

glossaries-extra.sty v1.41: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2019-04-09

Abstract

This is the documented code for the glossaries-extra package. See glossaries-extra-manual.pdf for the user manual.

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

Contents

1 Main Package Code (glossaries-extra.sty)	5
1.1 Package Initialisation and Options	5
1.2 Extra Utilities	30
1.3 Modifications to Commands Provided by glossaries	44
1.3.1 Existence Checks	49
1.3.2 Document Definitions	58
1.3.3 Existing Glossary Style Modifications	64
1.3.4 Entry Formatting, Hyperlinks and Indexing	68
1.3.5 Entry Counting	106
1.3.6 Acronym Modifications	121
1.3.7 Indexing and Displaying Glossaries	124
1.4 Link Counting	162
1.5 Integration with glossaries-accsupp	164
1.6 Categories	178
1.7 Abbreviations	204
1.7.1 Abbreviation Styles Setup	224
1.7.2 Predefined Styles (Default Font)	227
1.7.3 Predefined Styles (Small Capitals)	244
1.7.4 Predefined Styles (Fake Small Capitals)	259
1.7.5 Predefined Styles (Emphasized)	273
1.7.6 Predefined Styles (User Parentheses Hook)	294
1.7.7 Predefined Styles (Hyphen)	304
1.7.8 Predefined Styles (No Short on First Use)	318
1.8 Using Entries in Headings	320
1.9 Multi-Lingual Support	340
1.10 glossaries-extra-bib2gls.sty	341
2 Style Adjustments (glossaries-extra-stylemods.sty)	386
2.1 Package Initialisation	386
2.2 List-Like Styles	387
2.3 Longtable Styles	390
2.4 Long Ragged Styles	392
2.5 Supertabular Styles	394
2.6 Super Ragged Styles	396
2.7 Inline Style	398
2.8 Tree Styles	398
2.9 Multicolumn Styles	416

3	bookindex style (glossary-bookindex.sty)	422
3.1	Package Initialisation and Options	422
4	longextra styles (glossary-longextra.sty)	429
4.1	Package Initialisation and Options	429
5	topic styles (glossary-topic.sty)	452
5.1	Package Initialisation and Options	452
	Glossary	457
	Change History	458
	Index	480

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2019/04/09 v1.41 (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{\glsxtr@dooption}[1]{\setupglossaries{#1}}%
```

```
8 \let\@glsxtr@declareoption\@gls@declareoption
```

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

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

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

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

```
27 }
```

```
28 }
```

Declare package options.

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

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

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

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

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

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

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

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

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

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

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

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

```
41 \newcommand*\@glxtr@warn@onexistsordo}[1]{%
42 \GlossariesExtraWarning{\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*\@glxtr@redef@for@gl@sentries}{}
```

`\f@for@gl@sentries`

```
48 \newcommand*\@glxtr@do@redef@for@gl@sentries}{%
49 \renewcommand*\f@for@gl@sentries}[3][\gl@default@type]{%
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 [\glxtr@undefaction@val\glxtr@undefaction@nr]%
65 {warn,error}%
66 {%
67   \ifcase\glxtr@undefaction@nr\relax
68     \let\glxtrundefaction\@glxtr@warn@undefaction
69     \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
70     \let\@glxtr@redef@forglsentries\@glxtr@do@redef@forglsentries
71   \or
72     \let\glxtrundefaction\@glxtr@err@undefaction
73     \let\glxtr@warnonexistsordo\@gobble
74     \let\@glxtr@redef@forglsentries\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.

```

\@glxtr@record Does nothing by default.
77 \newcommand*{\@glxtr@record}[3]{

```

```

\glxtr@recordsee Does nothing by default.
78 \newcommand*{\glxtr@recordsee}[2]{

```

```

ultnumberformat
79 \newcommand*{\@glxtr@defaultnumberformat}{glsnumberformat}%

```

```

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

```

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

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

```

83 \newcommand*{\@glsxtr@do@record@wrglossary}[1]{%
84 \begingroup
85   \ifKV@gls@link@noindex
86   \else
87     \edef\@gls@label{\glsdetoklabel{#1}}%
88     \let\glslabel\@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 }

```

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 \beginingroup
```

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 \if@glsxtr@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}

```

lslink@postkeys

```

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

```

lossadd@prekeys

```

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

```

ossadd@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

```

Protect against non-expandable commands occurring in the location.

```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

`\gls@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 `\theglsentrycounter`, which is useful if it can't be obtained by prefixing `\theglsentrycounter`.

```

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

```

`\gls@recordcounter`

```

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

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

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

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

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

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

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

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

```

```

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

```

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

```

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

```

Add a key to store the group heading.

```

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

```

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

```

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

```

`setting@alsoindex`

```

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

```

`rd@setting@only`

```

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

```

`setting@nameref`

```

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

```

`@if@record@only`

```

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

```

```

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

```

ord@setting@off

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

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

```

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

```

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

```
338 \def\glsxtrsetaliasnoindex{}%
```

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

```
339 \ifdefined\@gls@setupsort@none{\@gls@setupsort@none}{}%
```

Warn about using \printglossary:

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

Load glossaries-extra-bib2gls:

```

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

```

record Now define the record package option.

```

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

```

Don't record.

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

Only record (don't index).

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

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

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

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

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

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

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

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


Need to provide conditional commands that are backward compatible:

`\if@glstrdocdef`

```
385 \newcommand*{\if@glstrdocdef}{\ifnum\@glstr@docdefval>0 }
```

`\glstrdocdeftrue`

```
386 \newcommand*{\@glstrdocdeftrue}{\def\@glstr@docdefval{1}}
```

`\glstrdocdeffalse`

```
387 \newcommand*{\@glstrdocdeffalse}{\def\@glstr@docdefval{0}}
```

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

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

`\docdefrestricted`

```
398 \newcommand*{\if@glstrdocdefrestricted}{\ifnum\@glstr@docdefval>1 }
```

`\glstrdoifexistsorwarn`

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

```
399 \newcommand*{\@glstrdoifexistsorwarn}{\glstrdoifexistsorwarn}
```

`\indexcrossrefs`

Automatically index cross references at the end of the document

```
400 \define@boolkey{glossaries-extra.sty}[\@glstr]{indexcrossrefs}[true]{%
401   \if@glstrindexcrossrefs
402   \else
403     \renewcommand*{\@glstr@autoindexcrossrefs}{}%
404   \fi
405 }
```

Switch off since this can increase the build time.

```
406 \@glstrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

`\autoindexcrossrefs`

```
407 \newcommand*{\@glstr@autoindexcrossrefs}{\@glstrindexcrossrefstrue}
```

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

```

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

```

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

```

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

```

\glsxtr@float

```

414 \let\glsxtr@float\@float

```

glsxtr@dblfloat

```

415 \let\glsxtr@dblfloat\@dblfloat

```

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

```

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

```

iesExtraWarning Allow users to suppress warnings.

```

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

```

raWarningNoLine Allow users to suppress warnings.

```

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

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

```

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

```

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

```

postdot Shortcut for nopostdot=false

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

nopostdot Needs to redefine \@glsxtr@defpostpunc

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

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

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

```
478 }%
479 }
```

`glsxtrabbrvtype` Glossary type for abbreviations.

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

`bbreviationsdef` Set by abbreviations option.

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

`bbreviationsdef`

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

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

```
492 \ifglsacronym
493 \else
494 \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
495 \fi
496 }%
```

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

```
497 \@glsxtr@declareoption{abbreviations}{%
498 \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
499 }
```

`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`).

```
500 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
501 \newcommand*{\ab}{\cglsl}%
502 \newcommand*{\abp}{\cglspl}%
503 \newcommand*{\as}{\glsxtrshort}%
504 \newcommand*{\asp}{\glsxtrshortpl}%
505 \newcommand*{\al}{\glsxtrlong}%
506 \newcommand*{\alp}{\glsxtrlongpl}%
507 \newcommand*{\af}{\glsxtrfull}%
508 \newcommand*{\afp}{\glsxtrfullpl}%
509 \newcommand*{\Ab}{\cGls}%
510 \newcommand*{\Abp}{\cGlspl}%
511 \newcommand*{\As}{\Glsxtrshort}%
512 \newcommand*{\Asp}{\Glsxtrshortpl}%
```

```

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

525 \providecommand*\newabbr{\newabbreviation}%

```

Disable this command after it's been used.

```

526 \let\GLSXtrDefineAbbreviationShortcuts\relax
527 }

```

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

```

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

553 \providecommand*\newabbr{\newabbreviation}%

```

Disable this command after it's been used.

```
554 \let\GlsXtrDefineAcShortcuts\relax
555 }
```

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

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

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

`\GlsXtrSetupShortcuts` Command used to set the shortcuts option.

```
568 \newcommand*\GlsXtrSetupShortcuts{}
```

`\GlsXtrShortcutsVal` Store the value of the shortcuts option. (Needed by bib2gls.)

```
569 \newcommand*\GlsXtrShortcutsVal{\ifglsacrshortcuts acro\else none\fi}%
```

`\GlsXtrShortcuts` Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides `shortcuts=true` and `shortcuts=false`, which are equivalent to `shortcuts=all` and `shortcuts=none`. Multiple use of this option in the *same* option list will override each other. New to v1.17: `shortcuts=ac` which implements `\GlsXtrDefineAcShortcuts` (not included in `shortcuts=all` as it conflicts with other shortcuts).

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

```

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

```

Leave none and false as last option.

```

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

```

`lsxtr@doaccsupp`

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

`accsupp` If `accsupp`, load `glossaries-accsupp` package.

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

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

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

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

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

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

xtr@redefstyles

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

stylemods

```
637 \define@key{glossaries-extra.sty}{stylemods}[default]{%
638   \ifstrequal{#1}{default}%
639   {%
640     \renewcommand*\@glxtr@redefstyles{%
641       \RequirePackage{glossaries-extra-stylemods}}%
642   }%
643   {%
644     \ifstrequal{#1}{all}%
645     {%
646       \renewcommand*\@glxtr@redefstyles{%
647         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
648         \RequirePackage{glossaries-extra-stylemods}%
649       }%
650     }%
651     {%
652       \renewcommand*\@glxtr@redefstyles{%
653         \for \@glxtr@tmp:=#1\do{%
654           \IfFileExists{glossary-\@glxtr@tmp.sty}%
655           {%
656             \eappto \@glxtr@redefstyles{%
657               \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
658             }%
659           {%
660             \PackageError{glossaries-extra}%
661             {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
662              doesn’t exist (did you mean to use the ‘style’ key?)}%
663             {The list of values (#1) in the ‘stylemods’ key should
664              match the glossary-xxx.sty files provided with
665              glossaries.sty}%
666           }%
667         }%
668       }%
669     }%
670   }%
671 }
```



```

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

```

glsxtr@do@style

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

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

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

Defer actual style change:

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

Set this as the default style:

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

Set this style:

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

```
677 }%
```

```
678 }
```

wrglossaryctr

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

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

locationHyperlink

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

The first two arguments are always control sequences.

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

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

```
682 }
```

locationhyperlink

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

```
684 \pageref{wrglossary.#3}%
```

```
685 }
```

indexcounter

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

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

```

686 \@glsxtr@declareoption{indexcounter}{%
687   \glsxtr@doooption{counter=wrglossary}%
688   \ifundef\c@wrglossary
689   {%
690     \newcounter{wrglossary}%
691     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
692   }%
693   {}%
694   \renewcommand*{\glsxtr@inc@wrglossaryctr}[1]{%
    Only increment if the current counter is wrglossary.
695     \ifdefstring\@gls@counter{wrglossary}%
696     {%
697       \refstepcounter{wrglossary}%
698       \label{wrglossary.\thewrglossary}%
699     }%
700     {}%
701   }%
702   \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
703     \ifdefstring\glsentrycounter{wrglossary}%
704     {%
705       \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
706     }%
707     {\glsxtrhyperlink{##1##2##3}{##3}}%
708   }%
709 }

```

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

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

```

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

```

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

```

`sxtrwrglossmark` Does nothing by default.

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

```

`debug` Provide extra debug options.

```

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

```

```

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

```

Pass all other options to glossaries.

```

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

```

Process options.

```
735 \ProcessOptionsX
```

Load glossaries if not already loaded.

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

Load the glossaries-accsupp package if required.

```
737 \@glsxtr@doaccsupp
```

Redefine \glspostdescription if required.

```
738 \@glsxtr@defpostpunc
```

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

```

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

```

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

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

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

```

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

`oindex@glossary`

```

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

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

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

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

```

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

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

```

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

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

```

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

Define abbreviations glossaries if required.

```

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

Setup shortcuts if required.

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

Redefine `\@glxtr@redef@forlgsentries` if required.

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

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

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

Now define the user command:

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

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

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

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

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

`saveentrycounter` Save original definition of `\@gls@saveentrycounter`.

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

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

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

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

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

```

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

```

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

sxtrdialecthook

```

839 \newcommand*{\@glsxtrdialecthook}{%
Set up record option if required.
840 \glsxtr@setup@record
Disable preamble-only options and switch on the undefined tag at the start of the docu-
ment.
841 \AtBeginDocument{%
842   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
843   \def\@glsxtrundeftag{\glsxtrundeftag}%
844 }

```

1.2 Extra Utilities

usedOrUndefined

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

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

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

trifemptyglossary

```
\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 `\glo@list@<type>`. If this hasn't been defined, the glossary doesn't exist. If it has been defined and is simply a comma, the glossary exists and is empty. (It's initialised to a comma.)

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

xtrifkeydefined

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

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

rovidestoragekey

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

```
863 \newcommand*{\glstrprovidestoragekey}{%
864   \@ifstar\@sglsxtr@provide@storagekey\@glstr@provide@storagekey
865 }
```

vide@storagekey

Unstarred version.

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

```

```

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

```

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

```

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

```

Provide the no-link command if not already defined.

```

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

```

`\provide@storagekey` Starred version.

```

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

```

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

`\GlsXtrFmtField`

```

901 \newcommand{\GlsXtrFmtField}{useri}

```

`\DefaultOptions`

```

902 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```


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

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

`\s@@glsxtrfmt` Pick up final optional argument.
 909 `\def\s@@glsxtrfmt#1#2#3[#4]{\@glsxtrfmt{#1}{#2}{#3}{#4}}`

`\@@glsxtrfmt` Actual inner working.
 910 `\newcommand*{\@@glsxtrfmt}[4]{%`
 Since there's no post-link hook to worry about, grouping can be added to provide some protection against nesting (but in general nested link text should be avoided).

```

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

```

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

```

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

```

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

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

```

\glstxtrentryfmt No link or indexing.
934 \ifdef\teorpdfstring
935 {
936   \newcommand*\glstxtrentryfmt[2]{%
937     \teorpdfstring{\@glstxtrentryfmt{#1}{#2}}{#2}%
938   }
939 }
940 {
941   \newcommand*\glstxtrentryfmt{\@glstxtrentryfmt}
942 }

```

```

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

```

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

```

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

```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```

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

```

`trfieldlistead` Similarly but uses `\listcseadd`.

```

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

```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```

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

```

Now provide commands to iterate over these lists.

fielddolistloop

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

fieldforlistloop

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

List element tests:

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

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

rfieldxifinlist Expands item.

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

lsxtrforcsvfield

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

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

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

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

lsxtrifhasfield Unstarred version adds grouping.

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

lsxtrifhasfield Starred version omits grouping.

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

rIfFieldNonZero Designed for numeric fields.

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

rIfFieldNonZero

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

sXtrIfFieldEqNum

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

Designed for numeric fields.

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

XtrIfFieldEqNum

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

XtrIfFieldEqNum

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

XtrIfFieldCmpNum

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

Designed for numeric fields.

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

trIfFieldCmpNum

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

trIfFieldCmpNum

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

sXtrIfFieldUndef

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

Just uses \ifcsundef.

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

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

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

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

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

```

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

```

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

```

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

```

`entryparentname`

```

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

```

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

```

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

```

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

```

1080 \newcommand*{\glsxtredefield}[2]{\protected@csedef{glo@glsdetoklabel{#1}@#2}}

```

`etfieldifexists`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`xGlsXtrSetField`

```

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

```

`eGlsXtrSetField`

```

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

```

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

```

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

```

`XtrIfFieldEqStr`

```

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

```

```

1117 }%
1118 {#5}%
1119 }

```

XtrIfFieldEqStr

```

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

```

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

```

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

```

rIfFieldEqXpStr

```

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

```

rIfFieldEqXpStr

```

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

```

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

```

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

```

fXpFieldEqXpStr

```

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

```



```

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

```

fXpFieldEqXpStr

```

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

```

lsXtrForeignText

```
\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`.

```

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

```

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

```

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

```

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

```

1183         \ifundef\TrackedDialectClosestSubMatch
1184         {%

```

```

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

```

```

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

```

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

```
1238 \newcommand*{\GlsXtrForeignTextField}{userii}
```

`\nDialectWarning`

```

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

```

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

```

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

```

```

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

```

lossary preamble

```

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

```

lossary preamble

```

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

```

1.3 Modifications to Commands Provided by glossaries

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

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

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

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

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

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

`ewglossaryentry`

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

`ewglossaryentry` Starred version.

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

`ewglossaryentry` Unstarred version.

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

```

The following is different from the base glossaries.sty:

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

```

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

```

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

```

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

```

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

`\ignoredglossary` Redefine to check for star.

```

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

```

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

```

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

```

`\ignoredglossary` Starred form.

```

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

```

```

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

```

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

```

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

```

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

```

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

```

ignoredglossary Unstarred version.

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

ignoredglossary Starred form.

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


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

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

1.3.1 Existence Checks

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

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

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

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

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

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

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

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

```

```

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

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

`\glxtr@usesee`

```

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

```

`\@glxtr@usesee`

```

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

```

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

```

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

```

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

```

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

```

`\glxtrhiername`

`\glxtrhiername{<label>}`

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

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

`\Glsxtrhiername`

`\Glsxtrhiername{<label>}`

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

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

`\GlsXtrhiername`

`\GlsXtrhiername{<label>}`

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

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

`\GLSxtrhiername`

`\GLSxtrhiername{<label>}`

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

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

`\GLSXTRhiername`

`\GLSXTRhiername{<label>}`

As above but displays all names in all-caps.

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

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

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

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

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

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

```
1624 \newcommand*{\glxtruseseealsoformat}[1]{%
1625   \glssseeformat[\seealsoname]{#1}{}%
1626 }
```

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

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

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

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

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

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

Add the cross-reference class definition to the hook.

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

Append to class list.

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

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

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

```

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

```

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

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

```

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

```

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

```

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

```

Add to the key mappings.

```

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

```

Set the default value.

```

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

```

Assign the field values.

```

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

```

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

```

1684 \ifdefvoid\@glo@alias

```

```

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

```

Provide user-level commands to access the values.

`\glxtralias`

```

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

```

`trseealsolabels`

```

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

```

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

```

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

```

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

```

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

```

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

`\glxtralias`

```

1714 \glsaddstoragekey*{alias}{\glxtralias}

```

`trseealsolabels`

```

1715 \glsaddstoragekey*{seealso}{\glxtrseealsolabels}

```

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

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

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

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

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

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

```
1738 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}
```

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

```
1739 \newcommand*\@glxtraddallcrossrefs{%
1740   \forallglossaries{\@glo@type}%
1741   {%
1742     \forglentries[\@glo@type]{\@glo@label}%
1743     {%
1744       \ifglused{\@glo@label}%
1745       {\expandafter\@glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1746     }%
1747   }%
1748 }
```

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

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

```

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

```

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

```

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

```

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

```

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

```

`glsxtrunusedformat`

```

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

```

1.3.2 Document Definitions

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

```

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

```

```

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

```

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

```

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

```

Write all currently defined entries.

```

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

```

m@glossaryentry

```

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

```

noidxglossaries

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

```

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

```

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

```

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

```

```

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

```

Check for docdefs=restricted:

```

1830 \ifglsxtrdocdefrestricted

```

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

```

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

```

Disable document definitions.

```

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

```

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

```

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

```

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

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

Restricted mode just needs to allow the see value.

