%
```

```

11534     \Glsfirst[noindex,hyper=false]{#1}[]%
11535 }%
11536 }%
11537 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11538 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11539   \Glsfirst[noindex,hyper=false]{#1}[]%
11540 }

```

headfirstplural As above but for the firstplural value.

```

11541 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11542   \protect\NoCaseChange
11543   {%
11544     \glsifattribute{#1}{headuc}{true}%
11545     {%
11546       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11547     }%
11548     {%
11549       \glsfirstplural[noindex,hyper=false]{#1}[]%
11550     }%
11551   }%
11552 }

```

itlefirstplural Command to display firstplural value in section title and table of contents.

```

11553 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11554   \glsfirstplural[noindex,hyper=false]{#1}[]%
11555 }

```

headfirstplural First letter converted to upper case

```

11556 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11557   \protect\NoCaseChange
11558   {%
11559     \glsifattribute{#1}{headuc}{true}%
11560     {%
11561       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11562     }%
11563     {%
11564       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11565     }%
11566   }%
11567 }

```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11568 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11569   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11570 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```
11571 \newcommand*{\glxstrheadlong}[1]{%
11572   \protect\NoCaseChange
11573   {%
11574     \gl@ifattribute{#1}{headuc}{true}%
11575     {%
11576       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11577     }%
11578     {%
11579       \glxstrlong[noindex,hyper=false]{#1}[]%
11580     }%
11581   }%
11582 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
11583 \newrobustcmd*{\glxstrtitlelong}[1]{%
11584   \glxstrlong[noindex,hyper=false]{#1}[]%
11585 }
```

`lsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
11586 \newcommand*{\glxtrheadlongpl}[1]{%
11587   \protect\NoCaseChange
11588   {%
11589     \gl@ifattribute{#1}{headuc}{true}%
11590     {%
11591       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11592     }%
11593     {%
11594       \glxstrlongpl[noindex,hyper=false]{#1}[]%
11595     }%
11596   }%
11597 }
```

`sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
11598 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
11599   \glxstrlongpl[noindex,hyper=false]{#1}[]%
11600 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
11601 \newcommand*{\Glsxtrheadlong}[1]{%
11602   \protect\NoCaseChange
11603   {%
11604     \gl@ifattribute{#1}{headuc}{true}%
11605     {%
11606       \GLSxtrlong[noindex,hyper=false]{#1}[]%
```

```

11607 }%
11608 {%
11609     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11610 }%
11611 }%
11612 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11613 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11614     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11615 }

```

lgsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11616 \newcommand*{\Glsxtrheadlongpl}[1]{%
11617     \protect\NoCaseChange
11618     {%
11619         \glsifattribute{#1}{headuc}{true}%
11620         {%
11621             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11622         }%
11623         {%
11624             \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11625         }%
11626     }%
11627 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11628 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11629     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11630 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11631 \newcommand*{\glsxtrheadfull}[1]{%
11632     \protect\NoCaseChange
11633     {%
11634         \glsifattribute{#1}{headuc}{true}%
11635         {%
11636             \GLSxtrfull[noindex,hyper=false]{#1}[]%
11637         }%
11638         {%
11639             \glsxtrfull[noindex,hyper=false]{#1}[]%
11640         }%
11641     }%
11642 }

```


`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
11643 \newrobustcmd*{\glsxtrtitlefull}[1]{%  
11644   \glsxtrfull[noindex,hyper=false]{#1}[]%  
11645 }
```