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

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

`rEnableOnTheFly`

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

`rEnableOnTheFly`

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

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

`sxtrstarflywarn`

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

rEnableOnTheFly

```
1896 \newcommand*{\@GlsXtrEnableOnTheFly}{%
```

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

These definitions are all assigned the category given by:

```
\glsxtrcat
```

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

```
\glsxtr
```

```
1898 \newcommand*{\glsxtr}[1] [] {%
```

```
1899 \def\glsxtr@keylist{##1}%
```

```
1900 \@glsxtr
```

```
1901 }
```

```
\@glsxtr
```

```
1902 \newcommand*{\@glsxtr}[2] [] {%
```

```
1903 \ifglsentryexists{##2}%
```

```
1904 {%
```

```
1905 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
```

```
1906 }%
```

```
1907 {%
```

```
1908 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
```

```
1909 description={\nopostdesc},##1}%
```

```
1910 }%
```

```
1911 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
```

```
1912 }
```

```
\Glsxtr
```

```
1913 \newcommand*{\Glsxtr}[1] [] {%
```

```
1914 \def\glsxtr@keylist{##1}%
```

```
1915 \@Glsxtr
```

```
1916 }
```

```
\@Glsxtr
```

```
1917 \newcommand*{\@Glsxtr}[2] [] {%
```

```
1918 \ifglsentryexists{##2}%
```

```
1919 {%
```

```
1920 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
```

```
1921 }%
```

```
1922 {%
```

```
1923 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
```

```
1924 description={\nopostdesc},##1}%
```

```
1925 }%
```

```
1926 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
```

```
1927 }
```

`\glsxtrpl`

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

`\@glsxtrpl`

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

`\Glsxtrpl`

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

`\@Glsxtrpl`

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

`\GlsXtrWarning`

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

Disable commands after the glossary:

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

```

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

```

abledflycommand

```

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

```

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

```

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

```

1.3.3 Existing Glossary Style Modifications

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

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

```

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

```

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

etglossarystyle

```

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

```

Only set the current style if it exists.

```

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

```



```

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

```

In case we have an old version of glossaries:

```

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

```

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

```

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

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

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

```

and for `\glspagelistwidth`:

`lspagelistwidth`

```

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

```

```

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

```

aryentrynumbers Has the nonnumberlist option been used?

```

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

```

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

```

2064 \newcommand*{\GlsXtrFormatLocationList}[1]{#1}

```

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

ePreLocationTag

```

2065 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
2066 \let\@glsxtrpreloctag\@glsxtrpreloctag
2067 \let\@glsxtrpostloctag\@glsxtrpostloctag
2068 \renewcommand*{\@glsxtr@pagetag}{#1}%
2069 \renewcommand*{\@glsxtr@pagestag}{#2}%
2070 \renewcommand*{\@glsxtr@savepreloctag}[2]{%

```

```

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

```

glxtrpreloctag

```

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

```

glxtrpreloctag

```

2101 \newcommand*{@glxtrpreloctag}{}

```

@glxtr@pagetag

```

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

```

glxtr@pagetag

```

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

```

lsxtrpostloctag

```

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

```

```

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

```

lsxtrpostloctag

```

2111 \newcommand*\@glxtrpostloctag{}

```

lsxtr@preloctag

```

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

```

glxtr@doloctag

```

2115 \newcommand*\@glxtr@doloctag{}

```

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

```

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

```

1.3.4 Entry Formatting, Hyperlinks and Indexing

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

```

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

```

```

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

```

`sxtrregularfont` Font used for regular entries.

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

`bbreviationfont` Font used for abbreviation entries.

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

Commands like `\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.

```
2145 \renewcommand{\@gls@field@link}[4] [] {%
```

If the `record` option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

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

```

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

```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

Post-link hook commands need initialising.

```

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

```

`glsxtr@wrgloss` Set the default if the `wrgloss` is omitted.

```

2221 \newcommand*{\glsxtr@wrgloss}{%
2222 \glsifattribute{\glslabel}{wrgloss}{after}%
2223 {%
2224 \glsxtr@wrglossbeforefalse
2225 }%
2226 {%
2227 \glsxtr@wrglossbeforetrue
2228 }%
2229 }

```

`gls@wrglossbefore` Conditional to determine if the indexing should be done before the link text.

```

2230 \newif\ifglsxtr@wrglossbefore
2231 \glsxtr@wrglossbeforetrue

```

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

```

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

```

```

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

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

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

```

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

```

2244 \define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
2245 \glsxtr@hyperoutsidetrue

```

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

```

2246 \define@key{glslink}{textformat}{%
2247   \ifcsdef{#1}
2248   {%
2249     \letcs{\@glsxtr@local@textformat}{#1}%
2250   }%
2251   {%
2252     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2253   }%
2254 }

```

```

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

```

`nithyperoutside` Set the default if the hyperoutside is omitted.

```

2256 \newcommand*{\glsxtrinithyperoutside}{%
2257   \glsifattribute{\glslabel}{hyperoutside}{false}%
2258   {%
2259     \glsxtr@hyperoutsidetrue
2260   }%
2261   {%
2262     \glsxtr@hyperoutsidetrue
2263   }%
2264 }

```

`r@inc@linkcount` Does nothing by default.

```

2265 \newcommand*{\glsxtr@inc@linkcount}{}

```

`slinkpresetkeys` User hook performed immediately before options are set. Does nothing by default.

```

2266 \newcommand*{\glslinkpresetkeys}{}

```


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

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

`\@gls@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`.

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

`\@glsxtr@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`).

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

`\GlsXtrAutoAddOnFormat`

```
\GlsXtrAutoAddOnFormat [<label>]{<format list>}{<glsadd options>}
```

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

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

`\@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`.

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

```

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

  Save current value of \glslinkprefix:
2298 \let\@glsxtr@org@glslinkprefix@glslinkprefix
  Initialise \@glsxtr@local@textformat
2299 \let\@glsxtr@local@textformat\relax
  Initialise thevalue and theHvalue (v1.19).
2300 \def\@glsxtr@thevalue{}%
2301 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  Initialise when indexing should occur (new to v1.14).
2302 \glsxtrinitwrgloss
  Initialise whether \hyperlink should be outside \gls@textformat (new to v1.21).
2303 \glsxtrinithyperoutside
  Note that the default link options may override \glsxtrinitwrgloss.
2304 \@gls@setdefault@glslink@opts
  Increment link counter if enabled (new to v1.26).
2305 \glsxtr@inc@linkcount
  Check if the equations option has been set (new to v1.37).
2306 \if@glsxtr@equations
2307   \@glsxtr@use@equation@counter
2308 \fi
  As the original definition.
2309 \do@gls@disablehyperinlist
2310 \do@gls@link@checkfirsthyper
  User hook before options are set (new to v1.26):
2311 \glslinkpresetkeys
  Set options.
2312 \setkeys{glslink}{#1}%
  Perform auto add if set (new to v1.37)
2313 \glsxtr@do@autoadd{glslink}%
  User hook after options are set:
2314 \glslinkpostsetkeys
  Check thevalue and theHvalue before saving (v1.19).
2315 \ifdefempty{\@glsxtr@thevalue}%
2316 {%
2317   \@gls@saveentrycounter
2318 }%
2319 {%

```

```

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

Check if the textformat key has been used.
2324 \ifx\@glstr@local@textformat\relax

Check textformat attribute (new to v1.21).
2325 \gls@hasattribute{\glslabel}{textformat}%
2326 {%
2327 \edef\@glstr@attrval{\gls@getattribute{\glslabel}{textformat}}%
2328 \ifcsdef{\@glstr@attrval}%
2329 {%
2330 \letcs{\@glstr@textformat}{\@glstr@attrval}%
2331 }%
2332 {%
2333 \GlossariesExtraWarning{Unknown control sequence name
2334 '\@glstr@attrval' supplied in textformat attribute
2335 for entry '\glslabel'. Reverting to default \string\gls@textformat}%
2336 \let\@glstr@textformat\gls@textformat
2337 }%
2338 }%
2339 {%
2340 \let\@glstr@textformat\gls@textformat
2341 }%
2342 \else
2343 \let\@glstr@textformat\@glstr@local@textformat
2344 \fi

```

Do write if it should occur before the link text:

```

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

```

Do the link text:

```

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

```

Do write if it should occur after the link text:

```

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

```

Restore original value of \glolinkprefix:

```
2365 \let\glolinkprefix\@glsextr@org@glolinkprefix
```

As the original definition:

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

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

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

lsaddpresetkeys

```
2370 \newcommand*{\glsaddpresetkeys}{}
```

saddpostsetkeys

```
2371 \newcommand*{\glsaddpostsetkeys}{}
```

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

```

2372 \renewrobustcmd*{\glsadd}[2][]{%
2373   \glsextrifinmark
2374   {}%
2375   {%
2376     \@gls@adjustmode
2377     \begingroup
2378     \@glsextr@record{#1}{#2}{glossadd}%
2379     \glsdoifexists{#2}%
2380     {%
2381       \let\@glsnumberformat\@glsextr@defaultnumberformat
2382       \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
2383       \def\@glsextr@thevalue{}%
2384       \def\@glsextr@theHvalue{\@glsextr@thevalue}%

```

Implement any default settings (before options are set)

```

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

```

Implement any default settings (after options are set)

```

2387     \glsaddpostsetkeys
2388     \ifdefempty{\@glsextr@thevalue}%
2389     {%
2390       \@gls@saveentrycounter
2391     }%
2392     {%
2393       \let\theglentrycounter\@glsextr@thevalue
2394       \def\theHglentrycounter{\@glsextr@theHvalue}%
2395     }%

```

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

```
2396     \@gls@setsort{#2}%  
Ensure that indexing occurs (since that's the point of \glsadd). If indexing has been switched  
off by default, don't want the setting to affect \glsadd. The ignored format \glsignore can  
be used for selection without location, but the indexing still needs to be performed.  
2397     \KV@glslink@noindexfalse  
2398     \@do@wrglossary{#2}%  
2399     }%  
2400 \endgroup  
2401 }%  
2402 }
```

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

```
2403 \newrobustcmd{\glsaddeach}[2] [] {%  
2404   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%  
2405 }
```

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

```
2406 \newcommand*{\@glsxtr@field@linkdefs}{%  
2407   \let\glsxtrifwasfirstuse\@secondoftwo  
2408   \let\glsifplural\@secondoftwo  
2409   \let\glscapscase\@firstofthree  
2410   \let\glsinsert\@empty  
2411 }
```

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

`assignfieldfont`

```
2412 \newcommand*{\glsxtrassignfieldfont}[1] {%  
2413   \ifglentryexists{#1}%  
2414   {%  
2415     \ifglshasshort{#1}%  
2416     {%  
2417       \glssetabbrvfmt{\glscategory{#1}}%  
2418       \glsifregular{#1}%  
2419       {\let\@gls@field@font\glsxtrregularfont}%  
2420       {\let\@gls@field@font\@firstofone}%  
2421     }%  
2422     {%  
2423       \glsifnotregular{#1}%  
2424       {\let\@gls@field@font\@firstofone}%  
2425       {\let\@gls@field@font\glsxtrregularfont}%  
2426     }%  
2427   }%  
2428   {%  
2429     \let\@gls@field@font\@gobble  
2430   }%  
2431 }
```

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

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

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

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

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

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

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

`ecknohyperfirst`

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

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

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

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

```
2457 \def\@Glsfirst@#1#2[#3]{%
2458   \glstrassignfieldfont{#2}%
2459   \@gls@field@link
2460   [\let\gls@trifwasfirstuse\@firstoftwo
2461   \let\gls@scapscase\@secondofthree
2462   \glstrchecknohyperfirst{#2}%
2463   ]{#1}{#2}%
2464   {\@Gls@field@font{\Gls@accessfirst{#2}#3}}%
2465 }
```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

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

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

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

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

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

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

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

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



```

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

```

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

```

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

```

`\@glsdesc@`

```

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

```

`\@Glsdesc@` First letter uppercase version.

```

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

```

`\@GLSdesc@` All uppercase version.

```

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

```

`@glsdescplural@` No case-changing version.

```

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

```

`@Glsdescplural@` First letter uppercase version.

```

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

```

@GLSdescplural@ All uppercase version.

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

\@glsymbol@

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

\@GLssymbol@ First letter uppercase version.

```
2584 \def\@GLssymbol@#1#2[#3]{%
2585   \glstrassignfieldfont{#2}%
2586   \@gls@field@link
2587   [\let\gls@scaps@case\@secondoftwo]%
2588   {#1}{#2}{\@gls@field@font{\GLS@accesssymbol{#2}#3}}%
2589 }
```

\@GLSsymbol@ All uppercase version.

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

lssymbolplural@ No case-changing version.

```
2595 \def\@glsymbolplural@#1#2[#3]{%
2596   \glstrassignfieldfont{#2}%
2597   \@gls@field@link
2598   [\let\gls@scaps@case\@secondoftwo
2599   \let\gls@sifplural\@firstoftwo
2600   ]{#1}{#2}{\@gls@field@font{\gls@accesssymbolplural{#2}#3}}%
2601 }
```

lssymbolplural@ First letter uppercase version.

```
2602 \def\@GLssymbolplural@#1#2[#3]{%
2603   \glstrassignfieldfont{#2}%
2604   \@gls@field@link
2605   [\let\gls@scaps@case\@secondoftwo
2606   \let\gls@sifplural\@firstoftwo
2607   ]{#1}{#2}{\@gls@field@font{\GLS@accesssymbolplural{#2}#3}}%
2608 }
```

LSSymbolplural@ All uppercase version.

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

\@Glsuseri@ First letter uppercase version.

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

\@GLSuseri@ All uppercase version.

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

\@Glsuserii@ First letter uppercase version.

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

\@GLSuserii@ All uppercase version.

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

\@Glsuseriii@ First letter uppercase version.

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

```

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

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

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

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

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

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

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

```

```

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

```

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

`\@acrshort` No case change.

```

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

```

`\@Acrshort` First letter uppercase.

```

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

```

`\@ACRshort` All uppercase.

```

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

```

```

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

```

\@acrshortpl No case change.

```

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

```

\@Acrshortpl First letter uppercase.

```

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

```

\@ACRshortpl All uppercase.

```

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

```

```

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

```

\@acrlong No case change.

```

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

```

\@Acrlong First letter uppercase.

```

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

```

\@ACRlong All uppercase.

```

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

```

```

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

```

\@acrlongpl No case change.

```

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

```

\@Acrlongpl First letter uppercase.

```

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

```

\@ACRlongpl All uppercase.


```

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

```

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

`\@glsaddkey`

```

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

```

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

```

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

```

```

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

```

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

```

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

```

Finally the all caps version (modified):

```

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

```

```

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

```

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

```

2944 \providecommand*\@gls@link@nocheckfirsthyper{}

```

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

```

2945 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2946 \renewcommand*\@gls@link@checkfirsthyper{%

```

\ifglsused isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since \@gls@link@checkfirsthyper is only used by commands like \gls but not by other commands, this seems the best place to put it to automatically set the value for the commands that change the first use flag. The other commands should set \glsxtrifwasfirstuse to \@secondoftwo (which is done in \@glsxtr@field@linkdefs).

```

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

```

Store the category label for convenience.

```

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

```

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

```

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

```

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

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

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

`lt@glslink@opts`

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

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

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

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

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

`setaliasnoindex`

```
2990 \newcommand*\@glsxtrsetaliasnoindex}{%
2991 \glsxtrifhasfield{alias}{\glslabel}}%
2992 {%
2993   \let\glsxtrindexaliased\@glsxtrindexaliased
2994   \glsxtrsetaliasnoindex
2995   \let\glsxtrindexaliased\@no@glsxtrindexaliased
2996 }%
2997 {}}%
2998 }
```

`xtrindexaliased`

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

```

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

```

xtrindexaliased

```

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

```

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

```

3015 \let\glsxtrindexaliased\@no@glsxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

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

```

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

```

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

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

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

```

3043 }

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

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

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

Similarly for the “noidx” version:

`s@noidxglossary`

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

`xtr@do@@wrindex`

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

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

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

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

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

`alt@gls@hyp@opt` User version

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

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

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

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

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

`\rSetAltModifier`

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

Let `\bib2gls` know the modifier.

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

`\org@dohyperlink`

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

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

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

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

Scope:

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

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

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

```

3097 {%
3098   \glshasattribute{\glslabel}{targetname}%
3099   {%
3100     \glshasattribute{\glslabel}{targetcategory}%
3101     {%
3102       \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
3103         {\glsgetattribute{\glslabel}{targetcategory}}%
3104         {\glsgetattribute{\glslabel}{targetname}}%
3105         {\{\glsxtrprotectlinks#2}}}%
3106       }%
3107     {%
3108       \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
3109         {}%
3110         {\glsgetattribute{\glslabel}{targetname}}%
3111         {\{\glsxtrprotectlinks#2}}}%
3112       }%
3113     }%
3114   {%
3115     \href{\glsgetattribute{\glslabel}{targeturl}}{%
3116       {\{\glsxtrprotectlinks#2}}}%
3117     }%
3118   }%
3119   {%

```

Check for alias.

```

3120   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
3121   \ifdefvoid\gloaliaslabel
3122   {%
3123     \glsxtrhyperlink{#1}{\{\glsxtrprotectlinks#2}}%
3124     }%
3125   {%

```

Redirect link to the alias target.

```

3126     \glsxtrhyperlink
3127     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
3128     {\{\glsxtrprotectlinks#2}}%
3129     }%
3130   }%
3131 }

```

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

```

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

```


3141 }

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

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

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

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

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

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

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

```
3160 \def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}
```

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

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

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

```

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

```

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

@glxtr@p@text@

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

@Glsxtr@p@text@

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

@GLSxtr@p@text@

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

lsxtr@p@plural@

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

lsxtr@p@plural@

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

LSxtr@p@plural@

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

glsxtr@p@short@

```
3208 \def\@glsxtr@p@short@#1#2[#3]{%  
3209 {%  
3210 \glssetabbrvfmt{\glscategory{#2}}%  
3211 \glsabbrvfont{\glsentryshort{#2}}#3%  
3212 }%  
3213 }
```

Glsxtr@p@short@

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

GLSxtr@p@short@

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

sxtr@p@shortpl@

```
3226 \def\@glsxtr@p@shortpl@#1#2[#3]{%  
3227 {%  
3228 \glssetabbrvfmt{\glscategory{#2}}%  
3229 \glsabbrvfont{\glsentryshortpl{#2}}#3%  
3230 }%  
3231 }
```

Sxtr@p@shortpl@

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

Sxtr@p@shortpl@

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

```

3239  {%
3240    \glssetabbrvfmt{\glscategory{#2}}%
3241    \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
3242  }%
3243 }

```

@glsxtr@p@long@

```
3244 \def\@glsxtr@p@long@#1#2[#3]{\glsentrylong{#2}#3}
```

@Glsxtr@p@long@

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

@GLSxtr@p@long@

```
3246 \def\@GLSxtr@p@long@#1#2[#3]{%
3247   {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}
```

lsxtr@p@longpl@

```
3248 \def\@glsxtr@p@longpl@#1#2[#3]{\glsentrylongpl{#2}#3}
```

lSxtr@p@longpl@

```
3249 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsentrylongpl{#2}}#3}
```

LSxtr@p@longpl@

```
3250 \def\@GLSxtr@p@longpl@#1#2[#3]{%
3251   {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}
```

xtr@p@acrshort@

```
3252 \def\@glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\glsentryshort{#2}}#3}
```

xtr@p@acrshort@

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

xtr@p@acrshort@

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

r@p@acrshortpl@

```
3256 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glsentryshortpl{#2}}#3}
```

r@p@acrshortpl@

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

r@p@acrshortpl@

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

sxtr@p@acrlong@

```
3260 \def\@glsxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}
```

```

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

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

tr@p@acrlongpl@
3264 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

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

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

```

Commands to minimise conflict.

```

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

\glsxtrsetpopts Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3269 \newcommand*\glsxtrsetpopts[1]{%
3270 \renewcommand*\@glsxtrp@opt{#1}%
3271 }

\lossxtrsetpopts Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3272 \newcommand*\glossxtrsetpopts{%
3273 \glsxtrsetpopts{noindex}%
3274 }

\@@glsxtrp
3275 \newrobustcmd*\@@glsxtrp[2]{%
Add scope.
3276 {%
3277 \let\glspostlinkhook\relax
3278 \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
3279 }%
3280 }

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

```

```

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

```

\@Glsxtrp

```

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

```

\@GLSxtrp

```

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

```

\glsxtr@entry@p

```

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

```

`\glsxtrp` Not robust as it needs to expand somewhat.

```

3338 \ifdef\teorpdfstring
3339 {
3340 \newcommand{\glsxtrp}[2]{%
3341 \protect\NoCaseChange
3342 {%
3343 \protect\teorpdfstring
3344 {%
3345 \protect\glsxtrifinmark
3346 {%
3347 \ifcsdef{glsxtrhead#1}%
3348 {%
3349 {\protect\csuse{glsxtrhead#1}{#2}}%
3350 }%
3351 {%
3352 \glsxtr@headentry@p{#2}{#1}%
3353 }%
3354 }%
3355 {%
3356 \@glsxtrp{#1}{#2}%
3357 }%
3358 }%
3359 {%
3360 \protect\@gls@entry@field{#2}{#1}%
3361 }%
3362 }%
3363 }
3364 }
3365 {
3366 \newcommand{\glsxtrp}[2]{%
3367 \protect\NoCaseChange
3368 {%
3369 \protect\glsxtrifinmark
3370 {%
3371 \ifcsdef{glsxtrhead#1}%
3372 {%
3373 {\protect\csuse{glsxtrhead#1}}%
3374 }%
3375 {%

```

```

3376         \glsxtr@headentry@p{#2}{#1}%
3377     }%
3378 }%
3379 {%
3380     \@glsxtrp{#1}{#2}%
3381 }%
3382 }%
3383 }
3384 }

```

Provide short synonyms for the most common option.

`\glsps`

```
3385 \newcommand*{\glsps}{\glsxtrp{short}}
```

`\glspt`

```
3386 \newcommand*{\glspt}{\glsxtrp{text}}
```

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

```

3387 \ifdef\teorpdfstring
3388 {
3389     \newcommand{\Glsxtrp}[2]{%
3390         \protect\NoCaseChange
3391         {%
3392             \protect\teorpdfstring
3393             {%
3394                 \protect\glsxtrifinmark
3395                 {%
3396                     \ifcsdef{Glsxtrhead#1}%
3397                     {%
3398                         {\protect\csuse{Glsxtrhead#1}{#2}}%
3399                     }%
3400                     {%
3401                         \protect\@Gls@entry@field{#2}{#1}%
3402                     }%
3403                 }%
3404             }%
3405             \@Glsxtrp{#1}{#2}%
3406         }%
3407     }%
3408     {%
3409         \protect\@Gls@entry@field{#2}{#1}%
3410     }%
3411 }%
3412 }
3413 }
3414 {
3415     \newcommand{\Glsxtrp}[2]{%

```



```

3416 \protect\NoCaseChange
3417 {%
3418 \protect\glxtrifinmark
3419 {%
3420 \ifcsdef{Glsxtrhead#1}%
3421 {%
3422 {\protect\csuse{Glsxtrhead#1}}%
3423 }%
3424 {%
3425 \protect\@Gls@entry@field{#2}{#1}%
3426 }%
3427 }%
3428 {%
3429 \@Glsxtrp{#1}{#2}%
3430 }%
3431 }%
3432 }
3433 }

```

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

```

3434 \ifdef\teorpdfstring
3435 {
3436 \newcommand{\GLSxtrp}[2]{%
3437 \protect\NoCaseChange
3438 {%
3439 \protect\teorpdfstring
3440 {%
3441 \protect\glxtrifinmark
3442 {%
3443 \ifcsdef{GLSxtr#1}%
3444 {%
3445 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3446 }%
3447 {%
3448 \protect\mfirstucMakeUppercase
3449 {%
3450 \protect\@gls@entry@field{#2}{#1}%
3451 }%
3452 }%
3453 }%
3454 {%
3455 \@GLSxtrp{#1}{#2}%
3456 }%
3457 }%
3458 {%
3459 \protect\@gls@entry@field{#2}{#1}%
3460 }%
3461 }%
3462 }

```

```

3463 }
3464 {
3465   \newcommand{\GLSxtrp}[2]{%
3466     \protect\NoCaseChange
3467     {%
3468       \protect\glsxtrifinmark
3469       {%
3470         \ifcsdef{GLSxtr#1}%
3471         {%
3472           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3473         }%
3474         {%
3475           \protect\mfirstucMakeUppercase
3476           {%
3477             \protect\@gls@entry@field{#2}{#1}%
3478           }%
3479         }%
3480       }%
3481       {%
3482         \@GLSxtrp{#1}{#2}%
3483       }%
3484     }%
3485   }
3486 }

```

1.3.5 Entry Counting

The (use) entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine \gls etc to use \cgl instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceed 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.
3487 \newcommand*{\@glsxtr@unset}[1]{%
3488   \@@glsunset{#1}%
3489   \glsxtrpostunset{#1}%
3490 }%

```

```

\@glsunset  Global unset.
3491 \let\@glsunset\@glsxtr@unset

```

glsxtrpostunset

```
3492 \newcommand*\glsxtrpostunset}[1]{}
```

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

tUnsetBuffering

```
3493 \newcommand*\GlsXtrStartUnsetBuffering}{%
3494 \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3495 }
```

tUnsetBuffering Unstarred version doesn't check for duplicates.

```
3496 \newcommand*\@GlsXtrStartUnsetBuffering}{%
3497 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3498 \def\@glsxtr@unset@buffer{}%
3499 \let\@glsunset\@glsxtrbuffer@unset
3500 }
```

tUnsetBuffering Starred version checks for duplicates.

```
3501 \newcommand*\s@GlsXtrStartUnsetBuffering}{%
3502 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3503 \def\@glsxtr@unset@buffer{}%
3504 \let\@glsunset\@glsxtrbuffer@nodup@unset
3505 }
```

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

```
3506 \newcommand*\@glsxtrbuffer@unset}[1]{%
3507 \listxadd\@glsxtr@unset@buffer{#1}%
3508 }
```

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using \xifinlist as the added complexity might cause problems that the buffering is trying to overcome.)

```
3509 \newcommand*\@glsxtrbuffer@nodup@unset}[1]{%
3510 \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
3511 {\listxadd\@glsxtr@unset@buffer{#1}}}%
3512 }
```

pUnsetBuffering

```
3513 \newcommand*\GlsXtrStopUnsetBuffering}{%
3514 \ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3515 }
```

pUnsetBuffering Unstarred form (global unset).

```
3516 \newcommand*\@GlsXtrStopUnsetBuffering}{%
3517 \let\@glsunset\@glsxtr@unset
3518 \forlistloop\@glsunset\@glsxtr@unset@buffer
3519 \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3520 }
```

pUnsetBuffering Starred form (local unset).

```
3521 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3522   \forlistloop\@glslocalunset\@glxtr@unset@buffer
3523   \let\@glsunset\@glxtr@unset
3524 }
```

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

```
3525 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3526   \forlistloop#1\@glxtr@unset@buffer
3527 }
```

\@glslocalunset Local unset.

```
3528 \renewcommand*{\@glslocalunset}[1]{%
3529   \@glslocalunset{#1}%
3530   \glxtrpostlocalunset{#1}%
3531 }%
```

rpostlocalunset

```
3532 \newcommand*{\glxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
3533 \renewcommand*{\@glsreset}[1]{%
3534   \@glsreset{#1}%
3535   \glxtrpostreset{#1}%
3536 }%
```

glxtrpostreset

```
3537 \newcommand*{\glxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
3538 \renewcommand*{\@glslocalreset}[1]{%
3539   \@glslocalreset{#1}%
3540   \glxtrpostlocalreset{#1}%
3541 }%
```

rpostlocalreset

```
3542 \newcommand*{\glxtrpostlocalreset}[1]{}
```

slocalreseteach Locally reset a list of entries.

```
3543 \newcommand*{\glslocalreseteach}[1]{%
3544   \gls@ifnotmeasuring
3545   {%
3546     \@for\@gls@thislabel:=#1\do{%
3547       \glsdoifexists{\@gls@thislabel}%
3548       {%
3549         \@glslocalreset{\@gls@thislabel}%
3550       }%
3551     }%
```

```

3552 }%
3553 }

```

`glslocalunseteach` Locally unset a list of entries.

```

3554 \newcommand*{\glslocalunseteach}[1]{%
3555   \gls@ifnotmeasuring
3556   {%
3557     \@for\@gls@thislabel:=#1\do{%
3558       \glsdoifexists{\@gls@thislabel}%
3559       {%
3560         \@glslocalunset{\@gls@thislabel}%
3561       }%
3562     }%
3563   }%
3564 }

```

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

```

3565 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```

3566   \glsenableentrycount

```

Redefine `\gls` etc:

```

3567   \renewcommand*{\gls}{\cglsl}%
3568   \renewcommand*{\Gls}{\cGls}%
3569   \renewcommand*{\glspl}{\cglspl}%
3570   \renewcommand*{\Glspl}{\cGlspl}%
3571   \renewcommand*{\GLS}{\cGLS}%
3572   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3573   \@glsxtr@setentrycountunsetattr{#1}{#2}%

```

In case this command is used again:

```

3574   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3575   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3576     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3577       can't be used with \string\GlsXtrEnableEntryCounting}%
3578     {Use one or other but not both commands}}%
3579 }

```

`ycountunsetattr`

```

3580 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
3581   \@for\@glsxtr@cat:=#1\do
3582   {%
3583     \ifdefempty{\@glsxtr@cat}{}%
3584     {%
3585       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3586     }%

```

```
3587 }%
3588 }
```

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

nableentrycount

```
3589 \renewcommand*\glsenableentrycount}{%
```

Enable new fields:

```
3590 \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
```

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

```
3591 \renewcommand*\gls@defdocnewglossaryentry}{%
```

```
3592 \renewcommand*\newglossaryentry[2]{%
```

```
3593 \PackageError{glossaries}{\string\newglossaryentry\space
```

```
3594 may only be used in the preamble when entry counting has
```

```
3595 been activated}{If you use \string\glsenableentrycount\space
```

```
3596 you must place all entry definitions in the preamble not in
```

```
3597 the document environment}%
```

```
3598 }%
```

```
3599 }%
```

New commands to access new fields:

```
3600 \newcommand*\glsentrycurrcount}[1]{%
```

```
3601 \ifcsundef{glo@glsdetoklabel{##1}@currcount}%
```

```
3602 {0}{\@gls@entry@field{##1}{currcount}}%
```

```
3603 }%
```

```
3604 \newcommand*\glsentryprevcount}[1]{%
```

```
3605 \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
```

```
3606 {0}{\@gls@entry@field{##1}{prevcount}}%
```

```
3607 }%
```

Adjust post unset and reset:

```
3608 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
```

```
3609 \renewcommand*\glsxtrpostunset}[1]{%
```

```
3610 \@glsxtr@entrycount@org@unset{##1}%
```

```
3611 \@gls@increment@currcount{##1}%
```

```
3612 }%
```

```
3613 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
```

```
3614 \renewcommand*\glsxtrpostlocalunset}[1]{%
```

```
3615 \@glsxtr@entrycount@org@localunset{##1}%
```

```
3616 \@gls@local@increment@currcount{##1}%
```

```
3617 }%
```

```
3618 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
```

```
3619 \renewcommand*\glsxtrpostreset}[1]{%
```

```
3620 \@glsxtr@entrycount@org@reset{##1}%
```

```
3621 \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
```

```
3622 }%
```

```
3623 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
```

```
3624 \renewcommand*\glsxtrpostlocalreset}[1]{%
```

```
3625 \@glsxtr@entrycount@org@localreset{##1}%
```

```

3626 \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3627 }%

```

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

```

3628 \let\@cgl@s\@cgl@s@
3629 \let\@cgl@spl\@cgl@spl@

3630 \let\@cGls\@cGls@
3631 \let\@cGlspl\@cGlspl@
3632 \let\@cGLS\@cGLS@
3633 \let\@cGLSpl\@cGLSpl@

```

The rest is as the original definition.

```

3634 \AtEndDocument{\@gls@write@entrycounts}%
3635 \renewcommand*{\@gls@entry@count}[2]{%
3636 \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
3637 }%
3638 \let\glsenableentrycount\relax
3639 \renewcommand*{\glsenableentryunitcount}{%
3640 \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3641 can't be used with \string\glsenableentrycount}%
3642 {Use one or other but not both commands}%
3643 }%
3644 }

```

`\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.

```

3645 \renewcommand*{\@gls@write@entrycounts}{%
3646 \immediate\write\@auxout
3647 {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3648 \count@=0\relax
3649 \forallglsentries{\@glsentry}{%
3650 \gls@hasattribute{\@glsentry}{entrycount}%
3651 {%
3652 \ifglsused{\@glsentry}%
3653 {%
3654 \immediate\write\@auxout
3655 {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3656 }%
3657 }%
3658 \advance\count@ by \@ne
3659 }%
3660 }%
3661 }%
3662 \ifnum\count@=0
3663 \GlossariesExtraWarningNoLine{Entry counting has been enabled
3664 \MessageBreak with \string\glsenableentrycount\space but the
3665 \MessageBreak attribute 'entrycount' hasn't
3666 \MessageBreak been assigned to any of the defined

```

```

3667 \MessageBreak entries}%
3668 \fi
3669 }

```

```

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

```

```

3670 \newcommand*\glxtrifcounttrigger}[3]{%
3671 \glshasattribute{#1}{entrycount}%
3672 {%
3673 \ifnum\glssentryprevcount{#1}>\glsggetattribute{#1}{entrycount}\relax
3674 #3%
3675 \else
3676 #2%
3677 \fi
3678 }%
3679 {#3}%
3680 }

```

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

\@@cgl@

```

3681 \def\@@cgl@#1#2[#3]{%
3682 \glxtrifcounttrigger{#2}%
3683 {%
3684 \cglformat{#2}{#3}%
3685 \glssunset{#2}%
3686 }%
3687 {%
3688 \@gls@{#1}{#2}[#3]%
3689 }%
3690 }%

```

\@@cglsp1@

```

3691 \def\@@cglsp1@#1#2[#3]{%
3692 \glxtrifcounttrigger{#2}%
3693 {%
3694 \cglsp1format{#2}{#3}%
3695 \glssunset{#2}%
3696 }%
3697 {%
3698 \@glspl@{#1}{#2}[#3]%
3699 }%
3700 }%

```

\@@cGls@


```

3701 \def\@@cGls@#1#2[#3]{%
3702   \glxtrifcounttrigger{#2}%
3703   {%
3704     \cGlsformat{#2}{#3}%
3705     \glset{#2}%
3706   }%
3707   {%
3708     \@Gls@{#1}{#2}[#3]%
3709   }%
3710 }%

```

\@@cGlspl@

```

3711 \def\@@cGlspl@#1#2[#3]{%
3712   \glxtrifcounttrigger{#2}%
3713   {%
3714     \cGlsplformat{#2}{#3}%
3715     \glset{#2}%
3716   }%
3717   {%
3718     \@Glspl@{#1}{#2}[#3]%
3719   }%
3720 }%

```

\@@cGLS@

```

3721 \def\@@cGLS@#1#2[#3]{%
3722   \glxtrifcounttrigger{#2}%
3723   {%
3724     \cGLSformat{#2}{#3}%
3725     \glset{#2}%
3726   }%
3727   {%
3728     \@GLS@{#1}{#2}[#3]%
3729   }%
3730 }%

```

\@@cGLSpl@

```

3731 \def\@@cGLSpl@#1#2[#3]{%
3732   \glxtrifcounttrigger{#2}%
3733   {%
3734     \cGLSplformat{#2}{#3}%
3735     \glset{#2}%
3736   }%
3737   {%
3738     \@GLSpl@{#1}{#2}[#3]%
3739   }%
3740 }%

```

Remove default warnings from \cglS etc so that it can be used interchangeable with \glS etc.

```
\@cgl@s@
3741 \def\@cgl@s@#1#2[#3]{\@gls@{#1}{#2}[#3]}
```

```
\@cGls@
3742 \def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}
```

```
\@cglsp1@
3743 \def\@cglsp1@#1#2[#3]{\@glsp1@{#1}{#2}[#3]}
```

```
\@cGlsp1@
3744 \def\@cGlsp1@#1#2[#3]{\@Glsp1@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

```
\cGLS
3745 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

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

```
3746 \newcommand*{\@cGLS}[2][\%]
3747 \new@ifnextchar[\@cGLS@{#1}{#2}]{\@cGLS@{#1}{#2}[\%]}
3748 }
```

```
\@cGLS@
3749 \def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}
```

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

```
3750 \newcommand*{\cGLSformat}[2]{\%}
3751 \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}\%
3752 }
```

```
\cGLSp1
3753 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\@cGLSp1}
```

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

```
3754 \newcommand*{\@cGLSp1}[2][\%]
3755 \new@ifnextchar[\@cGLSp1@{#1}{#2}]{\@cGLSp1@{#1}{#2}[\%]}
3756 }
```

```
\@cGLSp1@
3757 \def\@cGLSp1@#1#2[#3]{\@GLSp1@{#1}{#2}[#3]}
```

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

```
3758 \newcommand*{\cGLSp1format}[2]{\%}
3759 \expandafter\mfirstucMakeUppercase\expandafter{\cglsp1format{#1}{#2}}\%
3760 }
```

Modify the trigger formats to check for the regular attribute.

`\cglsformat`

```
3761 \renewcommand*\cglsformat}[2]{%
3762   \glsifregular{#1}
3763   {\glsentryfirst{#1}}%
3764   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
3765 }
```

`\cGlsformat`

```
3766 \renewcommand*\cGlsformat}[2]{%
3767   \glsifregular{#1}
3768   {\Glsentryfirst{#1}}%
3769   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
3770 }
```

`\cglsplformat`

```
3771 \renewcommand*\cglsplformat}[2]{%
3772   \glsifregular{#1}
3773   {\glsentryfirstplural{#1}}%
3774   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
3775 }
```

`\cGlsplformat`

```
3776 \renewcommand*\cGlsplformat}[2]{%
3777   \glsifregular{#1}
3778   {\Glsentryfirstplural{#1}}%
3779   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
3780 }
```

New code similar to above for unit counting.

`defunitcounters`

```
3781 \newcommand*\@@newglossaryentry@defunitcounters{%
3782   \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
3783   \ifdefvoid\@glo@countunit
3784   {}%
3785   {%
3786     \@glsxtr@ifunitcounter{\@glo@countunit}%
3787     {}%
3788     {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
3789   }%
3790 }
```

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

```
3791 \newcommand*\@glsxtr@unitcountlist{}
```

@addunitcounter

```
3792 \newcommand*\@glsxtr@addunitcounter}[1]{%
3793 \listadd{\@glsxtr@unitcountlist}{#1}%
3794 \ifcsundef{glsxtr@theunit@#1}
3795 {%
3796 \ifcsdef{theH#1}%
3797 {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
3798 {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
3799 }%
3800 {}%
3801 }
```

r@ifunitcounter

```
3802 \newcommand*\@glsxtr@ifunitcounter}[3]{%
3803 \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
3804 }
```

urrentunitcount

```
3805 \newcommand*\@glsxtr@currentunitcount[1]{%
3806 glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
3807 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3808 }
```

eviousunitcount

```
3809 \newcommand*\@glsxtr@previousunitcount[1]{%
3810 glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
3811 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3812 }
```

t@currunitcount

```
3813 \newcommand*\@gls@increment@currunitcount}[1]{%
3814 \gls@hasattribute{#1}{unitcount}%
3815 {%
3816 \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3817 \ifcsundef{\@glsxtr@csname}%
3818 {%
3819 \csgdef{\@glsxtr@csname}{1}%
3820 \listcsxadd
3821 {glo@\glsdetoklabel{#1}@unitlist}%
3822 {\glsgetattribute{#1}{unitcount}.%
3823 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3824 }%
3825 }%
3826 {%
3827 \csxdef{\@glsxtr@csname}%
3828 {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3829 }%
3830 }%
3831 {}%
```

3832 }

t@currunitcount

```
3833 \newcommand*{\@gls@local@increment@currunitcount}[1]{%
3834   \glsattribute{#1}{unitcount}%
3835   {%
3836     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3837     \ifcsundef{\@glsxtr@csname}%
3838     {%
3839       \csdef{\@glsxtr@csname}{1}%
3840       \listcseadd
3841         {glo@glstetoklabel{#1}@unitlist}%
3842         {\glsgetattribute{#1}{unitcount}.%
3843         \csuse{glsxtr@theunit@glstetoklabel{#1}{unitcount}}%
3844         }%
3845     }%
3846     {%
3847       \csedef{\@glsxtr@csname}%
3848         {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3849     }%
3850   }%
3851   {}%
3852 }
```

r@currunitcount

```
3853 \newcommand*{\@glsxtr@currunitcount}[2]{%
3854   \ifcsundef
3855     {glo@glstetoklabel{#1}@currunit@#2}%
3856     {0}%
3857     {\csuse{glo@glstetoklabel{#1}@currunit@#2}}%
3858   }%
```

r@prevunitcount

```
3859 \newcommand*{\@glsxtr@prevunitcount}[2]{%
3860   \ifcsundef
3861     {glo@glstetoklabel{#1}@prevunit@#2}%
3862     {0}%
3863     {\csuse{glo@glstetoklabel{#1}@prevunit@#2}}%
3864   }%
```

eentryunitcount

```
3865 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3866   \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3867   \renewcommand*{\gls@defdocnewglossaryentry}{%
3868     \renewcommand*\newglossaryentry[2]{%
3869       \PackageError{glossaries}{\string\newglossaryentry\space
```

```

3870     may only be used in the preamble when entry counting has
3871     been activated}{If you use \string\glsenableentryunitcount\space
3872     you must place all entry definitions in the preamble not in
3873     the document environment}%
3874 }%
3875 }%

```

New commands to access new fields:

```

3876 \newcommand*{\glsentrycurrcount}[1]{%
3877   \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount} .%
3878   \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3879 }%
3880 \newcommand*{\glsentryprevcount}[1]{%
3881   \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount} .%
3882   \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3883 }%

```

Access total count:

```

3884 \newcommand*{\glsentryprevtotalcount}[1]{%
3885   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3886   {0}%
3887   {%
3888     \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3889   }%
3890 }%

```

Access max value:

```

3891 \newcommand*{\glsentryprevmaxcount}[1]{%
3892   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3893   {0}%
3894   {%
3895     \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3896   }%
3897 }%

```

Adjust post unset and reset:

```

3898 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
3899 \renewcommand*{\glsxtrpostunset}[1]{%
3900   \@glsxtr@entryunitcount@org@unset{##1}%
3901   \@gls@increment@currunitcount{##1}%
3902 }%
3903 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
3904 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3905   \@glsxtr@entryunitcount@org@localunset{##1}%
3906   \@gls@local@increment@currunitcount{##1}%
3907 }%
3908 \let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset
3909 \renewcommand*{\glsxtrpostreset}[1]{%
3910   \glsattribute{##1}{unitcount}%
3911   {%
3912     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%

```

```

3913     \ifcsundef{\@glxtr@csname}%
3914     {}%
3915     {\csgdef{\@glxtr@csname}{0}}%
3916     }%
3917     {}%
3918     }%
3919 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
3920 \renewcommand*\@glxtrpostlocalreset}[1]{%
3921   \@glxtr@entryunitcount@org@localreset{##1}%
3922   \glshasattribute{##1}{unitcount}%
3923   {%
3924     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3925     \ifcsundef{\@glxtr@csname}%
3926     {}%
3927     {\csdef{\@glxtr@csname}{0}}%
3928     }%
3929     {}%
3930     }%

```

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

```

3931 \let\@cgl@@\@cgl@
3932 \let\@cgl@pl@\@cgl@pl@

3933 \let\@cGl@\@cGl@
3934 \let\@cGl@pl@\@cGl@pl@
3935 \let\@cGL@\@cGL@
3936 \let\@cGL@pl@\@cGL@pl@

```

Write information to the aux file.