`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.

```
11646 \newcommand*{\glsxtrheadfullpl}[1]{%  
11647   \protect\NoCaseChange  
11648   {%  
11649     \glsifattribute{#1}{headuc}{true}%  
11650     {%  
11651       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%  
11652     }%  
11653     {%  
11654       \glsxtrfullpl[noindex,hyper=false]{#1}[]%  
11655     }%  
11656   }%  
11657 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
11658 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%  
11659   \glsxtrfullpl[noindex,hyper=false]{#1}[]%  
11660 }
```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
11661 \newcommand*{\Glsxtrheadfull}[1]{%  
11662   \protect\NoCaseChange  
11663   {%  
11664     \glsifattribute{#1}{headuc}{true}%  
11665     {%  
11666       \GLSxtrfull[noindex,hyper=false]{#1}[]%  
11667     }%  
11668     {%  
11669       \Glsxtrfull[noindex,hyper=false]{#1}[]%  
11670     }%  
11671   }%  
11672 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11673 \newrobustcmd*{\Glsxtrtitlefull}[1]{%  
11674   \Glsxtrfull[noindex,hyper=false]{#1}[]%  
11675 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11676 \newcommand*{\Glsxtrheadfullpl}[1]{%
11677   \protect\NoCaseChange
11678   {%
11679     \glsifattribute{#1}{headuc}{true}%
11680     {%
11681       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11682     }%
11683     {%
11684       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11685     }%
11686   }%
11687 }

```

`\sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11688 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11689   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11690 }

```

`\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.

```

11691 \ifdef\texorpdfstring
11692 {
11693   \newcommand*{\glsfmtshort}[1]{%
11694     \texorpdfstring
11695       {\glsxtrtitleshort{#1}}%
11696       {\glsentryshort{#1}}%
11697   }
11698 }
11699 {
11700   \newcommand*{\glsfmtshort}[1]{%
11701     \glsxtrtitleshort{#1}}
11702 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11703 \ifdef\texorpdfstring
11704 {
11705   \newcommand*{\glsfmtshortpl}[1]{%
11706     \texorpdfstring
11707       {\glsxtrtitleshortpl{#1}}%
11708       {\glsentryshortpl{#1}}%
11709   }
11710 }
11711 {
11712   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11713 \glstrtitleshortpl{#1}}
11714 }

```

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).

```

11715 \ifdef\texorpdfstring
11716 {
11717   \newcommand*{\Glsfmtshort}[1]{%
11718     \texorpdfstring
11719       {\glstrtitleshort{#1}}%
11720       {\glentryshort{#1}}%
11721   }
11722 }
11723 {
11724   \newcommand*{\Glsfmtshort}[1]{%
11725     \glstrtitleshort{#1}}
11726 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11727 \ifdef\texorpdfstring
11728 {
11729   \newcommand*{\Glsfmtshortpl}[1]{%
11730     \texorpdfstring
11731       {\glstrtitleshortpl{#1}}%
11732       {\glentryshortpl{#1}}%
11733   }
11734 }
11735 {
11736   \newcommand*{\Glsfmtshortpl}[1]{%
11737     \glstrtitleshortpl{#1}}
11738 }

```

`\glsfmtname` As above but for the name value.

```

11739 \ifdef\texorpdfstring
11740 {
11741   \newcommand*{\glsfmtname}[1]{%
11742     \texorpdfstring
11743       {\glstrtitlename{#1}}%
11744       {\glentryname{#1}}%
11745   }
11746 }
11747 {
11748   \newcommand*{\glsfmtname}[1]{%
11749     \glstrtitlename{#1}}
11750 }

```

`\Glsfmtname` First letter converted to upper case.

```

11751 \ifdef\texorpdfstring
11752 {
11753   \newcommand*{\Glsfmtname}[1]{%
11754     \texorpdfstring
11755     {\Glsxtrtitlename{#1}}%
11756     {\glsentryname{#1}}%
11757   }
11758 }
11759 {
11760   \newcommand*{\Glsfmtname}[1]{%
11761     \Glsxtrtitlename{#1}}
11762 }

```

`\glsfmttext` As above but for the text value.

```

11763 \ifdef\texorpdfstring
11764 {
11765   \newcommand*{\glsfmttext}[1]{%
11766     \texorpdfstring
11767     {\glsxtrtitletext{#1}}%
11768     {\glsentrytext{#1}}%
11769   }
11770 }
11771 {
11772   \newcommand*{\glsfmttext}[1]{%
11773     \glsxtrtitletext{#1}}
11774 }