```

3937 \AtEndDocument{\@gls@write@entryunitcounts}%
3938 \renewcommand*\@gls@entry@unitcount}[3]{%
3939   \csgdef{glo@glstdetoklabel{##1}@prevunit@##3}{##2}%
3940   \ifcsundef{glo@glstdetoklabel{##1}@prevunittotal}%
3941   {\csgdef{glo@glstdetoklabel{##1}@prevunittotal}{##2}}%
3942   {%
3943     \csxdef{glo@glstdetoklabel{##1}@prevunittotal}{
3944       \number\numexpr\csuse{glo@glstdetoklabel{##1}@prevunittotal}+##2}%
3945     }%
3946     \ifcsundef{glo@glstdetoklabel{##1}@prevunitmax}%
3947     {\csgdef{glo@glstdetoklabel{##1}@prevunitmax}{##2}}%
3948     {%
3949       \ifnum\csuse{glo@glstdetoklabel{##1}@prevunitmax}<##2
3950       \csgdef{glo@glstdetoklabel{##1}@prevunitmax}{##2}%
3951       \fi
3952     }%
3953     }%
3954 \let\glsenableentryunitcount\relax
3955 \renewcommand*\glsenableentrycount{%
3956   \PackageError{glossaries-extra}{\string\glsenableentrycount\space

```

```

3957     can't be used with \string\glsenableentryunitcount}%
3958     {Use one or other but not both commands}%
3959   }%
3960 }
3961 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3962 \newcommand*{\@gls@entry@unitcount}[3]{ }

```

ryunitcounts@do

```

3963 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3964   \immediate\write\@auxout
3965   {\string\@gls@entry@unitcount
3966    {\@gls@entry}%
3967    {\@gls@xtr@currunitcount{\@gls@entry}{#1}%
3968    }%
3969    {#1}}%
3970 }

```

entryunitcounts

```

3971 \newcommand*{\@gls@write@entryunitcounts}{%
3972   \immediate\write\@auxout
3973   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3974   \count@=0\relax
3975   \forallglsentries{\@gls@entry}{%
3976     \gls@hasattribute{\@gls@entry}{unitcount}%
3977     {%
3978       \ifglsused{\@gls@entry}%
3979       {%
3980         \forlistcsloop
3981           {\@gls@write@entryunitcounts@do}%
3982           {glo@\gls@detoklabel{\@gls@entry}@unitlist}%
3983       }%
3984     }%
3985     \advance\count@ by \@ne
3986   }%
3987 }%
3988 }%
3989 \ifnum\count@=0
3990   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3991   \MessageBreak with \string\glsenableentryunitcount\space but the
3992   \MessageBreak attribute 'unitcount' hasn't
3993   \MessageBreak been assigned to any of the defined
3994   \MessageBreak entries}%
3995 \fi
3996 }

```

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


```

3997 \newcommand*\GlsXtrEnableEntryUnitCounting}[3]{%
  Enable entry counting:
3998   \glsenableentryunitcount
  Redefine \gls etc:
3999   \renewcommand*\gls{\cglss}%
4000   \renewcommand*\Gls{\cGls}%
4001   \renewcommand*\glspl{\cglsspl}%
4002   \renewcommand*\Glspl{\cGlspl}%
4003   \renewcommand*\GlsS{\cGlsS}%
4004   \renewcommand*\GlsSpl{\cGlsSpl}%
  Set the entrycount attribute:
4005   \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
  In case this command is used again:
4006   \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
4007   \renewcommand*\GlsXtrEnableEntryCounting}[2]{%
4008     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
4009       can't be used with \string\GlsXtrEnableEntryUnitCounting}%
4010     {Use one or other but not both commands}}%
4011 }

```

tcountunsetattr

```

4012 \newcommand*\@glsxtr@setentryunitcountunsetattr}[3]{%
4013   \@for\@glsxtr@cat:=#1\do
4014   {%
4015     \ifdefempty{\@glsxtr@cat}{}%
4016     {%
4017       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
4018       \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
4019     }%
4020   }%
4021 }

```

1.3.6 Acronym Modifications

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

nericNewAcronym

```

4022 \renewcommand*\SetGenericNewAcronym}{%
4023   \let\@Gls@entryname\@Gls@acrenryname
4024   \renewcommand*\newacronym}[4][ ]{%
4025     \ifdefempty{\@glsacronymlists}%

```

```

4026   {%
4027     \def\@glo@type{\acronymtype}%
4028     \setkeys{glossentry}{##1}%
4029     \DeclareAcronymList{\@glo@type}%
4030   }%
4031   {}%
4032   \glskeylisttok{##1}%
4033   \glslabeltok{##2}%
4034   \glsshorttok{##3}%
4035   \glslongtok{##4}%
4036   \newacronymhook
4037   \protected@edef\@do@newglossaryentry{%
4038     \noexpand\newglossaryentry{\the\glslabeltok}%
4039     {%
4040       type=\acronymtype,%
4041       name={\expandonce{\acronymentry{##2}}},%
4042       sort={\acronymstok{\the\glsshorttok}{\the\glslongtok}},%
4043       text={\the\glsshorttok},%
4044       short={\the\glsshorttok},%
4045       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4046       long={\the\glslongtok},%
4047       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4048       category=acronym,%
4049       \GenericAcronymFields,%
4050       \the\glskeylisttok
4051     }%
4052   }%
4053   \@do@newglossaryentry
4054 }%
4055 \renewcommand*{\acrfullfmt}[3]{%
4056   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}%
4057 \renewcommand*{\Acrfullfmt}[3]{%
4058   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}%
4059 \renewcommand*{\ACRfullfmt}[3]{%
4060   \glslink[##1]{##2}{%
4061     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
4062 \renewcommand*{\acrfullplfmt}[3]{%
4063   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}%
4064 \renewcommand*{\Acrfullplfmt}[3]{%
4065   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}%
4066 \renewcommand*{\ACRfullplfmt}[3]{%
4067   \glslink[##1]{##2}{%
4068     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
4069 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
4070 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
4071 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
4072 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
4073 }

```

This will cause a problem for glossaries that contain a mixture of acronyms and abbrevia-

tions, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```
4074 \let\@glxtr@org@setacronymstyle\setacronymstyle
4075 \let\@glxtr@org@newacronymstyle\newacronymstyle
```

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

```
4076 \newcommand*\MakeAcronymsAbbreviations}{%
4077   \renewcommand*\newacronym}[4] []{%
4078     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4079   }%
4080   \renewcommand*\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4081   \renewcommand*\acronymfont}[1]{\glsabbrvfont{##1}}%
4082   \renewcommand*\setacronymstyle}[1]{%
4083     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
4084     unavailable.
4085     Use \string\setabbreviationstyle\space instead.
4086     The original acronym interface can be restored with
4087     \string\RestoreAcronyms}{}%
4088   }%
4089   \renewcommand*\newacronymstyle}[1]{%
4090     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4091     available unless you restore the original acronym interface with
4092     \string\RestoreAcronyms}%
4093     \@glxtr@org@newacronymstyle{##1}%
4094   }%
4095 }
```

Switch acronyms to abbreviations:

```
4096 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```
4097 \newcommand*\RestoreAcronyms}{%
4098   \SetGenericNewAcronym
4099   \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
4100   \renewcommand*\acronymfont}[1]{##1}%
4101   \let\setacronymstyle\@glxtr@org@setacronymstyle
4102   \let\newacronymstyle\@glxtr@org@newacronymstyle
```

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

```
4103 \renewcommand*\@gls@link@checkfirsthyper{%
4104   \ifglsused{\glslabel}%
4105   {\let\glxtrifwasfirstuse\@secondoftwo}
4106   {\let\glxtrifwasfirstuse\@firstoftwo}%
4107   \@glxtr@org@checkfirsthyper
4108 }
4109 \glssetcategoryattribute{acronym}{regular}{false}%
```

```
4110 \setacronymstyle{long-short}%
4111 }
```

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

```
4112 \renewcommand*{\glsacspace}[1]{%
4113 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4114 \ifdim\dimen@<\glsacspacemax~\else\space\fi
4115 }
```

`\glsacspacemax` Value used in the above.

```
4116 \newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

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

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

`r@reg@glosslist`

```
4117 \newcommand*{\@glsxtr@reg@glosslist}{}

```

Save the original definition of `\makeglossaries`:

```
4118 \let\@glsxtr@org@makeglossaries\makeglossaries

```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with record.

`\makeglossaries`

```
4119 \renewcommand*{\makeglossaries}[1] [] {%
4120 \@glsxtr@if@record@only
4121 {%
4122 \PackageError{glossaries-extra}{\string\makeglossaries\space
4123 not permitted\MessageBreak with record=\@glsxtr@record@setting\space
4124 package option}%
4125 {You may only use \string\makeglossaries\space with
4126 record=off or record=alsoindex options}%
4127 }%
4128 {%
4129 \ifblank{#1}%
4130 {\@glsxtr@org@makeglossaries}%
4131 {%
4132 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
4133 \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4134 not permitted\MessageBreak with record=alsoindex package option}%
4135 {You may only use the hybrid \string\makeglossaries[...]\space with
4136 record=off option}%
4137 \else
```

`\@gls@@automake@immediate` was introduced to glossaries v4.42 so it may not be defined.

```
4138 \ifdef\@gls@@automake@immediate{\@gls@@automake@immediate}{}%
4139 \edef\@glsxtr@reg@glosslist{#1}%
4140 \ifundef{glswrite}{\newwrite\glswrite}{}%
4141 \protected@write\@auxout{\string\providecommand
4142 \string\@glsorder[1]}{}}
4143 \protected@write\@auxout{\string\providecommand
4144 \string\@istfilename[1]}{}}
4145 \protected@write\@auxout{\string\@istfilename{\istfilename}}%
4146 \protected@write\@auxout{\string\@glsorder{\glsorder}}
4147 \protected@write\@auxout{\string\glsxtr@makeglossaries{#1}}
4148 \write\@auxout{\string\providecommand\string\@gls@reference[3]}{}}%
```

Iterate through each supplied glossary type and activate it.

```
4149 \@for\@glo@type:=#1\do{%
4150 \ifempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4151 }%
```

New glossaries must be created before `\makeglossaries`:

```
4152 \renewcommand*\newglossary[4][]{%
4153 \PackageError{glossaries}{New glossaries
4154 must be created before \string\makeglossaries}{You need
4155 to move \string\makeglossaries\space after all your
4156 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
4157 \let\@makeglossary\relax
4158 \let\makeglossary\relax
4159 \renewcommand\makeglossaries[1][]{}%
```

Disable all commands that have no effect after `\makeglossaries`

```
4160 \@disable@onlypremakeg
```

Allow see key:

```
4161 \let\gls@checkseeallowed\relax
```

Adjust `\@do@seeglossary`. This needs to check for the entry's existence but don't increment associated counter.

```
4162 \renewcommand*\@do@seeglossary[2]{%
4163 \glsdoifexists{##1}%
4164 {%
4165 \edef\@gls@label{\glsdetoklabel{##1}}%
4166 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4167 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4168 {\@glsxtr@org@doseeglossary{##1}{##2}}%
4169 {%
4170 \@@glsxtrwrglossmark
4171 \protected@write\@auxout{\string\@gls@reference
4172 \string\@gls@reference
4173 {\@gls@type}{\@gls@label}{\string\glsseeformat##2}}%
4174 }%
```

```

4175         }%
4176         }%
4177     }%

Adjust \@do@wrglossary
4178     \let\@glxtr@do@wrglossary\@do@wrglossary
4179     \def\@do@wrglossary{%
4180         \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4181         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
4182         {\@glxtr@do@wrglossary}%
4183         {\@gls@noidxglossary}%
4184     }%

Suppress warning about no \makeglossaries
4185     \let\warn@nomakeglossaries\relax
4186     \def\warn@noprintglossary{%
4187         \GlossariesWarningNoLine{No \string\printglossary\space
4188         or \string\printglossaries\space
4189         found.^^J(Remove \string\makeglossaries\space if you don't want
4190         any glossaries.)^^JThis document will not have a glossary}%
4191     }%

Only warn for glossaries not listed.
4192     \renewcommand{\@gls@nofwarn}[1]{%
4193         \edef\@gls@type{##1}%
4194         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
4195         {%
4196             \GlossariesExtraWarning{Can't use
4197             \string\printnoidxglossary[type={\@gls@type}]
4198             when '\@gls@type' is listed in the optional argument of
4199             \string\makeglossaries}%
4200         }%
4201         {%
4202             \GlossariesWarning{Empty glossary for
4203             \string\printnoidxglossary[type={##1}].
4204             Rerun may be required (or you may have forgotten to use
4205             commands like \string\gls)}%
4206         }%
4207     }%

Adjust display number list to check for type:
4208     \renewcommand*\@glsdisplaynumberlist[1]{%
4209         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
4210         {\@glxtr@idx@displaynumberlist{##1}}%
4211         {\@glxtr@noidx@displaynumberlist{##1}}%
4212     }%

Adjust entry list:
4213     \renewcommand*\@glsentrynumberlist[1]{%
4214         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
4215         {\@glxtr@idx@entrynumberlist{##1}}%

```

```

4216     {\@glsxtr@noidx@entrynumberlist{##1}}}%
4217     }%

```

Adjust number list loop

```

4218     \renewcommand*\glsnumberlistloop}[2]{%
4219     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4220     {%
4221         \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4222         not available for glossary ‘##1’}{}%
4223     }%
4224     {\@glsxtr@noidx@numberlistloop{##1}{##2}}}%
4225     }%

```

Only sanitize sort for normal indexing glossaries.

```

4226     \renewcommand*\glsprestandardsort}[3]{%
4227     \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4228     {%
4229         \glsdosanitizesort
4230     }%
4231     {%
4232         \ifglssanitizesort
4233         \@gls@noidx@sanitizesort
4234         \else
4235         \@gls@noidx@nosanitizesort
4236         \fi
4237     }%
4238     }%

```

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

```

4239     \renewcommand*\new@glossaryentry[2]{%
4240     \PackageError{glossaries-extra}{Glossary entries must be defined
4241     in the preamble\MessageBreak when you use the optional argument
4242     of \string\makeglossaries}{Either move your definitions to the
4243     preamble or don't use the optional argument of
4244     \string\makeglossaries}%
4245     }%

```

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

```

4246     \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
4247     \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

4248     \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
4249     type=\glsdefaulttype,\@end@glsxtr@gettype
4250     \def\@glo@sorttype{\@glo@default@sorttype}%
4251     }%

```

Check automake setting:

```

4252     \ifglautomake

```

```

4253     \renewcommand*{\@gls@doautomake}{%
4254     \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
4255     \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
4256     }%
4257     }%
4258     \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

4259     \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4260     \fi
4261     }%
4262 }%
4263 }

```

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

`\orgprintglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the `glstex` file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

4264 \newcommand{\@glsxtr@orgprintglossary}[2]{%
4265 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

4266 \def\glossarytitle{%
4267 \ifcsdef{\@glo@type@\@glo@type @title}%
4268 {\csuse{\@glo@type@\@glo@type @title}}%
4269 {\glossaryname}}%
4270 \def\glossarytoctitle{\glossarytitle}%
4271 \let\org@glossarytitle\glossarytitle
4272 \def\@glossarystyle{%
4273 \ifx\@glossary@default@style\relax
4274 \GlossariesWarning{No default glossary style provided \MessageBreak
4275 for the glossary '\@glo@type'. \MessageBreak
4276 Using deprecated fallback. \MessageBreak
4277 To fix this set the style with \MessageBreak
4278 \string\setglossarystyle\space or use the \MessageBreak
4279 style key=value option}%
4280 \fi
4281 }%
4282 \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
4283 \let\@org@glossaryentrynumbers\glossaryentrynumbers
4284 \bgroup
4285 \@printgloss@setsort
4286 \setkeys{printgloss}{#1}%
4287 \ifx\glossarytitle\org@glossarytitle
4288 \else
4289 \cslet{\@glo@type@\@glo@type @title}{\glossarytitle}%
4290 \fi

```



```

4291 \let\currentglossary\@glo@type
4292 \let\org@glossaryentrynumbers\glossaryentrynumbers
4293 \let\glsnonextpages\@glsnonextpages
4294 \let\glsnextpages\@glsnextpages

4295 \glsxtractivatenopost
4296 \gls@dotoc@title
4297 \@glossarystyle
4298 \let\gls@org@glossaryentryfield\glossentry
4299 \let\gls@org@glossarysubentryfield\subglossentry
4300 \renewcommand{\glossentry}[1]{%
4301   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4302   \gls@org@glossaryentryfield{##1}%
4303 }%
4304 \renewcommand{\subglossentry}[2]{%
4305   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4306   \gls@org@glossarysubentryfield{##1}{##2}%
4307 }%
4308 \@gls@preglossaryhook
4309 #2%
4310 \egroup
4311 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4312 \global\let\warn@noprintglossary\relax
4313 }

```

`tractivatenopost` Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```

4314 \newcommand*{\glsxtractivatenopost}{%
4315   \let\nopostdesc\@nopostdesc
4316   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4317 }

```

`lsxtrnopostpunc`

```

4318 \newrobustcmd*{\glsxtrnopostpunc}{}

```

`sxtr@nopostpunc` Provide a command that works like `\nopostdesc` but only switches of the punctuation without suppressing the post-description hook.

```

4319 \newcommand{\@glsxtr@nopostpunc}{%
4320   \let\@@glsxtr@org@postdescription\glspostdescription
4321   \ifglsnopostdot
4322     \renewcommand{\glspostdescription}{%
4323       \glsnopostdottrue
4324       \let\glspostdescription\@@glsxtr@org@postdescription
4325       \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
4326       \glsxtrpostdescription
4327       \@glsxtr@nopostpunc@postdesc}%
4328   \else
4329     \renewcommand{\glspostdescription}{%
4330       \let\glspostdescription\@@glsxtr@org@postdescription
4331       \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc

```

```

4332 \glxtrpostdescription
4333 \@glxtr@nopostpunc@postdesc}%
4334 \fi
4335 \glsnopostdotfalse
4336 }

```

stpunc@postdesc

```
4337 \newcommand*{\@glxtr@nopostpunc@postdesc}{}
```

estore@postpunc

```

4338 \newcommand*{\@glxtr@restore@postpunc}{%
4339 \def\@glxtr@nopostpunc@postdesc{%
4340 \@glxtr@org@postdescription
4341 \let\@glxtr@nopostpunc@postdesc\@empty
4342 \let\glxtrrestorepostpunc\@empty
4343 }%
4344 }

```

restorepostpunc Does nothing outside of glossary.

```
4345 \newcommand*{\glxtrrestorepostpunc}{}
```

\@printglossary Redefine.

```

4346 \renewcommand{\@printglossary}[2]{%
4347 \def\@glxtr@printglossopts{#1}%
4348 \@glxtr@orgprintglossary{#1}{#2}%
4349 }

```

Add a key that switches off the entry targets:

```

4350 \define@choicekey{printgloss}{target}
4351 [\@glxtr@printglossval\@glxtr@printglossnr]%
4352 {true,false}[true]%
4353 {%
4354 \ifcase\@glxtr@printglossnr
4355 \def\@glstarget{\glsdohypertarget}%
4356 \else
4357 \let\@glstarget\@secondoftwo
4358 \fi
4359 }

```

hypernameprefix

```
4360 \newcommand*{\@glxtrhypernameprefix}{}
```

New to v1.20:

```

4361 \define@key{printgloss}{targetnameprefix}{%
4362 \renewcommand*{\@glxtrhypernameprefix}{#1}%
4363 }

```

```

4364 \define@key{printgloss}{prefix}{%
4365   \renewcommand{\glolinkprefix}{#1}%
4366 }

```

```

4367 \define@key{printgloss}{label}{%
4368   \glxtrsetglossarylabel{#1}%
4369 }

```

`etglossarylabel` Set the label for subsequent glossaries. If the label is fixed (that is, doesn't change with each glossary) this will need to be scoped or changed again to prevent duplicate labels.

```

4370 \newcommand{\glxtrsetglossarylabel}[1]{%
4371   \renewcommand*{\@glossaryseclabel}{%
4372     \protected@edef\@currentlabelname{\glossarytoctitle}%
4373     \label{#1}%
4374   }%
4375 }

```

`lstdohypertarget` Redefine to insert `\@glxtrhypernameprefix` before the target name.

```

4376 \let\@glxtr@org@glstdohypertarget\glstdohypertarget
4377 \renewcommand{\glstdohypertarget}[2]{%
4378   \@glxtr@org@glstdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4379 }

```

Update `\@glstarget` to use `\def` instead being assigned with `\let` so that it can pick up the new definition and allow any further redefinitions:

```

4380 \ifx\@glstarget\@glxtr@org@glstdohypertarget
4381   \def\@glstarget{\glstdohypertarget}%
4382 \fi

```

`@makeglossaries` For the benefit of `makeglossaries`

```

4383 \newcommand*{\glxtr@makeglossaries}[1]{

```

`@glxtr@gettype` Get just the type.

```

4384 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4385   \def\@glo@type{#2}%
4386 }

```

`@assign@sortkey` Assign the sort key.

```

4387 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4388   \edef\@glo@type{\@glo@type}%
4389   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4390   {%
4391     \@glo@no@assign@sortkey{#1}%
4392   }%
4393   {%
4394     \@glo@assign@sortkey{#1}%
4395   }%
4396 }%

```

Display number list for the regular version:

splaynumberlist

```
4397 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4398 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4399 \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4400 \ifdef\@gls@loclist
4401 {%
4402 \def\@gls@noidxloclist@sep{%
4403 \def\@gls@noidxloclist@sep{%
4404 \def\@gls@noidxloclist@sep{%
4405 \glsnumlistsep
4406 }%
4407 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4408 }%
4409 }%
4410 \def\@gls@noidxloclist@finalsep{}%
4411 \def\@gls@noidxloclist@prev{}%
4412 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4413 \@gls@noidxloclist@finalsep
4414 \@gls@noidxloclist@prev
4415 }%
4416 {%
4417 \glxtrundeftag
4418 \glsdoifexists{#1}%
4419 {%
4420 \GlossariesWarning{Missing location list for ‘#1’. Either
4421 a rerun is required or you haven’t referenced the entry.}%
4422 }%
4423 }%
4424 }%
4425
```

And for the number list loop:

@numberlistloop

```
4426 \newcommand*{\@glxtr@noidx@numberlistloop}[3]{%
4427 \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4428 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4429 \let\@gls@org@glsseeformat\glsseeformat
4430 \let\glsnoidxdisplayloc#2\relax
4431 \let\glsseeformat#3\relax
4432 \ifdef\@gls@loclist
4433 {%
4434 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4435 }%
```

```

4436 {%
4437   \glxtrundeftag
4438   \glsdoifexists{#1}%
4439   {%
4440     \GlossariesWarning{Missing location list for ‘##1’. Either
4441       a rerun is required or you haven’t referenced the entry.}%
4442   }%
4443 }%
4444 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4445 \let\glsseeformat\@gls@org@glsseeformat
4446 }%

```

Same for entry number list.

entrynumberlist

```

4447 \newcommand*\@glsxtr@noidx@entrynumberlist}[1]{%
4448   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4449   \ifdef\@gls@loclist
4450     {%
4451       \glsnoidxloclist{\@gls@loclist}%
4452     }%
4453     {%
4454       \glxtrundeftag
4455       \glsdoifexists{#1}%
4456       {%
4457         \GlossariesWarning{Missing location list for ‘#1’. Either
4458           a rerun is required or you haven’t referenced the entry.}%
4459       }%
4460     }%
4461 }%

```

entrynumberlist

```

4462 \newcommand*\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4463 \renewcommand*\@gls@noidx@getgrouptitle}[2]{%
4464   \protected@edef\@glsxtr@titlelabel{#1}%
4465   \ifdefvoid\@glsxtr@titlelabel
4466     {}%
4467     {%
4468       \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4469     }%
4470   \ifdefvoid{\@glsxtr@titlelabel}%
4471     {%
4472       \DTLifint{#1}%
4473     }%
4474     \ifnum#1<256\relax
4475       \edef#2{\char#1\relax}%

```

```

4476     \else
4477         \edef#2{#1}%
4478     \fi
4479 }%
4480 {%
4481     \ifcsundef{#1groupname}%
4482     {\def#2{#1}}%
4483     {\letcs#2{#1groupname}}%
4484 }%
4485 }%
4486 {%
4487     \let#2\@glsxtr@titlelabel
4488 }%
4489 }

```

`g@getgrouptitle` Save original definition of `\@gls@getgrouptitle`
4490 `\let\glsxtr@org@getgrouptitle\@gls@getgrouptitle`

`trgetgrouptitle` Provide a user-level command to fetch the group title. The first argument is the group label.
The second argument is a control sequence in which to store the title.

```

4491 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4492   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4493   \@onelevel@sanitize\@glsxtr@titlelabel
4494   \ifcsdef{\@glsxtr@titlelabel}
4495   {\letcs{#2}{\@glsxtr@titlelabel}}%
4496   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4497 }
4498 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

`trsetgrouptitle` Sets the title for the given group label.

```

4499 \newcommand{\glsxtrsetgrouptitle}[2]{%
4500   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4501   \@onelevel@sanitize\@glsxtr@titlelabel
4502   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4503 }

```

`alsetgrouptitle` As above put only locally defines the title.

```

4504 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4505   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4506   \@onelevel@sanitize\@glsxtr@titlelabel
4507   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4508 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4509 \renewcommand*{\glsnavigation}{%
4510   \def\@gls@between{}%
4511   \ifcsundef{@gls@hypergroup@list@\@glo@type}%
4512   {%

```

```

4513   \def\@gls@list{%
4514 }%
4515 {%
4516   \expandafter\let\expandafter\@gls@list
4517     \csname @gls@hypergroup@list@\@glo@type\endcsname
4518 }%
4519 \@for\@gls@tmp:=\@gls@list\do{%
4520   \@gls@between
4521   \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4522   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4523   \let\@gls@between\glshypernavsep
4524 }%
4525 }

```

`\noidx@glossary`

```

4526 \renewcommand*{\@print@noidx@glossary}{%
4527   \ifcsdef{\@gls@ref@\@glo@type}%
4528   {%
4529     \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4530     {%
4531       \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4532     }%
4533   }%
4534   \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{%
4535 }%
4536   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4537   \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4538   \def\@gls@currentlettergroup{%
4539   \begin{theglossary}%
4540   \glossaryheader
4541   \glsresetentrylist
4542   \forlistcsloop{\@gls@noidx@do}{\@gls@ref@\@glo@type}%
4543   \end{theglossary}%
4544   \glossarypostamble
4545 }%
4546 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4547   \glsxtrifemptyglossary{\@glo@type}%
4548   }%
4549   {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4550   \@gls@noref@warn{\@glo@type}%
4551 }%
4552 }

```

`\noidxdisplayloc` Patch to check for range formations.

```

4553 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4554   \setentrycounter[#1]{#2}%
4555   \@glsxtr@display@loc#3\empty\endglsxtr@display@loc{#4}%
4556 }

```

`xtr@display@loc` Patch to check for range formations.

```

4557 \def\@glsxtr@display@loc#1#2\endglsxtr@display@loc#3{%
4558   \ifx#1(\relax
4559     \glsxtrdisplaystartloc{#2}{#3}%
4560   \else
4561     \ifx#1)\relax
4562       \glsxtrdisplayendloc{#2}{#3}%
4563     \else
4564       \glsxtrdisplaysingleloc{#1#2}{#3}%
4565     \fi
4566   \fi
4567 }

```

`isplayingleloc` Single location.

```

4568 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4569   \csuse{#1}{#2}%
4570 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

`displaystartloc` Start of a location range.

```

4571 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4572   \edef\glsxtrlocrangefmt{#1}%
4573   \ifx\glsxtrlocrangefmt\empty
4574     \def\glsxtrlocrangefmt{glsnumberformat}%
4575   \fi
4576   \expandafter\glsxtrdisplaysingleloc
4577   \expandafter{\glsxtrlocrangefmt}{#2}%
4578 }

```

`trdisplayendloc` End of a location range.

```

4579 \newcommand*{\glsxtrdisplayendloc}[2]{%
4580   \edef\@glsxtr@tmp{#1}%
4581   \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}}%
4582   \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4583   \else
4584     \GlossariesExtraWarning{Mismatched end location range
4585       (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
4586   \fi
4587   \expandafter\glsxtrdisplayendloohook\expandafter{\@glsxtr@tmp}{#2}%
4588   \expandafter\glsxtrdisplaysingleloc
4589   \expandafter{\glsxtrlocrangefmt}{#2}%
4590   \def\glsxtrlocrangefmt{}%
4591 }

```


splayendlochook Allow the user to hook into the end of range command.

```
4592 \newcommand*\glxtrdisplayendlochook}[2] {}
```

sxtrlocrangefmt Current range format. Empty if not in a range.

```
4593 \newcommand*\glxtrlocrangefmt {}
```

setentrycounter Adjust \setentrycounter to save the original prefix.

```
4594 \renewcommand*\setentrycounter}[2] [] {}%
4595 \def\glxtrcounterprefix{#1}%
4596 \ifx\glxtrcounterprefix\@empty
4597 \def\@glo@counterprefix{.}%
4598 \else
4599 \def\@glo@counterprefix{.#1.}%
4600 \fi
4601 \def\glsetentrycounter{#2}%
4602 }
```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```
4603 \def\@gls@removespaces#1 #2\@nil{%
4604 \toks@=\expandafter{\the\toks@#1}%
4605 \ifx\#2\%
```

```
4606 \edef\@glo@tmp{\the\toks@}%
4607 \ifx\@glo@tmp\empty
4608 \else
```

Expand location (just in case \toks@ is needed for something else).

```
4609 \expandafter\glxtrlocationhyperlink\expandafter
4610 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4611 \fi
4612 \else
4613 \@gls@ReturnAfterFi{%
4614 \@gls@removespaces#2\@nil
4615 }%
4616 \fi
4617 }
```

ocationhyperlink `\glxtrlocationhyperlink{<counter>}{<prefix>}{<location>}`

```
4618 \newcommand*\glxtrlocationhyperlink}[3] {%
4619 \ifdefvoid\glxtrsupplocationurl
4620 {%
4621 \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4622 }%
4623 {%
4624 \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
```

```
4625 }%
4626 }
```

supphypernumber

```
4627 \newcommand*{\glxtrsupphypernumber}[1]{%
4628 {%
4629   \glshasattribute{\glscurrententrylabel}{externallocation}%
4630   {%
4631     \def\glxtrsupplocationurl{%
4632       \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4633   }%
4634   {%
4635     \def\glxtrsupplocationurl{}%
4636   }%
4637   \glshypernumber{#1}%
4638 }%
4639 }
```

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

@print@glossary

```
4640 \renewcommand{\@print@glossary}{%
4641   \makeatletter
4642   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4643   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4644   {}%
4645   {\glxtrNoGlossaryWarning{\@glo@type}}%
4646   \ifglxindy
4647     \ifcsundef{@xdy@\@glo@type @language}%
4648     {%
4649       \edef\@do@auxoutstuff{%
4650         \noexpand\AtEndDocument{%
4651           \noexpand\immediate\noexpand\write\@auxout{%
4652             \string\providecommand\string\@xdylanguage[2]{}%
4653           \noexpand\immediate\noexpand\write\@auxout{%
4654             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4655         }%
4656       }%
4657     }%
4658   {%
4659     \edef\@do@auxoutstuff{%
4660       \noexpand\AtEndDocument{%
4661         \noexpand\immediate\noexpand\write\@auxout{%
4662           \string\providecommand\string\@xdylanguage[2]{}%
4663         \noexpand\immediate\noexpand\write\@auxout{%
4664           \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4665             @language\endcsname}}%
4666       }%
4667     }%
4668   }
```

```

4667     }%
4668 }%
4669 \@do@auxoutstuff
4670 \edef\@do@auxoutstuff{%
4671     \noexpand\AtEndDocument{%
4672         \noexpand\immediate\noexpand\write\@auxout{%
4673             \string\providecommand\string\@gls@codepage[2]{}}%
4674         \noexpand\immediate\noexpand\write\@auxout{%
4675             \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4676     }%
4677 }%
4678 \@do@auxoutstuff
4679 \fi
4680 \renewcommand*{\@warn@nomakeglossaries}{%
4681     \GlossariesWarningNoLine{\string\makeglossaries\space
4682     hasn't been used,^^Jthe glossaries will not be updated}%
4683 }%
4684 }

```

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

`oGlsWarningHead` Header message.

```

4685 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4686 This document is incomplete. The external file associated with
4687 the glossary '#1' (which should be called \texttt{#2})
4688 hasn't been created.%
4689 }

```

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

```

4690 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4691 This has probably happened because there are no entries defined
4692 in this glossary.%
4693 }

```

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

```

4694 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4695 If you don't want this glossary,
4696 add \texttt{nomain} to your package option list when you load
4697 \texttt{glossaries-extra.sty}. For example:%
4698 }

```

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

```

4699 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4700 Did you forget to use \texttt{type=#1} when you defined your
4701 entries? If you tried to load entries into this glossary with
4702 \texttt{\string\loadglsentries} did you remember to use
4703 \texttt{[#1]} as the optional argument? If you did, check that
4704 the definitions in the file you loaded all had the type set
4705 to \texttt{\string\glsdefaulttype}.%
4706 }

```

arningCheckFile Advisory message to check the file contents.

```
4707 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4708   Check the contents of the file \texttt{#1}. If
4709   it's empty, that means you haven't indexed any of your entries in this
4710   glossary (using commands like \texttt{\string\gls} or
4711   \texttt{\string\glsadd}) so this list can't be generated.
4712   If the file isn't empty, the document build process hasn't been
4713   completed.%
4714 }
```

WarningAutoMake Message when automake option has been used.

```
4715 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4716   You may need to rerun \LaTeX. If you already have, it may be that
4717   \TeX's shell escape doesn't allow you to run
4718   \ifglxindy xindy\else makeindex\fi. Check the
4719   transcript file \texttt{\jobname.log}. If the shell escape is
4720   disabled, try one of the following:
4721
4722   \begin{itemize}
4723     \item Run the external (Lua) application:
4724
4725       \texttt{makeglossaries-lite \string"\jobname\string"}
4726
4727     \item Run the external (Perl) application:
4728
4729       \texttt{makeglossaries \string"\jobname\string"}
4730   \end{itemize}
4731
4732   Then rerun \LaTeX\ on this document.
4733   \GlossariesExtraWarning{Rerun required to build the
4734   glossary '#1' or check TeX's shell escape allows
4735   you to run \ifglxindy xindy\else makeindex\fi}%
4736 }
```

WarningMismatch Mismatching \makenoidxglossaries.

```
4737 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4738   You need to either replace \texttt{\string\makenoidxglossaries}
4739   with \texttt{\string\makeglossaries} or replace
4740   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4741   \texttt{\string\printnoidxglossary}
4742   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4743   this document.%
4744 }
```

arningBuildInfo Build advice.

```
4745 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4746   Try one of the following:
4747   \begin{itemize}
```

```

4748 \item Add \texttt{automake} to your package option list when you load
4749 \texttt{glossaries-extra.sty}. For example:
4750
4751 \texttt{\string\usepackage[automake]%
4752 \glsopenbrace glossaries-extra\glsclosebrace}
4753
4754 \item Run the external (Lua) application:
4755
4756 \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4757
4758 \item Run the external (Perl) application:
4759
4760 \texttt{makeglossaries \string"\jobname\string"}
4761 \end{itemize}
4762
4763 Then rerun \LaTeX\ on this document.%.
4764 }

```

trRecordWarning Paragraph for record=only.

```

4765 \newcommand{\GlsXtrRecordWarning}[1]{%
4766 \texttt{\string\printglossary} doesn't work
4767 with the \texttt{record=only} package option
4768 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4769 instead (or change the package option).%
4770 }

```

oGlsWarningTail Final paragraph.

```

4771 \newcommand{\GlsXtrNoGlsWarningTail}{%
4772 This message will be removed once the problem has been fixed.%
4773 }

```

GlsWarningNoOut No out file created. Build advice.

```

4774 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4775 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4776 \texttt{\string\makeglossaries} or you have used
4777 \texttt{\string\nofiles}. If this is just a draft version of the
4778 document, you can suppress this message using the
4779 \texttt{nomissingglstext} package option.%.
4780 }

```

glossarywarning

```

4781 \newcommand*{@glsxtr@defaultnoglossarywarning}[1]{%
4782 \glossarysection[\glossarytoctitle]{\glossarytitle}
4783 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glo@type @in\endcsname}
4784 \par
4785 \glsxtrifemptyglossary{#1}%
4786 {%
4787 \GlsXtrNoGlsWarningEmptyStart\space
4788 \ifthenelse{equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par

```

```

4789     \medskip
4790     \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4791         \glsopenbrace glossaries-extra\glsclosebrace}
4792     \medskip
4793     }%
4794     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4795 }%
4796 {%
4797     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4798     {%
4799         \GlsXtrNoGlsWarningCheckFile
4800         {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4801
4802         \ifglsautomake
4803
4804         \GlsXtrNoGlsWarningAutoMake{#1}
4805
4806         \else
4807
4808         \ifthenelse{\equal{#1}{main}}%
4809         {%
4810             \GlsXtrNoGlsWarningEmptyMain\par
4811             \medskip
4812             \noindent\texttt{\string\usepackage[nomain]%
4813                 \glsopenbrace glossaries-extra\glsclosebrace}
4814             \medskip
4815         }%
4816         {}%
4817
4818         \ifdefequal\makeglossaries\@no@makeglossaries
4819         {%
4820             \GlsXtrNoGlsWarningMisMatch
4821         }%
4822         {%
4823             \GlsXtrNoGlsWarningBuildInfo
4824         }%
4825     \fi
4826 }%
4827 {%
4828     \GlsXtrNoGlsWarningNoOut
4829     {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4830 }%
4831 }%
4832 \par
4833 \GlsXtrNoGlsWarningTail
4834 }

```

`glossarywarning` Warn about using `\printglossary` with record

```
4835 \newcommand*{\@glsxtr@record@noglossarywarning}[1]{%
```

```

4836 \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4837 with record=only package option\MessageBreak(use
4838 \string\printunsrtglossary[type=#1])\MessageBreak
4839 instead (or change the package option)}%
4840 \glossarysection[\glossarytoctitle]{\glossarytitle}
4841 \GlsXtrRecordWarning{#1}
4842 \GlsXtrNoGlsWarningTail
4843 }

```

Provide some commands to accompany the record option for use with [bib2gls](#).

ResourceOptions Default resource options.

```
4844 \newcommand*\GlsXtrDefaultResourceOptions{}
```

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.

```
4845 \newcommand*\glsxtrresourcefile[2] []{%
```

The record option can't be set after this command.

```

4846 \disable@keys{glossaries-extra.sty}{record}%
4847 \glsxtr@writefields
4848 \ifdefempty\GlsXtrDefaultResourceOptions
4849 {%
4850 \protected@write\@auxout{\glsxtrresourceinit}%
4851 {\string\glsxtr@resource{#1}{#2}}%
4852 }%
4853 {%
4854 \protected@write\@auxout{\glsxtrresourceinit}%
4855 {\string\glsxtr@resource{\GlsXtrDefaultResourceOptions,#1}{#2}}%
4856 }%
4857 \let\@glsxtr@org@see@noindex\@gl@see@noindex
4858 \let\@gl@see@noindex\relax
4859 \IfFileExists{#2.glstex}%
4860 {%

```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```

4861 \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4862 \makeatletter
4863 \@input{#2.glstex}%
4864 \@bibgls@restreat

```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```

4865 \@glsxtr@check@bibgls@nameref
4866 }%
4867 {%
4868 \GlossariesExtraWarning{No file '#2.glstex'}%
4869 }%
4870 \let\@gl@see@noindex\@glsxtr@org@see@noindex

```

```
4871 }
4872 \@onlypreamble\glxtrresourcefile
```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```
4873 \newcommand{\@glxtr@check@bibgls@nameref}{%
4874 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
4875 \ifdef\bibglshrefchar
4876 {}%
4877 {%
4878 \GlossariesExtraWarning{record=nameref requires at least
4879 version 1.8 of bib2gls}%
4880 }%
4881 \fi
4882 \let\@glxtr@check@bibgls@nameref\relax
4883 }
```

xtrresourceinit Code used during the protected write operation.

```
4884 \newcommand*{\glxtrresourceinit}{}
```

trresourcecount

```
4885 \newcount\glxtrresourcecount
```

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```
4886 \newcommand*{\GlsXtrLoadResources}[1][1]{%
4887 \ifnum\glxtrresourcecount=0\relax
4888 \glxtrresourcefile[#1]{\jobname}%
4889 \else
4890 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4891 \fi
4892 \advance\glxtrresourcecount by 1\relax
4893 }
```

glxtr@resource

```
4894 \newcommand*{\glxtr@resource}[2]{}
```

\glxtr@fields

```
4895 \newcommand*{\glxtr@fields}[1]{}
```

xtr@texencoding

```
4896 \newcommand*{\glxtr@texencoding}[1]{}
```

\glxtr@langtag

```
4897 \newcommand*{\glxtr@langtag}[1]{}
```

@pluralsuffixes

```
4898 \newcommand*{\glxtr@pluralsuffixes}[4]{}
```


tr@shortcutsval

```
4899 \newcommand*\glstr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
4900 \newcommand*\glstr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
4901 \newcommand*\glstr@writefields{%
```

```
4902 \protected@write\@auxout{%
```

```
4903   {\string\providecommand*\string\glstr@fields}[1]{}}%
```

```
4904 \protected@write\@auxout{%
```

```
4905   {\string\providecommand*\string\glstr@resource}[2]{}}%
```

```
4906 \protected@write\@auxout{%
```

```
4907   {\string\providecommand*\string\glstr@pluralsuffixes}[4]{}}%
```

```
4908 \protected@write\@auxout{%
```

```
4909   {\string\providecommand*\string\glstr@shortcutsval}[1]{}}%
```

```
4910 \protected@write\@auxout{%
```

```
4911   {\string\providecommand*\string\glstr@linkprefix}[1]{}}%
```

```
4912 \protected@write\@auxout{\string\glstr@fields{\glstr@keymap}}%
```

```
4913 \protected@write\@auxout{%
```

```
4914   {\string\providecommand*\string\glstr@record}[5]{}}%
```

```
4915 \ifx\glstr@record@setting\glstr@record@setting@nameref
```

```
4916   \protected@write\@auxout{%
```

```
4917     {\string\providecommand*\string\glstr@record@nameref}[8]{}}%
```

```
4918 \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 `\glstrresourcefile`.