```

`\Glsfmttext` First letter converted to upper case.

```

11775 \ifdef\texorpdfstring
11776 {
11777   \newcommand*{\Glsfmttext}[1]{%
11778     \texorpdfstring
11779     {\Glsxtrtitletext{#1}}%
11780     {\glsentrytext{#1}}%
11781   }
11782 }
11783 {
11784   \newcommand*{\Glsfmttext}[1]{%
11785     \Glsxtrtitletext{#1}}
11786 }

```

`\glsfmtplural` As above but for the plural value.

```

11787 \ifdef\texorpdfstring
11788 {
11789   \newcommand*{\glsfmtplural}[1]{%
11790     \texorpdfstring
11791     {\glsxtrtitleplural{#1}}%
11792     {\glsentryplural{#1}}%
11793   }

```

```

11794 }
11795 {
11796   \newcommand*{\glsfmtplural}[1]{%
11797     \glsxtrtitleplural{#1}}
11798 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11799 \ifdef\texorpdfstring
11800 {
11801   \newcommand*{\Glsfmtplural}[1]{%
11802     \texorpdfstring
11803       {\Glsxtrtitleplural{#1}}%
11804       {\glsentryplural{#1}}%
11805   }
11806 }
11807 {
11808   \newcommand*{\Glsfmtplural}[1]{%
11809     \Glsxtrtitleplural{#1}}
11810 }

```

`\glsfmtfirst` As above but for the first value.

```

11811 \ifdef\texorpdfstring
11812 {
11813   \newcommand*{\glsfmtfirst}[1]{%
11814     \texorpdfstring
11815       {\glsxtrtitlefirst{#1}}%
11816       {\glsentryfirst{#1}}%
11817   }
11818 }
11819 {
11820   \newcommand*{\glsfmtfirst}[1]{%
11821     \glsxtrtitlefirst{#1}}
11822 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11823 \ifdef\texorpdfstring
11824 {
11825   \newcommand*{\Glsfmtfirst}[1]{%
11826     \texorpdfstring
11827       {\Glsxtrtitlefirst{#1}}%
11828       {\glsentryfirst{#1}}%
11829   }
11830 }
11831 {
11832   \newcommand*{\Glsfmtfirst}[1]{%
11833     \Glsxtrtitlefirst{#1}}
11834 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11835 \ifdef\texorpdfstring
11836 {
11837   \newcommand*\glsfmtfirstpl}[1]{%
11838     \texorpdfstring
11839     {\glsxtrtitlefirstplural{#1}}%
11840     {\glsentryfirstplural{#1}}%
11841   }
11842 }
11843 {
11844   \newcommand*\glsfmtfirstpl}[1]{%
11845     \glsxtrtitlefirstplural{#1}}
11846 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11847 \ifdef\texorpdfstring
11848 {
11849   \newcommand*\Glsfmtfirstpl}[1]{%
11850     \texorpdfstring
11851     {\Glsxtrtitlefirstplural{#1}}%
11852     {\glsentryfirstplural{#1}}%
11853   }
11854 }
11855 {
11856   \newcommand*\Glsfmtfirstpl}[1]{%
11857     \Glsxtrtitlefirstplural{#1}}
11858 }

```

`\glsfmtlong` As above but for the long value.

```

11859 \ifdef\texorpdfstring
11860 {
11861   \newcommand*\glsfmtlong}[1]{%
11862     \texorpdfstring
11863     {\glsxtrtitlelong{#1}}%
11864     {\glsentrylong{#1}}%
11865   }
11866 }
11867 {
11868   \newcommand*\glsfmtlong}[1]{%
11869     \glsxtrtitlelong{#1}}
11870 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11871 \ifdef\texorpdfstring
11872 {
11873   \newcommand*\Glsfmtlong}[1]{%
11874     \texorpdfstring
11875     {\Glsxtrtitlelong{#1}}%
11876     {\glsentrylong{#1}}%
11877   }

```

```

11878 }
11879 {
11880   \newcommand*{\Glsfmtlong}[1]{%
11881     \Glsxtrtitlelong{#1}}
11882 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11883 \ifdef\texorpdfstring
11884 {
11885   \newcommand*{\glsfmtlongpl}[1]{%
11886     \texorpdfstring
11887     {\Glsxtrtitlelongpl{#1}}%
11888     {\Glsentrylongpl{#1}}%
11889   }
11890 }
11891 {
11892   \newcommand*{\glsfmtlongpl}[1]{%
11893     \Glsxtrtitlelongpl{#1}}
11894 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11895 \ifdef\texorpdfstring
11896 {
11897   \newcommand*{\Glsfmtlongpl}[1]{%
11898     \texorpdfstring
11899     {\Glsxtrtitlelongpl{#1}}%
11900     {\Glsentrylongpl{#1}}%
11901   }
11902 }
11903 {
11904   \newcommand*{\Glsfmtlongpl}[1]{%
11905     \Glsxtrtitlelongpl{#1}}
11906 }

```

`\glsfmtfull` In-line full format.

```

11907 \ifdef\texorpdfstring
11908 {
11909   \newcommand*{\glsfmtfull}[1]{%
11910     \texorpdfstring
11911     {\Glsxtrtitlefull{#1}}%
11912     {\Glsxtrinlinefullformat{#1}{}}%
11913   }
11914 }
11915 {
11916   \newcommand*{\glsfmtfull}[1]{%
11917     \Glsxtrtitlefull{#1}}
11918 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11919 \ifdef\txorpdfstring
11920 {
11921   \newcommand*{\Glsfmtfull}[1]{%
11922     \txorpdfstring
11923     {\Glsxtrtitlefull{#1}}%
11924     {\Glsxtrinlinefullformat{#1}{}}%
11925   }
11926 }
11927 {
11928   \newcommand*{\Glsfmtfull}[1]{%
11929     \Glsxtrtitlefull{#1}}
11930 }

```

`\glsfmtfullpl` In-line full plural format.

```

11931 \ifdef\txorpdfstring
11932 {
11933   \newcommand*{\glsfmtfullpl}[1]{%
11934     \txorpdfstring
11935     {\glsxtrtitlefullpl{#1}}%
11936     {\glsxtrinlinefullplformat{#1}{}}%
11937   }
11938 }
11939 {
11940   \newcommand*{\glsfmtfullpl}[1]{%
11941     \glsxtrtitlefullpl{#1}}
11942 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11943 \ifdef\txorpdfstring
11944 {
11945   \newcommand*{\Glsfmtfullpl}[1]{%
11946     \txorpdfstring
11947     {\Glsxtrtitlefullpl{#1}}%
11948     {\Glsxtrinlinefullplformat{#1}{}}%
11949   }
11950 }
11951 {
11952   \newcommand*{\Glsfmtfullpl}[1]{%
11953     \Glsxtrtitlefullpl{#1}}
11954 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11955 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```



```

11956 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11957 }

```

sariesExtraLang

```

11958 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11959   \ProvidesFile{glossariesxtr-#1.ldf}%
11960 }

```

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.)

`xtr@loaddialect` The dialect label should be stored in `\this@dialect` before using this command.

```

11961 \newcommand{\glxtr@loaddialect}{%
11962   \IfTrackedLanguageFileExists{\this@dialect}%
11963   {glossariesxtr-}% prefix
11964   {.ldf}%
11965   {%
11966     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11967   }%
11968   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11969 \@glxtrdialecthook
11970 }

```

```

11971 \ifpackageloaded{tracklang}
11972 {%
11973   \AnyTrackedLanguages
11974   {%
11975     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11976   }%
11977   {}%
11978 }
11979 {}

```

Load `glossaries-extra-stylemods` if required.

```

11980 \@glxtr@redefstyles

```

and set the style:

```

11981 \@glxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11982 \NeedsTeXFormat{LaTeX2e}
11983 \ProvidesPackage{glossaries-extra-bib2gls}[2018/11/30 1.37 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11984 \newcommand*{\glshex}{\string\u}

```

`\glscapturedgroup`

```

11985 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```

11986 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11987   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11988 }

```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like \providecommand in the document but like \renewcommand in bib2gls.

```

11989 \newcommand*{\GlsXtrProvideCommand}{\providecommand}

```

`\GlsXtrRenewCommand` Like \renewcommand but only generates a warning rather than an error if the command isn't defined.