```
4919 \ifdef\CurrentTrackedLanguageTag
```

```
4920   {%
```

```
4921     \protected@write\@auxout{%
```

```
4922       \string\glstr@langtag{\CurrentTrackedLanguageTag}}%
```

```
4923   }%
```

```
4924   {%
```

```
4925     \protected@write\@auxout{\string\glstr@pluralsuffixes
```

```
4926       {\glsppluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}}%
```

```
4927       {\glstrabbrvpluralsuffix}}%
```

```
4928 \ifdef\inputencodingname
```

```
4929   {%
```

```
4930     \protected@write\@auxout{\string\glstr@texencoding{\inputencodingname}}%
```

```
4931   }%
```

```
4932   {%
```

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.)

```

4933 \ifpackageloaded{fontspec}%
4934 {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
4935 {}%
4936 }%
4937 \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.

```

4938 \AtBeginDocument
4939 {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
4940 \let\glsxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. This has to be done before the aux file is opened (so package options automake=immediate and automake=true are identical if just bib2gls is used). The double-quotes around \jobname have been removed (v1.19) since \jobname will include double-quotes if the file name has spaces.

```

4941 \ifglsautomake
4942 \IfFileExists{\jobname.aux}%
4943 {\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.

```

4944 \ifx\@gls@doautomake\@gls@doautomake@err
4945 \let\@gls@doautomake\relax
4946 \fi
4947 \fi
4948 }

```

do@automake@err

```

4949 \newcommand*{\@gls@doautomake@err}{%
4950 \PackageError{glossaries}{You must use
4951 \string\makeglossaries\space with automake=true}
4952 {%
4953 Either remove the automake=true setting or
4954 add \string\makeglossaries\space to your document preamble.%
4955 }%
4956 }

```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```

4957 \newcommand*{\glsxtr@record}[5]{}

```

@record@nameref Used with record=nameref to include current label information.

```

4958 \newcommand*{\glsxtr@record@nameref}[8]{}

```

r@counterrecord Aux file command.

```

4959 \newcommand*{\glsxtr@counterrecord}[3]{%
4960 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4961 }

```

counterrecordhook Hook used by \@glsxtr@dorecord.

```
4962 \newcommand*{\@glsxtr@counterrecordhook}{}
```

XtrRecordCounter Activate recording for a particular counter (identified in the argument).

```
4963 \newcommand*{\GlsXtrRecordCounter}[1]{%
```

```
4964   \@glsxtr@recordcounter{#1}}%
```

```
4965 }
```

```
4966 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4967 \newcommand*{\@glsxtr@docounterrecord}[1]{%
```

```
4968   \protected@write\@auxout{}{\string\glsxtr@counterrecord
```

```
4969     {\@gls@label}{#1}{\csuse{the#1}}}%
```

```
4970 }
```

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.

```
4971 \newcommand*{\glsxtrglossentry}[1]{%
```

```
4972   \glsxtrtitleorpdforheading
```

```
4973   {\@glsxtrglossentry{#1}}%
```

```
4974   {\glsentryname{#1}}%
```

```
4975   {\glsxtrheadname{#1}}%
```

```
4976 }
```

lsxtrglossentry Another test is needed in case \@glsxtrglossentry has been written to the table of contents.

```
4977 \newrobustcmd*{\@glsxtrglossentry}[1]{%
```

```
4978   \glsxtrtitleorpdforheading
```

```
4979   {%
```

```
4980     \glsdoifexists{#1}}%
```

```
4981     {%
```

```
4982       \begingroup
```

```
4983         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
```

```
4984         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
```

```
4985         \ifglshasparent{#1}%
```

```
4986         {\GlsXtrStandaloneSubEntryItem{#1}}%
```

```
4987         {\glsentryitem{#1}}%
```

```
4988         \GlsXtrStandaloneEntryName{#1}}%
```

```
4989       \endgroup
```

```
4990     }%
```

```
4991   }%
```

```
4992   {\glsentryname{#1}}%
```

```
4993   {\glsxtrheadname{#1}}%
```

```
4994 }
```

standaloneEntryName

```
4995 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
4996   \glstarget{#1}{\glossentryname{#1}}%
4997 }
```

standaloneGlossaryType To make it easier to adjust the definition of `\currentglossary` within `\glsxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```
4998 \newcommand{\GlsXtrStandaloneGlossaryType}{\glstentrytype{\glscurrententrylabel}}
```

standaloneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```
4999 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
5000   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
5001 }
```

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.

```
5002 \newcommand*{\glsxtrglossentryother}[3]{%
5003   \ifstrempy{#1}%
5004   {%
5005     \ifcsdef{glsxtrhead#3}%
5006     {%
5007       \glsxtrtitleorpdforheading
5008       {\@glsxtrglossentryother{#2}{#3}{#1}}%
5009       {\@gls@entry@field{#2}{#3}}%
5010       {\csuse{glsxtrhead#3}{#2}}%
5011     }%
5012     {%
5013       \glsxtrtitleorpdforheading
5014       {\@glsxtrglossentryother{#2}{#3}{#1}}%
5015       {\@gls@entry@field{#2}{#3}}%
5016       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
5017     }%
5018   }%
5019   {%
5020     \glsxtrtitleorpdforheading
5021     {\@glsxtrglossentryother{#2}{#3}{#1}}%
5022     {\@gls@entry@field{#2}{#3}}%
5023     {#1}%
5024   }%
5025 }
```

glossentryother As `\@glsxtrglossentry` but uses a different field.

```
5026 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
5027   \glsxtrtitleorpdforheading
5028   {%
```

```

5029 \glsdoifexists{#1}%
5030 {%
5031   \begingroup
5032     \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
5033     \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5034     \ifglshasparent{#1}%
5035       {\GlsXtrStandaloneSubEntryItem{#1}}%
5036       {\glsentryitem{#1}}%
5037     \GlsXtrStandaloneEntryOther{#1}%
5038   \endgroup
5039 }%
5040 }%
5041 {\@gls@entry@field{#1}{#2}}%
5042 {#3}%
5043 }

```

alonedEntryOther

```

5044 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
5045   \glstarget{#1}{\glossentrynameother{#1}{#2}}%
5046 }

```

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

```

5047 \newcommand*{\printunsrtglossary}{%
5048   \@ifstar\s@printunsrtglossary\@printunsrtglossary
5049 }

```

ntunsrtglossary Unstarred version.

```

5050 \newcommand*{\@printunsrtglossary}[1] []{%
5051   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
5052 }

```

ntunsrtglossary Starred version.

```

5053 \newcommand*{\s@printunsrtglossary}[2] []{%
5054   \begingroup
5055     #2%
5056     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
5057   \endgroup
5058 }

```

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

```

5059 \newcommand*{\printunsrtglossaries}{%
5060   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
5061 }

```

@unsrt@glossary

```

5062 \newcommand*{\@print@unsrt@glossary}{%

```

```

5063 \glossarysection[\glossarytoctitle]{\glossarytitle}%
5064 \glossarypreamble
    check for empty list
5065 \glstrifemptyglossary{\@glo@type}%
5066 {%
5067     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
5068 }%
5069 {%

5070 \key@ifundefined{glossentry}{group}%
5071 {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
5072 {\let\@gls@getgrouptitle\@glstr@unsrt@getgrouptitle}%
5073 \def\@gls@currentlettergroup{}}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

5074 \def\@glstr@doglossary{%
5075     \begin{theglossary}%
5076     \glossaryheader
5077     \glsresetentrylist
5078 }%
5079 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
5080 : \expandafter=\csname glist@\@glo@type\endcsname\do{%
5081     \ifdefempty{\glscurrententrylabel}
5082     }%
5083     {%

```

Provide a hook (for example to measure width).

```

5084     \let\glstr@process\@firstofone
5085     \let\printunsrtglossaryskipentry
5086         \@glstr@printunsrtglossaryskipentry
5087     \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5088     \glstr@process
5089     {%
5090     \ifglshasparent{\glscurrententrylabel}{}%
5091     {%
5092         \@glstr@checkgroup\glscurrententrylabel
5093         \expandafter\appto\expandafter\@glstr@doglossary\expandafter
5094         {\@glstr@groupheading}%
5095     }%
5096     \eappto\@glstr@doglossary{%
5097         \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5098     }%
5099     }%
5100 }%
5101 \appto\@glstr@doglossary{\end{theglossary}}%
5102 \printunsrtglossarypredoglossary
5103 \@glstr@doglossary
5104 }%

```

```
5105 \glossarypostamble
5106 }
```

entryprocesshook

```
5107 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}
```

glossaryskipentry

```
5108 \newcommand*{\printunsrtglossaryskipentry}{%
5109 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
5110 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5111 }
```

entryprocesshook

```
5112 \newcommand*{\@glstr@printunsrtglossaryskipentry}{%
5113 \let\glstr@process\@gobble
5114 }
```

entrypredoglossary

```
5115 \newcommand*{\printunsrtglossarypredoglossary}{}
```

glossary@handler

```
5116 \newcommand{\@printunsrtglossary@handler}[1]{%
5117 \xdef\glscurrententrylabel{#1}%
5118 \printunsrtglossaryhandler\glscurrententrylabel
5119 }
```

glossaryhandler

```
5120 \newcommand{\printunsrtglossaryhandler}[1]{%
5121 \glstrunsrtdo{#1}%
5122 }
```

glstriflabelinlist

```
\glstriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@glstr@ifinlist` which ensures the label and list are fully expanded.

```
5123 \newrobustcmd*{\glstriflabelinlist}[4]{%
5124 \protected@edef\@glstr@doiflabelinlist{\noexpand\@glstr@ifinlist{#1}{#2}}%
5125 \@glstr@doiflabelinlist{#3}{#4}%
5126 }
```

printunsrtglossaryunit

```
5127 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5128 \s@printunsrtglossary[type=\glstr@defaulttype,#1]{%
5129 \printunsrtglossaryunitsetup{#2}%
5130 }%
5131 }
```

glossaryunitsetup

```
5132 \newcommand*\printunsrtglossaryunitsetup}[1]{%
5133   \renewcommand{\printunsrtglossaryhandler}[1]{%
5134     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5135     {\glxtrunsrtdo{##1}}%
5136     {}}%
5137   }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glolinkprefix`. The `\@gobble` part discards `\glolinkprefix`.

```
5138   \ifcsundef{theH#1}%
5139   {%
5140     \renewcommand*\@glxtrhypernameprefix{record.#1.\csuse{the#1}.\@gobble}%
5141   }%
5142   {%
5143     \renewcommand*\@glxtrhypernameprefix{record.#1.\csuse{theH#1}.\@gobble}%
5144   }%
5145   \renewcommand*\glossarysection[2][ ]{}%
5146   \appto\glossarypostamble{\glspare\medskip\glspare}%
5147 }
```

unsrtglossaryunit

```
5148 \newcommand{\print@noop@unsrtglossaryunit}[2][ ]{%
5149   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5150     requires the record=only or record=alsoindex package option}{}%
5151 }
```

unsrt@getgrouptitle

```
5152 \newrobustcmd*\@glxtr@unsrt@getgrouptitle[2]{%
5153   \protected@edef\@glxtr@titlelabel{glxtr@grouptitle@#1}%
5154   \@onelevel@sanitize\@glxtr@titlelabel
5155   \ifcsdef{\@glxtr@titlelabel}
5156   {\letcs{#2}{\@glxtr@titlelabel}}%
5157   {\def#2{#1}}%
5158 }
```

`\glxtrunsrtdo` Provide a user-level call to `\@glxtr@noidx@do` to make it easier to define a new handler.

```
5159 \newcommand{\glxtrunsrtdo}{\@glxtr@noidx@do}
```

`glxtrgroupfield` `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).

```
5160 \newcommand*\glxtrgroupfield{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 `\@glxtr@doglossary` is being constructed rather than in the handler.

`sxtr@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.

```

5161 \newcommand*\@glsxtr@checkgroup}[1]{%
5162   \def\@glsxtr@groupheading{}%
5163   \key@ifundefined{glossentry}{group}%
5164   {%
5165     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
5166     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5167   }%
5168   {%

5169     \protected@edef\@glo@thislettergrp{%
5170       \csuse{glo@\glsdetoklabel{#1}@glsxtrgroupfield}}%
5171     }%
5172     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5173     {}%
5174     {%
5175       \ifdefempty{\@gls@currentlettergroup}{}%
5176       {\def\@glsxtr@groupheading{\gls@groupskip}}%
5177       \eappto\@glsxtr@groupheading{%
5178         \noexpand\gls@groupheading{\expandonce\@glo@thislettergrp}%
5179       }%
5180     }%
5181     \let\@gls@currentlettergroup\@glo@thislettergrp
5182 }

```

`trLocationField` Stores the internal name of the location field.

```

5183 \newcommand*\GlsXtrLocationField{location}

```

`glsxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```

5184 \newcommand{\@glsxtr@noidx@do}[1]{%
5185   \ifglsentryexists{#1}%
5186   {%
5187     \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
5188     \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@GlsXtrLocationField}%
5189     \ifglsahasparent{#1}%
5190     {%
5191       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5192       \ifdefvoid{\@gls@location}%
5193       {%
5194         \ifdefvoid{\@gls@loclist}%
5195         {%
5196           \subglossentry{\gls@level}{#1}{}%
5197         }%
5198       }%
5199       \subglossentry{\gls@level}{#1}%

```

```

5200     {%
5201     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5202     }%
5203     }%
5204     }%
5205     {%
5206     \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5207     }%
5208     }%
5209     {%

5210     \ifdefvoid{\@gls@location}%
5211     {%
5212     \ifdefvoid{\@gls@loclist}
5213     {%
5214     \glossentry{#1}{}%
5215     }%
5216     {%
5217     \glossentry{#1}%
5218     {%
5219     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5220     }%
5221     }%
5222     }%
5223     {%
5224     \glossentry{#1}%
5225     {%
5226     \glossaryentrynumbers{\@gls@location}%
5227     }%
5228     }%
5229     }%
5230     }%
5231     {}%
5232     }

```

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
```

```
5233 \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
```

```
5234 \newcommand*{\@glsxtr@providenewgls}{%
```

```
5235 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
```

```
5236 \let\@glxtr@providenewgls\relax
5237 }
```

identifyglslike Identify the command given in the second argument for the benefit of **bib2gls**.

```
5238 \newcommand{\glxtridentifyglslike}[2]{%
5239 \ifdefequal\@glxtr@record@setting\@glxtr@record@setting@off
5240 }%
5241 {%
5242 \@glxtr@providenewgls
5243 \protected@write\@auxout{}\string\@glxtr@newglslike{#1}{\string#2}}%
5244 }%
5245 }
```

`\@glxtrnewgls` `\glxtrnewgls[<options>]{<prefix>}{<cs>}{<inner cs name>}`

```
5246 \newcommand*\@glxtrnewgls[4]{%
5247 \ifdef{#3}%
5248 {%
5249 \PackageError{glossaries-extra}{Command \string#3\space already
5250 defined}{}%
5251 }%
5252 {%
```

Write information to the aux file for bib2gls.

```
5253 \glxtridentifyglslike{#2}{#3}%
5254 \ifcsdef{@#4like@#2}%
5255 {%
5256 \advance\@glxtrnewgls@inner by \@one
5257 \def\@glxtrnewgls@innercsname{@#4like\number\@glxtrnewgls@inner @#2}%
5258 }%
5259 {\def\@glxtrnewgls@innercsname{@#4like@#2}}%
5260 \expandafter\newrobustcmd\expandafter*\expandafter
5261 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glxtrnewgls@innercsname\endcsname}%
5262 \ifstrempy{#1}%
5263 {%
5264 \expandafter\newcommand\expandafter*\csname\@glxtrnewgls@innercsname\endcsname [2] [] {%
5265 \new@ifnextchar [%
5266 {\csname @#4@\endcsname{##1}{#2##2}}%
5267 {\csname @#4@\endcsname{##1}{#2##2} [] }%
5268 }%
5269 }%
5270 {%
5271 \expandafter\newcommand\expandafter*\csname\@glxtrnewgls@innercsname\endcsname [2] [] {%
5272 \new@ifnextchar [%
5273 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5274 {\csname @#4@\endcsname{#1,##1}{#2##2} [] }%
5275 }
```

```

5275     }%
5276     }%
5277     }%
5278 }

```

```

\glsxtrnewgls \glsxtrnewgls[<options>]{<prefix>}{<cs>}

```

The first argument prepends to the options and the second argument is the prefix.

```

5279 \newrobustcmd*{\glsxtrnewgls}[3] []{%
5280   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
5281 }

```

`\glsxtrnewglslike` 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.

```

5282 \newrobustcmd*{\glsxtrnewglslike}[6] []{%
5283   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
5284   \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
5285   \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
5286   \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
5287 }

```

`\glsxtrnewGLSlike` 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.

```

5288 \newrobustcmd*{\glsxtrnewGLSlike}[4] []{%
5289   \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
5290   \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
5291 }

```

`\glsxtrnewrgls` As `\glsxtrnewgls` but for `\rgls`.

```

5292 \newrobustcmd*{\glsxtrnewrgls}[3] []{%
5293   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
5294 }

```

`\glsxtrnewrglslike` As `\glsxtrnewglslike` but for `\rgls` etc.

```

5295 \newrobustcmd*{\glsxtrnewrglslike}[6] []{%
5296   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
5297   \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
5298   \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
5299   \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
5300 }

```

`\glsxtrnewrGLSlike` As `\glsxtrnewGLSlike` but for `\rGLS` etc.

```

5301 \newrobustcmd*{\glsxtrnewrGLSlike}[4] []{%
5302   \@glsxtrnewgls{#1}{#2}{#3}{rGLS}%
5303   \@glsxtrnewgls{#1}{#2}{#4}{rGLSpl}%
5304 }

```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
5305 \newcommand*\GlsXtrTotalRecordCount}[1]{%
5306 \ifcsdef{glo@glstetoklabel{#1}@recordcount}%
5307 {\csname glo@glstetoklabel{#1}@recordcount\endcsname}%
5308 {0}%
5309 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
5310 \newcommand*\GlsXtrRecordCount}[2]{%
5311 \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2}%
5312 {\csname glo@glstetoklabel{#1}@recordcount.#2\endcsname}%
5313 {0}%
5314 }
```

tionRecordCount Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless `\glstetoklocation` can be set to something sensible.

```
5315 \newcommand*\GlsXtrLocationRecordCount}[3]{%
5316 \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}}%
5317 {\csname glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}\endcsname}%
5318 {0}%
5319 }
```

trdetoklocation

```
5320 \newcommand*\glstetoklocation}[1]{#1}
```

ablerecordcount

```
5321 \newcommand*\glstrenablerecordcount}{%
5322 \renewcommand*\gls}{\rgls}%
5323 \renewcommand*\Gls}{\rGls}%
5324 \renewcommand*\glspl}{\rglspl}%
5325 \renewcommand*\Glspl}{\rGlspl}%
5326 \renewcommand*\GLS}{\rGLS}%
5327 \renewcommand*\GLSpl}{\rGLSpl}%
5328 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
5329 \newcommand*\glstxrrecordtriggervalue}[1]{%
5330 \GlsXtrTotalRecordCount{#1}%
5331 }
```

dCountAttribute

```
5332 \newcommand*\GlsXtrSetRecordCountAttribute}[2]{%
5333 \@for\@glstxr@cat:=#1\do
```

```

5334 {%
5335   \ifdefempty{\@glsxtr@cat}{}%
5336   {%
5337     \glssetcategoryattribute{\@glsxtr@cat}{recordcount}{#2}%
5338   }%
5339 }%
5340 }

```

`\glsxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}`

```

5341 \newcommand*\glsxtrifrecordtrigger}[3]{%
5342   \glshasattribute{#1}{recordcount}%
5343   {%
5344     \ifnum\glsxtrrecordtriggervalue{#1}>\glsgetattribute{#1}{recordcount}\relax
5345     #3%
5346   \else
5347     #2%
5348   \fi
5349 }%
5350 {#3}%
5351 }

```

`trigger@record` Still need a record to ensure that bib2gls selects the entry.

```

5352 \newcommand*\@glsxtr@rglstrigger@record}[3]{%
5353   \edef\glslabel{\glsdetoklabel{#2}}%
5354   \let\@gls@link@label\glslabel
5355   \def\@glsxtr@thevalue{}%
5356   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
5357   \def\@glsnumberformat{glstriggerrecordformat}%
5358   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5359   \edef\@glsstype{\csname glo@\glslabel @type\endcsname}%
5360   \def\@glsxtr@thevalue{}%
5361   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
5362   \glsxtrinitwrgloss
5363   \glslinkpresetkeys
5364   \setkeys{glslink}{#1}%
5365   \glslinkpostsetkeys
5366   \ifdefempty{\@glsxtr@thevalue}%
5367   {%
5368     \@gls@saveentrycounter
5369   }%
5370   {%
5371     \let\theglentrycounter\@glsxtr@thevalue
5372     \def\theHglentrycounter{\@glsxtr@theHvalue}%
5373   }%
5374   \ifglsxtrinitwrglossbefore

```

```

5375 \do@wrglossary{#2}%
5376 \fi
5377 #3%
5378 \ifglstrinitwrglossbefore
5379 \else
5380 \do@wrglossary{#2}%
5381 \fi
5382 \ifKV@glslink@local
5383 \glsllocalunset{#2}%
5384 \else
5385 \glunset{#2}%
5386 \fi
5387 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```
5388 \newcommand*{\glstriggerrecordformat}[1]{}
```

\rgls

```
5389 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}
```

\@rgls

```

5390 \newcommand*{\@rgls}[2][ ]{%
5391 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[ ]}%
5392 }

```

\@rgls@

```

5393 \def\@rgls@#1#2[#3]{%
5394 \glstrifrecordtrigger{#2}%
5395 {%
5396 \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5397 }%
5398 {%
5399 \@gls@{#1}{#2}[#3]%
5400 }%
5401 }%

```

\rglsp1

```
5402 \newrobustcmd*{\rglsp1}{\@gls@hyp@opt\@rglsp1}
```

\@rglsp1

```

5403 \newcommand*{\@rglsp1}[2][ ]{%
5404 \new@ifnextchar[{\@rglsp1@{#1}{#2}}{\@rglsp1@{#1}{#2}[ ]}%
5405 }

```

\@rglsp1@

```

5406 \def\@rglsp1@#1#2[#3]{%
5407 \glstrifrecordtrigger{#2}%

```

```

5408 {%
5409   \@glxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5410 }%
5411 {%
5412   \@glsp1@{#1}{#2}[#3]%
5413 }%
5414 }%

```

\rGls

```
5415 \newrobustcmd*{\rGls}{\@gls@hyp@opt\rGls}
```

\@rGls

```

5416 \newcommand*{\@rGls}[2][{}]{%
5417   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5418 }

```

\@rGls@

```

5419 \def\@rGls@#1#2[#3]{%
5420   \glxtrifrecordtrigger{#2}%
5421   {%
5422     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5423   }%
5424   {%
5425     \@Gls@{#1}{#2}[#3]%
5426   }%
5427 }%

```

\rGlspl

```
5428 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}
```

\@rGlspl

```

5429 \newcommand*{\@rGlspl}[2][{}]{%
5430   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5431 }

```

\@rGlspl@

```

5432 \def\@rGlspl@#1#2[#3]{%
5433   \glxtrifrecordtrigger{#2}%
5434   {%
5435     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5436   }%
5437   {%
5438     \@Glspl@{#1}{#2}[#3]%
5439   }%
5440 }%

```

\rGLS

```
5441 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}
```


\@rGLS

```
5442 \newcommand*{\@rGLS}[2] [] {%
5443   \new@ifnextchar [{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2} []}%
5444 }
```

\@rGLS@

```
5445 \def\@rGLS@#1#2[#3] {%
5446   \glsxtrifrecordtrigger{#2}%
5447   {%
5448     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
5449   }%
5450   {%
5451     \@GLS@{#1}{#2}[#3]%
5452   }%
5453 }%
```

\rGLSpl

```
5454 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}
```

\@rGLSpl

```
5455 \newcommand*{\@rGLSpl}[2] [] {%
5456   \new@ifnextchar [{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2} []}%
5457 }
```

\@rGLSpl@

```
5458 \def\@rGLSpl@#1#2[#3] {%
5459   \glsxtrifrecordtrigger{#2}%
5460   {%
5461     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5462   }%
5463   {%
5464     \@GLSpl@{#1}{#2}[#3]%
5465   }%
5466 }%
```

\rglsformat

```
5467 \newcommand*{\rglsformat}[2] {%
5468   \glsifregular{#1}
5469   {\glsentryfirst{#1}}%
5470   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5471 }
```

\rglspformat

```
5472 \newcommand*{\rglspformat}[2] {%
5473   \glsifregular{#1}
5474   {\glsentryfirstplural{#1}}%
5475   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5476 }
```

`\rGlsformat`

```
5477 \newcommand*\rGlsformat}[2]{%
5478   \glsifregular{#1}
5479   {\Glsentryfirst{#1}}%
5480   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2}%
5481 }
```

`\rGlsplformat`

```
5482 \newcommand*\rGlsplformat}[2]{%
5483   \glsifregular{#1}
5484   {\Glsentryfirstplural{#1}}%
5485   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}#2}%
5486 }
```

`\rGLSformat`

```
5487 \newcommand*\rGLSformat}[2]{%
5488   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsformat{#1}{#2}}%
5489 }
```

`\rGLSplformat`

```
5490 \newcommand*\rGLSplformat}[2]{%
5491   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsplformat{#1}{#2}}%
5492 }
```

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`.

```
5493 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
5494   \glsifattribute{\glslabel}{linkcount}{true}%
5495   {%
  Does the counter exist?
5496   \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5497   {%
  Counter doesn't exist, so define it.
5498   \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
5499 \glshasattribute{\glslabel}{linkcountmaster}%  
5500 {%
```

Need to ensure values are fully expanded.

```
5501 \begingroup  
5502 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@  
5503 {\glsgetattribute{\glslabel}{linkcountmaster}}}%  
5504 \x  
5505 }%  
5506 {}%  
5507 }%
```

Increment counter:

```
5508 \glsxtrinclinkcounter{glsxtr@linkcount@  
5509 }%  
5510 {}%  
5511 }
```

`\refinclinkcounter` May be redefined to use `\refstepcounter` if required.

```
5512 \newcommand*{\glsxtrinclinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
5513 \newcommand*{\GlsXtrLinkCounterValue}[1]{%  
5514 \ifcsundef{c@  
5515 }
```

`\TheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5516 \newcommand*{\GlsXtrTheLinkCounter}[1]{%  
5517 \ifcsundef{theglsxtr@linkcount@#1}{0}%  
5518 {\csname theglsxtr@linkcount@#1\endcsname}%  
5519 }
```

`\IfLinkCounterDef` Tests if the counter has been defined

```
5520 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%  
5521 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%  
5522 }
```

`\LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
5523 \newcommand*{\GlsXtrLinkCounterName}[1]{glsxtr@linkcount@#1}
```

`\enableLinkCounting` `\GlsXtrEnableLinkCounting[(master counter)]{(categories)}`

Enable link counting for the given categories.

```
5524 \newcommand*{\GlsXtrEnableLinkCounting}[2] [] {%  
5525 \let\glsxtr@inc@linkcount\glsxtr@do@inc@linkcount
```

```

5526 \@for\@glxtr@label:=#2\do
5527 {%
5528 \glsssetcategoryattribute{\@glxtr@label}{linkcount}{true}%
5529 \ifstrempy{#1}{}%
5530 {%
5531 \ifcsundef{c@#1}%
5532 {\@nocounterr{#1}}%
5533 {\glsssetcategoryattribute{\@glxtr@label}{linkcountmaster}{#1}}%
5534 }%
5535 }%
5536 }
5537 \@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.

```

5538 \@ifpackageloaded{glossaries-accsupp}
5539 {

```

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

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

```

5540 \newcommand*\glsaccessname[1]{%
5541 \glsnameaccessdisplay
5542 {%
5543 \glstentryname{#1}%
5544 }%
5545 {#1}%
5546 }

```

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

```

5547 \newcommand*\Glsaccessname[1]{%
5548 \glsnameaccessdisplay
5549 {%
5550 \Glstentryname{#1}%
5551 }%
5552 {#1}%
5553 }

```

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

```

5554 \newcommand*\GLSaccessname[1]{%
5555 \glsnameaccessdisplay
5556 {%
5557 \mfirstucMakeUppercase{\glstentryname{#1}}%

```

```

5558   }%
5559   {#1}%
5560 }

```

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

```

5561 \newcommand*{\glsaccesstext}[1]{%
5562   \glstextaccessdisplay
5563   {%
5564     \glsentrytext{#1}%
5565   }%
5566   {#1}%
5567 }

```

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

```

5568 \newcommand*{\Glsaccesstext}[1]{%
5569   \glstextaccessdisplay
5570   {%
5571     \Glsentrytext{#1}%
5572   }%
5573   {#1}%
5574 }

```

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

```

5575 \newcommand*{\GLSaccesstext}[1]{%
5576   \glstextaccessdisplay
5577   {%
5578     \mfirstucMakeUppercase{\glsentrytext{#1}}%
5579   }%
5580   {#1}%
5581 }

```

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

```

5582 \newcommand*{\glsaccessplural}[1]{%
5583   \glspluralaccessdisplay
5584   {%
5585     \glsentryplural{#1}%
5586   }%
5587   {#1}%
5588 }

```

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

```

5589 \newcommand*{\Glsaccessplural}[1]{%
5590   \glspluralaccessdisplay
5591   {%
5592     \Glsentryplural{#1}%
5593   }%

```

```
5594   {#1}%
5595 }
```

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

```
5596 \newcommand*{\GLSaccessplural}[1]{%
5597   \glspluralaccessdisplay
5598   {%
5599     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5600   }%
5601   {#1}%
5602 }
```

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

```
5603 \newcommand*{\glsaccessfirst}[1]{%
5604   \glsfirstaccessdisplay
5605   {%
5606     \glsentryfirst{#1}%
5607   }%
5608   {#1}%
5609 }
```

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

```
5610 \newcommand*{\Glsaccessfirst}[1]{%
5611   \glsfirstaccessdisplay
5612   {%
5613     \Glsentryfirst{#1}%
5614   }%
5615   {#1}%
5616 }
```

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

```
5617 \newcommand*{\GLSaccessfirst}[1]{%
5618   \glsfirstaccessdisplay
5619   {%
5620     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
5621   }%
5622   {#1}%
5623 }
```

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

```
5624 \newcommand*{\glsaccessfirstplural}[1]{%
5625   \glsfirstpluralaccessdisplay
5626   {%
5627     \glsentryfirstplural{#1}%
5628   }%
5629   {#1}%
5630 }
```

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

```
5631 \newcommand*{\Glsaccessfirstplural}[1]{%
5632   \glsfirstpluralaccessdisplay
5633   {%
5634     \Glsentryfirstplural{#1}%
5635   }%
5636   {#1}%
5637 }
```

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

```
5638 \newcommand*{\GLSaccessfirstplural}[1]{%
5639   \glsfirstpluralaccessdisplay
5640   {%
5641     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5642   }%
5643   {#1}%
5644 }
```

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

```
5645 \newcommand*{\glsaccesssymbol}[1]{%
5646   \glsymbolaccessdisplay
5647   {%
5648     \glsentrysymbol{#1}%
5649   }%
5650   {#1}%
5651 }
```

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

```
5652 \newcommand*{\GLSaccesssymbol}[1]{%
5653   \glsymbolaccessdisplay
5654   {%
5655     \Glsentrysymbol{#1}%
5656   }%
5657   {#1}%
5658 }
```

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

```
5659 \newcommand*{\GLSaccesssymbol}[1]{%
5660   \glsymbolaccessdisplay
5661   {%
5662     \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5663   }%
5664   {#1}%
5665 }
```

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

```

5666 \newcommand*\glsaccesssymbolplural}[1]{%
5667   \glsymbolpluralaccessdisplay
5668   {%
5669     \glsentrysymbolplural{#1}%
5670   }%
5671   {#1}%
5672 }

```

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

```

5673 \newcommand*\Glsaccesssymbolplural}[1]{%
5674   \glsymbolpluralaccessdisplay
5675   {%
5676     \Glsentrysymbolplural{#1}%
5677   }%
5678   {#1}%
5679 }

```

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

```

5680 \newcommand*\GLSaccesssymbolplural}[1]{%
5681   \glsymbolpluralaccessdisplay
5682   {%
5683     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5684   }%
5685   {#1}%
5686 }

```

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

```

5687 \newcommand*\glsaccessdesc}[1]{%
5688   \glsdescriptionaccessdisplay
5689   {%
5690     \glsentrydesc{#1}%
5691   }%
5692   {#1}%
5693 }

```

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

```

5694 \newcommand*\Glsaccessdesc}[1]{%
5695   \glsdescriptionaccessdisplay
5696   {%
5697     \Glsentrydesc{#1}%
5698   }%
5699   {#1}%
5700 }

```

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

```

5701 \newcommand*\GLSaccessdesc}[1]{%

```



```

5702   \glsdescriptionaccessdisplay
5703   {%
5704     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5705   }%
5706   {#1}%
5707   }

```

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

```

5708   \newcommand*{\glsaccessdescplural}[1]{%
5709     \glsdescriptionpluralaccessdisplay
5710     {%
5711       \glsentrydescplural{#1}%
5712     }%
5713     {#1}%
5714   }

```

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

```

5715   \newcommand*{\Glsaccessdescplural}[1]{%
5716     \glsdescriptionpluralaccessdisplay
5717     {%
5718       \Glsentrydescplural{#1}%
5719     }%
5720     {#1}%
5721   }

```

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

```

5722   \newcommand*{\GLSaccessdescplural}[1]{%
5723     \glsdescriptionpluralaccessdisplay
5724     {%
5725       \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5726     }%
5727     {#1}%
5728   }

```

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

```

5729   \newcommand*{\glsaccessshort}[1]{%
5730     \glsshortaccessdisplay
5731     {%
5732       \glsentryshort{#1}%
5733     }%
5734     {#1}%
5735   }

```

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

```

5736   \newcommand*{\Glsaccessshort}[1]{%
5737     \glsshortaccessdisplay

```

```

5738   {%
5739     \Glentryshort{#1}%
5740   }%
5741   {#1}%
5742 }

```

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

```

5743 \newcommand*\GLSaccessshort}[1]{%
5744   \glshortaccessdisplay
5745   {%
5746     \mfirstucMakeUppercase{\glentryshort{#1}}%
5747   }%
5748   {#1}%
5749 }

```

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

```

5750 \newcommand*\glaccessshortpl}[1]{%
5751   \glshortpluralaccessdisplay
5752   {%
5753     \glentryshortpl{#1}%
5754   }%
5755   {#1}%
5756 }

```

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

```

5757 \newcommand*\lSaccessshortpl}[1]{%
5758   \glshortpluralaccessdisplay
5759   {%
5760     \Glentryshortpl{#1}%
5761   }%
5762   {#1}%
5763 }

```

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

```

5764 \newcommand*\LSaccessshortpl}[1]{%
5765   \glshortpluralaccessdisplay
5766   {%
5767     \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5768   }%
5769   {#1}%
5770 }

```

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

```

5771 \newcommand*\glsaccesslong}[1]{%
5772   \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5773 }

```

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

```
5774
5775 \newcommand*\GLsaccesslong}[1]{%
5776   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5777 }
```

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

```
5778 \newcommand*\GLSaccesslong}[1]{%
5779   \glslongaccessdisplay
5780   {%
5781     \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5782   }%
5783   {#1}%
5784 }
```

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

```
5785 \newcommand*\glsaccesslongpl}[1]{%
5786   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5787 }
```

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

```
5788
5789 \newcommand*\Glsaccesslongpl}[1]{%
5790   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5791 }
```

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

```
5792 \newcommand*\GLSaccesslongpl}[1]{%
5793   \glslongpluralaccessdisplay
5794   {%
5795     \mfirstucMakeUppercase{\Glsentrylongpl{#1}}%
5796   }%
5797   {#1}%
5798 }
```

Keys for accessibility support.

```
5799 \define@key{glsxtrabbrv}{access}{%
5800   \def\@gls@nameaccess{#1}%
5801 }
5802 \define@key{glsxtrabbrv}{textaccess}{%
5803   \def\@gls@textaccess{#1}%
5804 }
5805 \define@key{glsxtrabbrv}{firstaccess}{%
5806   \def\@gls@firstaccess{#1}%
5807 }
5808 \define@key{glsxtrabbrv}{shortaccess}{%
5809   \def\@gls@shortaccess{#1}%
5810 }
```

```

5811 \define@key{glsxtrabbrv}{shortpluralaccess}{%
5812   \def\@gls@shortaccesspl{#1}%
5813 }

```

@initaccesskeys

```

5814 \newcommand*\@gls@initaccesskeys{%
5815   \def\@gls@nameaccess{}%
5816   \def\@gls@textaccess{}%
5817   \def\@gls@firstaccess{}%
5818   \def\@gls@shortaccess{}%
5819   \def\@gls@shortaccesspl{}%
5820 }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```

5821 \newcommand*\@gls@ifaccessattribute@set}[3]{%
5822   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5823   {#2}%
5824   {%
5825     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5826     {#3}%
5827     {%
5828       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5829       {#2}%
5830       {#3}%
5831     }%
5832   }%
5833 }

```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5834 \newcommand*\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```

5835   \ifdefempty\@gls@shortaccess
5836   {%
5837     \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5838     {%
5839       \glsxtr@insertdots\@gls@shortaccess{#1}%
5840       \eappto\ExtraCustomAbbreviationFields{%
5841         shortaccess={\expandonce\@gls@shortaccess},}%
5842       }%
5843     }%
5844   }%
5845 }%

```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5846 \ifdefempty\@gls@shortaccess
5847 {}%
5848 {%
5849 \ifdefempty\@gls@shortaccesspl
5850 {%
5851 \@gls@ifaccessattribute@set{aposplural}%
5852 {%
5853 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5854 \@gls@shortaccess'\abbrvpluralsuffix}%
5855 }%
5856 {%
5857 \@gls@ifaccessattribute@set{noshortplural}%
5858 {%
5859 \let\@gls@shortaccesspl\@gls@shortaccess
5860 }%
5861 {%
5862 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5863 \@gls@shortaccess\abbrvpluralsuffix}%
5864 }%
5865 }%
5866 \eappto\ExtraCustomAbbreviationFields{%
5867 shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5868 }%
5869 {}%
5870 }%
```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```
5871 \ifdefempty\@gls@nameaccess
5872 {%
5873 \glsifcategoryattribute{\gls@categorylabel}{nameshortaccess}{true}%
5874 }
```

Do nothing if the shortaccess key hasn't been set.

```
5875 \ifdefempty\@gls@shortaccess
5876 {}%
5877 {%
5878 \eappto\ExtraCustomAbbreviationFields{%
5879 access={\expandonce\@gls@shortaccess},%
5880 }%
5881 }%
5882 }%
5883 {}%
5884 }%
5885 }
```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```
5886 \ifdefempty\@gls@textaccess
5887 {%
5888 \glsifcategoryattribute{\gls@categorylabel}{textshortaccess}{true}%
5889 }
```

Do nothing if the shortaccess key hasn't been set.

```
5890     \ifdefempty\@gls@shortaccess
5891     {}%
5892     {%
5893         \eappto\ExtraCustomAbbreviationFields{%
5894             textaccess={\expandonce\@gls@shortaccess},%
5895         }%
5896     }%
5897 }%
5898 {}%
5899 }%
5900 {}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5901     \ifdefempty\@gls@firstaccess
5902     {%
5903         \glsifcategoryattribute{\gls@categorylabel}{firstshortaccess}{true}%
5904     }
```

Do nothing if the shortaccess key hasn't been set.

```
5905     \ifdefempty\@gls@shortaccess
5906     {}%
5907     {%
5908         \eappto\ExtraCustomAbbreviationFields{%
5909             firstaccess={\expandonce\@gls@shortaccess},%
5910         }%
5911     }%
5912 }%
5913 {}%
5914 }%
5915 {}%
5916 }
```

End of if accsupp part

```
5917 }
5918 {
```

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

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

```
5919     \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.