```

11990 \newcommand*{\GlsXtrRenewCommand}{\@star@or@long\GlsXtr@renewcommand}

```

`\GlsXtr@renewcommand`

```

11991 \newcommand*{\GlsXtr@renewcommand}[1]{%
11992   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
11993   \expandafter\@ifundefined\@gtempa
11994   {%
11995     \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
11996   }%
11997   \relax
11998   \relax
11999   \let\@ifdefinable\@rc@ifdefinable
12000   \new@command#1%
12001 }

```

`\GlsXtr@Wrglossarylocation` For use with indexcounter and bib2gls.

```

12002 \newcommand*{\GlsXtr@Wrglossarylocation}[2]{#1}

```

`\GlsXtrIndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with indexcounter and bib2gls.

```

12003 \ifdef\hyperref
12004 {%
12005   \newcommand*{\GlsXtrIndexCounterLink}[2]{%

```

```

12006 \glstrifhasfield{indexcounter}{#2}%
12007 {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
12008 {#1}%
12009 }
12010 }
12011 {
12012 \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
12013 }

```

`\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.

```

12014 \newcommand*{\GlsXtrDualField}{dual}

```

`\GlsXtrDualBackLink` `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```

12015 \newcommand*{\GlsXtrDualBackLink}[2]{%
12016 \glstrifhasfield{\GlsXtrDualField}{#2}%
12017 {\glshyperlink[#1]{\glscurrentfieldvalue}}%
12018 {#2}%
12019 }

```

`\TeXEntryAliases` Convenient shortcut for use with entry-type-aliases to alias standard Bib_T_EX entry types to @bibtexentry.

```

12020 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
12021 article=bibtexentry,
12022 book=bibtexentry,
12023 booklet=bibtexentry,
12024 conference=bibtexentry,
12025 inbook=bibtexentry,
12026 incollection=bibtexentry,
12027 inproceedings=bibtexentry,
12028 manual=bibtexentry,
12029 mastersthesis=bibtexentry,
12030 misc=bibtexentry,
12031 phdthesis=bibtexentry,
12032 proceedings=bibtexentry,
12033 techreport=bibtexentry,
12034 unpublished=bibtexentry
12035 }

```

`\provideBibTeXFields` Convenient shortcut to define the standard BibTeX fields.

```
12036 \newcommand*\GlsXtrProvideBibTeXFields{%
12037   \glsaddstoragekey{address}{}\{\glsxtrbibaddress}%
12038   \glsaddstoragekey{author}{}\{\glsxtrbibauthor}%
12039   \glsaddstoragekey{booktitle}{}\{\glsxtrbibbooktitle}%
12040   \glsaddstoragekey{chapter}{}\{\glsxtrbibchapter}%
12041   \glsaddstoragekey{edition}{}\{\glsxtrbibedition}%
12042   \glsaddstoragekey{howpublished}{}\{\glsxtrbibhowpublished}%
12043   \glsaddstoragekey{institution}{}\{\glsxtrbibinstitution}%
12044   \glsaddstoragekey{journal}{}\{\glsxtrbibjournal}%
12045   \glsaddstoragekey{month}{}\{\glsxtrbibmonth}%
12046   \glsaddstoragekey{note}{}\{\glsxtrbibnote}%
12047   \glsaddstoragekey{number}{}\{\glsxtrbibnumber}%
12048   \glsaddstoragekey{organization}{}\{\glsxtrbiborganization}%
12049   \glsaddstoragekey{pages}{}\{\glsxtrbibpages}%
12050   \glsaddstoragekey{publisher}{}\{\glsxtrbibpublisher}%
12051   \glsaddstoragekey{school}{}\{\glsxtrbibschooll}%
12052   \glsaddstoragekey{series}{}\{\glsxtrbibseries}%
12053   \glsaddstoragekey{title}{}\{\glsxtrbibtitle}%
12054   \glsaddstoragekey{bibtextype}{}\{\glsxtrbibtype}%
12055   \glsaddstoragekey{volume}{}\{\glsxtrbibvolume}%
12056 }
```

Multiple supplementary references are only supported with `bib2gls`.

`\ltisupplocation` This is like `\glsxtrsupphypernumber` 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.

```
12057 \newcommand*\glsxtrmultisupplocation}[3]{%
12058   {%
12059     \def\glsxtrsupplocationurl{#2}%
12060     \glsxtrhypernumber{#1}%
12061   }%
12062 }
```

`\trdisplaysupplloc` `\glsxtrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```
12063 \newcommand*\glsxtrdisplaysupplloc[5]{%
12064   \setentrycounter{#1}{#2}%
12065   \glsxtrmultisupplocation{#5}{#4}{#3}%
12066 }
```

`splaylocnameref` `\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`.

```
12067 \ifundef\hyperlink
12068 {
12069   \newcommand*\glstrdisplaylocnameref[8]{%
12070     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
12071   }
12072 }
12073 {
```

Default action uses `<hcounter>`. Equations and pages typically don't have a title, so check the counter name.

```
12074   \newcommand*\glstrdisplaylocnameref[8]{%
12075     \ifstrequal{#2}{equation}%
12076     {\glsxtrnameref link{#3}{( #4 )}{#2.#7}{#8}}%
12077     {%
12078       \ifstreempty{#5}%
12079       {%
```

No title, so just use the location as the link text.

```
12080       \glsxtrnameref link{#3}{#4}{#2.#7}{#8}%
12081     }%
12082   }%
12083     \ifstrequal{#2}{page}%
12084     {\glsxtrnameref link{#3}{#4}{#2.#7}{#8}}%
12085     {\glsxtrnameref link{#3}{#5}{#2.#7}{#8}}%
12086   }%
12087 }%
12088 }
12089 }
```

`lsxtrnameref link` `\glstrfmt nameref link{<format>}{<title>}{<href>}{<external file>}`

```
12090 \newcommand*\glsxtrnameref link[4]{%
```

Locally change `\glshypernumber` to `\@firstofone` to remove the normal location hyperlink.

```
12091   \begingroup
12092   \let\glshypernumber\@firstofone
```

If the `<external file>` argument is empty, an internal link is used, otherwise an external one is needed.

```
12093   \ifstreempty{#4}%
12094   {\glstrfmt internal nameref{#3}{#1}{#2}}%
```

```

12095     {\glxstrfmtexternalnameref{#3}{#1}{#2}{#4}}%
12096 \endgroup
12097 }

```

```

lsxtrnameloclink \glxstrnamerefloclink{<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.

```

12098 \newcommand{\glxtrnameloclink}[6]{%
12099 \begingroup
12100 \setentrycounter[#1]{#2}%
12101 \def\glxtr@locationhypertext{#5}%
12102 \let\glshypernumber\@firstofone
12103 \def\@glsnumberformat{#3}%
12104 \def\glxtrsupplocationurl{#6}%
12105 \toks@={}%
12106 \@glxtr@bibgls@removespaces#4 \@nil
12107 \endgroup
12108 }

```

`ls@removespaces`

```

12109 \def\@glxtr@bibgls@removespaces#1 #2\@nil{%
12110 \toks@=\expandafter{\the\toks@#1}%
12111 \ifx\#2\%
12112 \edef\x{\the\toks@}%
12113 \ifx\x\empty
12114 \else
12115 \protected@edef\x{\glentrycounter\@glo@counterprefix\the\toks@}%
12116 \ifvoid\glxtrsupplocationurl
12117 {%
12118 \expandafter\glxstrfmtinternalnameref\expandafter{\x}%
12119 {\@glsnumberformat}{\glxtr@locationhypertext}%
12120 }%
12121 {%
12122 \expandafter\glxstrfmtexternalnameref\expandafter{\x}%
12123 {\@glsnumberformat}{\glxtr@locationhypertext}{\glxtrsupplocationurl}%
12124 }%
12125 \fi
12126 \else
12127 \@gls@ReturnAfterFi{%
12128 \@glxtr@bibgls@removespaces#2\@nil
12129 }%
12130 \fi
12131 }

```