```
5920     \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

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

```
5921     \newcommand*{\GLSaccessname}[1]{%
```

```
5922     \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

`\glsaccessstext` Display the text value (no link and no check for existence).
5923 `\newcommand*{\glsaccessstext}[1]{\glsentrytext{#1}}`

`\Glsaccessstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.
5924 `\newcommand*{\Glsaccessstext}[1]{\Glsentrytext{#1}}`

`\GLSaccessstext` Display the text value (no link and no check for existence). converted to upper case.
5925 `\newcommand*{\GLSaccessstext}[1]{%`
5926 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
5927 `\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.
5928 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
5929 `\newcommand*{\GLSaccessplural}[1]{%`
5930 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
5931 `\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.
5932 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
5933 `\newcommand*{\GLSaccessfirst}[1]{%`
5934 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`glsaccessfirstplural` Display the firstplural value (no link and no check for existence).
5935 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`Glsaccessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5936 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`GLSaccessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5937 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5938 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).
5939 `\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.
5940 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
5941 `\newcommand*{\GLSaccesssymbol}[1]{%`
5942 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).
5943 `\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.
5944 `\newcommand*{\GLSaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.
5945 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5946 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).
5947 `\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.
5948 `\newcommand*{\GLSaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.
5949 `\newcommand*{\GLSaccessdesc}[1]{%`
5950 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`ccessdescplural` Display the descplural value (no link and no check for existence).
5951 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.
5952 `\newcommand*{\GLSaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.
5953 `\newcommand*{\GLSaccessdescplural}[1]{%`
5954 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).
5955 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).
5956 `\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.
5957 `\newcommand*{\GLSaccessshort}[1]{%`
5958 `\protect\mfirstucMakeUppercase{\glsentryshort{#1}}}`

`laccessshortpl` Display the short plural form (no link and no check for existence).
5959 `\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}`

`laccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).
5960 `\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.
5961 `\newcommand*{\GLSaccessshortpl}[1]{%`
5962 `\protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).
5963 `\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}`

`\Glsaccesslong` Display the long form (no link and no check for existence).
5964 `\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.
5965 `\newcommand*{\GLSaccesslong}[1]{%`
5966 `\protect\mfirstucMakeUppercase{\glsentrylong{#1}}}`

`glsaccesslongpl` Display the long plural form (no link and no check for existence).
5967 `\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}`

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).
5968 `\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}`

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.
5969 `\newcommand*{\GLSaccesslongpl}[1]{%`
5970 `\protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}`

`@initaccesskeys` This does nothing if there's no accessibility support.
5971 `\newcommand*{@gls@initaccesskeys}{}}`

`lt@short@access` This does nothing if there's no accessibility support.
5972 `\newcommand{\@gls@setup@default@short@access}[1]{%`
End of else part
5973 `}`

1.6 Categories

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

```
5974 \glsaddstoragekey{category}{general}{\glscategory}
```

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

```
5975 \newcommand{\glsifcategory}[4]{%
5976 \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
5977 }
```

Categories can have attributes.

`categoryattribute`

```
\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

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

```
5978 \newcommand*{\glssetcategoryattribute}[3]{%
5979 \csdef{@glsxtr@categoryattr@#1@#2}{#3}%
5980 }
```

`categoryattribute`

```
\glsgetcategoryattribute{<category>}{<attribute-label>}
```

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

```
5981 \newcommand*{\glsgetcategoryattribute}[2]{%
5982 \csuse{@glsxtr@categoryattr@#1@#2}%
5983 }
```

`categoryattribute`

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

Tests if the category has the given attribute set.

```
5984 \newcommand*{\glshascategoryattribute}[4]{%
5985 \ifcvoid{@glsxtr@categoryattr@#1@#2}{#4}{#3}%
5986 }
```

`\glssetattribute`

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

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

```
5987 \newcommand*{\glssetattribute}[3]{%
```

```
5988 \glssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5989 }
```

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

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

```
5990 \newcommand*\glsgetattribute}[2]{%
5991 \glssetcategoryattribute{\glscategory{#1}}{#2}%
5992 }
```

```
\glschasattribute \glschasattribute{<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.

```
5993 \newcommand*\glschasattribute}[4]{%
5994 \ifglentryexists{#1}%
5995 {\glschascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5996 {#4}%
5997 }
```

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

True if category has the attribute with the given value.

```
5998 \newcommand{\glsifcategoryattribute}[5]{%
5999 \ifcsundef{@glsxtr@categoryattr@#1@#2}%
6000 {#5}%
6001 {\ifcsstring{@glsxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
6002 }
```

```
\glsifattribute \glsifattribute{<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.

```
6003 \newcommand{\glsifattribute}[5]{%
6004 \ifglentryexists{#1}%
6005 {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
6006 {#5}%
6007 }
```

Set attributes for the default general category:

```
6008 \glssetcategoryattribute{general}{regular}{true}
```

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

```
6009 \glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to add the regular attribute.

```
6010 \newcommand*{\glssetregularcategory}[1]{%
6011 \glssetcategoryattribute{#1}{regular}{true}%
6012 }
```

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

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

```
6013 \newcommand{\glsifregularcategory}[3]{%
6014 \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
6015 }
```

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

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

```
6016 \newcommand{\glsifnotregularcategory}[3]{%
6017 \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
6018 }
```

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

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

```
6019 \newcommand{\glsifregular}[3]{%
6020 \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
6021 }
```

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

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

```
6022 \newcommand{\glsifnotregular}[3]{%
6023 \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
6024 }
```

oreachincategory

```
\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.

```
6025 \newcommand{\glsforeachincategory}[5][\@glo@types]{%  
6026   \forallglossaries[#1]{#3}%  
6027   {%  
6028     \forglentries[#3]{#4}%  
6029     {%  
6030       \glsifcategory{#4}{#2}{#5}{}%  
6031     }%  
6032   }%  
6033 }
```

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.

```
6034 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%  
6035   \forallglossaries[#1]{#4}%  
6036   {%  
6037     \forglentries[#4]{#5}%  
6038     {%  
6039       \glsifattribute{#5}{#2}{#3}{#6}{}%  
6040     }%  
6041   }%  
6042 }
```

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

```
6043 \ifdef\newterm  
6044 {%
```

`\newterm`

```
6045   \renewcommand*{\newterm}[2][ ]{%  
6046     \newglossaryentry{#2}%  
6047     {type={index},category=index,name={#2},%
```

```

6048     description={\glstrpostdescription\nopostdesc},#1}%
6049 }

```

Indexed terms are regular by default.

```

6050 \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

6051 \newcommand*{\glstrpostdescindex}{}

```

```

6052 }

```

```

6053 {}

```

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.

```

6054 \ifdef\printsymbols

```

```

6055 {%

```

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

```

6056 \newcommand*{\glsxtrnewsymbol}[3][[]]{%

```

```

6057   \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%

```

```

6058 }

```

Symbols are regular by default.

```

6059 \glsssetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

6060 \newcommand*{\glsxtrpostdescsymbol}{}

```

```

6061 }

```

```

6062 {}

```

Similar for the numbers option.

```

6063 \ifdef\printnumbers

```

```

6064 {%

```

glsxtrnewnumber

```

6065 \ifdef\printnumbers

```

```

6066 \newcommand*{\glsxtrnewnumber}[3][[]]{%

```

```

6067   \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%

```

```

6068 }

```

Numbers are regular by default.

```

6069 \glsssetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

6070 \newcommand*{\glsxtrpostdescnumber}{}

```

```
6071 }
6072 {}
```

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

```
6073 \newcommand*\glstrsetcategory}[2]{%
6074   \for\@glstr@label:=#1\do
6075   {%
6076     \glsfieldxdef{\@glstr@label}{category}{#2}%
6077   }%
6078 }
```

`\glstrcategoryforall` 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.

```
6079 \newcommand*\glstrcategoryforall}[2]{%
6080   \forallglossaries[#1]{\@glstr@type}{%
6081     \for\glsentries[\@glstr@type]{\@glstr@label}%
6082     {%
6083       \glsfieldxdef{\@glstr@label}{category}{#2}%
6084     }%
6085   }%
6086 }
```

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

Apply title casing to the contents of the given field.

```
6087 \newcommand*\glstrfieldtitlecase}[2]{%
6088   \expandafter\glstrfieldtitlecasesecs\expandafter
6089   {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
6090 }
```

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

```
6091 \newcommand*\glstrfieldtitlecasesecs}[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.

```
6092 \@ifpackageloaded{glossaries-accsupp}
6093 {
6094   \renewcommand*\glossentrydesc}[1]{%
6095     \glsdoifexistsorwarn{#1}%
6096     {%
6097       \glssetabbrvfmt{\glscategory{#1}}%
```

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

```

6098 \glshasattribute{#1}{glossdescfont}%
6099 {%
6100 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
6101 \ifcsdef{\@glxtr@attrval}%
6102 {%
6103 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6104 }%
6105 {%
6106 \GlossariesExtraWarning{Unknown control sequence name
6107 '\@glxtr@attrval' supplied in glossdescfont attribute
6108 for entry '#1'. Ignoring}%
6109 \let\@glxtr@glossdescfont\@firstofone
6110 }%
6111 }%
6112 {\let\@glxtr@glossdescfont\@firstofone}%
6113 \glusifattribute{#1}{glossdesc}{firstuc}%
6114 {%
6115 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6116 }%
6117 {%
6118 \glusifattribute{#1}{glossdesc}{title}%
6119 {%
6120 \@glxtr@do@titlecaps@warn
6121 \glsdescriptionaccessdisplay
6122 {%
6123 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6124 }%
6125 {#1}%
6126 }%
6127 {%
6128 \@glxtr@glossdescfont{\glaccessdesc{#1}}%
6129 }%
6130 }%
6131 }%
6132 }
6133 }
6134 {
6135 \renewcommand*{\glossentrydesc}[1]{%
6136 \glsdoifexistsorwarn{#1}%
6137 {%
6138 \glissetabbrfmt{\glscategory{#1}}%
6139 \glshasattribute{#1}{glossdescfont}%
6140 {%
6141 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
6142 \ifcsdef{\@glxtr@attrval}%
6143 {%
6144 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6145 }%

```



```

6146     {%
6147         \GlossariesExtraWarning{Unknown control sequence name
6148         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
6149         for entry ‘#1’. Ignoring}%
6150         \let\@glxtr@glossdescfont\@firstofone
6151     }%
6152 }%
6153 {\let\@glxtr@glossdescfont\@firstofone}%
6154 \glsifattribute{#1}{glossdesc}{firstuc}%
6155 {%
6156     \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
6157 }%
6158 {%
6159     \glsifattribute{#1}{glossdesc}{title}%
6160     {%
6161         \@glxtr@do@titlecaps@warn
6162         \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6163     }%
6164     {%
6165         \@glxtr@glossdescfont{\glentrydesc{#1}}%
6166     }%
6167 }%
6168 }%
6169 }
6170 }

```

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

```

6171 \ifpackageloaded{glossaries-accsupp}
6172 {
6173   \renewcommand*{\glossentryname}[1]{%
6174     \@glsdoifexistsorwarn{#1}%
6175     {%
6176       \glssetabbrvfmt{\glscategory{#1}}%

```

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

```

6177   \glshasattribute{#1}{glossnamefont}%
6178   {%
6179     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6180     \ifcsdef{\@glxtr@attrval}%
6181     {%
6182       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6183     }%
6184     {%
6185       \GlossariesExtraWarning{Unknown control sequence name
6186       ‘\@glxtr@attrval’ supplied in glossnamefont attribute
6187       for entry ‘#1’. Reverting to default \string\glsnamefont}%
6188       \let\@glxtr@glossnamefont\glsnamefont
6189     }%
6190 }%

```

```

6191     {\let\@glxtr@glossnamefont\glsnamefont}%
6192     \glsifattribute{#1}{glossname}{firstuc}%
6193     {%
6194         \glsnameaccessdisplay
6195         {%
6196             \@glxtr@glossnamefont{\Glsentryname{#1}}%
6197         }%
6198         {#1}%
6199     }%
6200     {%
6201         \glsifattribute{#1}{glossname}{title}%
6202         {%
6203             \@glxtr@do@titlecaps@warn
6204             \glsnameaccessdisplay
6205             {%
6206                 \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
6207             }%
6208             {#1}%
6209         }%
6210         {%
6211             \glsifattribute{#1}{glossname}{uc}%
6212             {%
6213                 \glsnameaccessdisplay
6214                 {%

```

Hide the label from the upper-casing command.

```

6215             \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6216             \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6217         }%
6218         {#1}%
6219     }%
6220     {%
6221         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6222         \glsnameaccessdisplay
6223         {%
6224             \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
6225         }%
6226         {#1}%
6227     }%
6228 }%
6229 }%

```

Do post-name hook:

```

6230     \glxtrpostnamehook{#1}%
6231     }%
6232 }
6233 }
6234 {
6235     \renewcommand*{\glossentryname}[1]{%
6236         \@glsdoifexistsorwarn{#1}%

```

```

6237  {%
6238    \glssetabbrvfmt{\glscategory{#1}}%
6239    \glsattribute{#1}{glossnamefont}%
6240  {%
6241    \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6242    \ifcsdef{\@glsxtr@attrval}%
6243    {%
6244      \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6245    }%
6246    {%
6247      \GlossariesExtraWarning{Unknown control sequence name
6248        '\@glsxtr@attrval' supplied in glossnamefont attribute
6249        for entry '#1'. Reverting to default \string\glsnamefont}%
6250      \let\@glsxtr@glossnamefont\glsnamefont
6251    }%
6252  }%
6253  {\let\@glsxtr@glossnamefont\glsnamefont}%
6254  \glsifattribute{#1}{glossname}{firstuc}%
6255  {%
6256    \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6257  }%
6258  {%
6259    \glsifattribute{#1}{glossname}{title}%
6260  {%
6261    \@glsxtr@do@titlecaps@warn
6262    \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
6263  }%
6264  {%
6265    \glsifattribute{#1}{glossname}{uc}%
6266  {%

```

Hide the label from the upper-casing command.

```

6267    \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6268    \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6269  }%
6270  {%

```

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

```

6271    \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6272    \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
6273  }%
6274  }%
6275  }%

```

Do post-name hook.

```

6276    \glsxtrpostnamehook{#1}%
6277  }%
6278  }
6279 }

```

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

```
6280 \ifpackageloaded{glossaries-accsupp}
6281 {
6282   \renewcommand*{\Glossentryname}[1]{%
6283     \@glsdoifexistsorwarn{#1}%
6284     {%
6285       \glssetabbrvfmt{\glscategory{#1}}%
```

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

```
6286     \glsattribute{#1}{glossnamefont}%
6287     {%
6288       \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6289       \ifcsdef{\@glsxtr@attrval}%
6290       {%
6291         \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6292         }%
6293         {%
6294           \GlossariesExtraWarning{Unknown control sequence name
6295             ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
6296             for entry ‘#1’. Reverting to default \string\glsnamefont}%
6297           \let\@glsxtr@glossnamefont\glsnamefont
6298           }%
6299         }%
6300         {\let\@glsxtr@glossnamefont\glsnamefont}%
6301         \glsnameaccessdisplay
6302         {%
6303           \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6304           }%
6305         {#1}%
```

Do post-name hook:

```
6306     \glsxtrpostnamehook{#1}%
6307     }%
6308   }
6309 }
6310 {
6311   \renewcommand*{\Glossentryname}[1]{%
6312     \@glsdoifexistsorwarn{#1}%
6313     {%
6314       \glssetabbrvfmt{\glscategory{#1}}%
6315       \glsattribute{#1}{glossnamefont}%
6316       {%
6317         \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6318         \ifcsdef{\@glsxtr@attrval}%
6319         {%
6320           \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6321           }%
6322           {%
6323             \GlossariesExtraWarning{Unknown control sequence name
6324               ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
```

```

6325         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6326         \let\@glsxtr@glossnamefont\glsnamefont
6327     }%
6328 }%
6329 {\let\@glsxtr@glossnamefont\glsnamefont}%
6330 \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6331     \glsxtrpostnamehook{#1}%
6332 }%
6333 }
6334 }

```

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

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

```

6335 \newcommand*{\glsxtrpostnamehook}[1]{%
6336   \let\@glsnumberformat\@glsxtr@defaultnumberformat
6337   \glsxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

6338   \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

6339   \csuse{glsxtrpostname\glscategory{#1}}%
6340 }

```

`trapostnamehook`

```

6341 \newcommand*{\glsextrapostnamehook}[1]{}%

```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```

6342 \newcommand*{\glsdefpostname}[2]{%
6343   \csdef{glsxtrpostname#1}{#2}%
6344 }

```

`etaccessdisplay`

```

6345 \@ifpackageloaded{glossaries-accsupp}
6346 {
6347   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6348     \ifcsdef{gls#1accessdisplay}%
6349     {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
6350     {%

```

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.

```

6351     \edef\@gls@thisval{#1}%
6352     \@for\@gls@map:=\@gls@keymap\do{%

```

```

6353     \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6354     \ifdequal{\@this@key}{\@gls@thisval}%
6355     {%
6356         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6357         \@endfortrue
6358     }%
6359     }%
6360 }%
6361 \ifcsdef{gls\@gls@thisval accessdisplay}%
6362 {\letcs\@glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
6363 {\let\@glsxtr@accessdisplay\@firstoftwo}%
6364 }%
6365 }
6366 }
6367 {%
6368 \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6369 \let\@glsxtr@accessdisplay\@firstoftwo}
6370 }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

6371 \newrobustcmd*{\glossentrynameother}[2]{%
6372 \@glsdoifexistsorwarn{#1}%
6373 }%

```

Accessibility support:

```

6374 \glsxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6375 \glssetabbrvfmt{\glscategory{#1}}%
6376 \glsattribute{#1}{glossnamefont}%
6377 {%
6378 \edef\@glsxtr@attrval{\glsattribute{#1}{glossnamefont}}%
6379 \ifcsdef{\@glsxtr@attrval}%
6380 {%
6381 \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6382 }%
6383 }%
6384 \GlossariesExtraWarning{Unknown control sequence name
6385 '\@glsxtr@attrval' supplied in glossnamefont attribute
6386 for entry '#1'. Reverting to default \string\glsnamefont}%
6387 \let\@glsxtr@glossnamefont\glsnamefont
6388 }%
6389 }%
6390 {\let\@glsxtr@glossnamefont\glsnamefont}%
6391 \glsifattribute{#1}{glossname}{firstuc}%
6392 {%
6393 \@glsxtr@accessdisplay
6394 {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6395 }%

```

```

6396 }%
6397 {%
6398   \glsifattribute{#1}{glossname}{title}%
6399   {%
6400     \@glsxtr@do@titlecaps@warn
6401     \@glsxtr@accessdisplay
6402     {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
6403     {#1}%
6404   }%
6405   {%
6406     \glsifattribute{#1}{glossname}{uc}%
6407     {%
6408       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6409       \@glsxtr@accessdisplay
6410       {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6411       {#1}%
6412     }%
6413     {%
6414       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6415       \@glsxtr@accessdisplay
6416       {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6417       {#1}%
6418     }%
6419   }%
6420 }%

```

Do post-name hook.

```

6421   \glsxtrpostnamehook{#1}%
6422 }%
6423 }

```

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

```

6424 \newif\if@glsxtr@format@override
6425 \@glsxtr@format@overridefalse

```

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

`xFormatOverride`

```

6426 \@ifpackageloaded{hyperref}
6427 {

```

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

```

6428   \ifHy@hyperindex
6429     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6430       \@glsxtr@format@override true
6431       \appto\theindex{\let\glshypernumber\@firstofone}%
6432     }
6433   \else

```

```

6434 \newcommand*\GlsXtrEnableIndexFormatOverride}{%
6435 \@glsxtr@format@overridetrue
6436 \appto\theindex{\let\glshypernumber\hyperpage}%
6437 }
6438 \fi
6439 }
6440 {
6441 \newcommand*\GlsXtrEnableIndexFormatOverride}{%
6442 \@glsxtr@format@overridetrue
6443 }
6444 }
6445 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6446 \newcommand*\glsxtrdoautoindexname}[2]{%
6447 \gls@attribute{#1}{#2}%
6448 {%

```

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.

```

6449 \@glsxtr@autoindex@setname{#1}%

```

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

```

6450 \protected@edef\@glsxtr@attrval{\gls@attribute{#1}{#2}}%
6451 \if@glsxtr@format@override

6452 \ifx\@glsnumberformat\@glsxtr@defaultnumberformat
6453 \else
6454 \let\@glsxtr@attrval\@glsnumberformat
6455 \fi
6456 \fi
6457 \ifdefstring{\@glsxtr@attrval}{true}%
6458 {}%
6459 {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}%
6460 \expandafter\glsxtrautoindex\expandafter{\@glo@name}%
6461 }%
6462 {}%
6463 }

```

glsxtrautoindex

```

6464 \newcommand*\glsxtrautoindex}{\index}

```

xtrautoindexesc

```

6465 \newcommand{\glsxtrautoindexesc}{%
6466 \@gls@checkmkidxchars\@glo@sort
6467 \@glsxtr@autoindex@doextra@esc\@glo@sort
6468 }

```

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

```

6469 \newcommand*\@glsxtr@autoindex@setname}[1]{%

```



```

6470 \protected@edef\@glo@name{\glxtrautoindexentry{#1}}%
6471 \glxtrautoindexassignsort{\@glo@sort}{#1}%
6472 \glxtrautoindexesc
6473 \epreto\@glo@name{\@glo@sort\@glxtr@autoindex@at}%
6474 }

```

`autoindexentry` Command used for the actual part when auto-indexing.

```
6475 \newcommand*\glxtrautoindexentry}[1]{\string\glstryname{#1}}
```

`indexassignsort` Used to assign the sort value when auto-indexing.

```

6476 \newcommand*\glxtrautoindexassignsort}[2]{%
6477 \glsletentryfield{#1}{#2}{sort}%
6478 }

```

`indexdoextra@esc`

```
6479 \newcommand*\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```

6480 \ifx\@glxtr@autoindex@esc\@gls@quotechar
6481 \else
6482 \def\@gls@checkedmkidx{}%
6483 \edef\@@glxtr@checkspch{%
6484 \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6485 \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6486 \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6487 \@@glxtr@checkspch
6488 \let#1\@gls@checkedmkidx\relax
6489 \fi

```

Escape actual character unless it has already been escaped.

```

6490 \ifx\@glxtr@autoindex@at\@gls@actualchar
6491 \else
6492 \def\@gls@checkedmkidx{}%
6493 \edef\@@glxtr@checkspch{%
6494 \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6495 \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6496 \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6497 \@@glxtr@checkspch
6498 \let#1\@gls@checkedmkidx\relax
6499 \fi

```

Escape level character unless it has already been escaped.

```

6500 \ifx\@glxtr@autoindex@level\@gls@levelchar
6501 \else
6502 \def\@gls@checkedmkidx{}%
6503 \edef\@@glxtr@checkspch{%
6504 \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6505 \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6506 \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6507 \@@glxtr@checkspch

```

```

6508 \let#1\@gls@checkedmkidx\relax
6509 \fi

```

Escape encap character unless it has already been escaped.

```

6510 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6511 \else
6512 \def\@gls@checkedmkidx{ }%
6513 \edef\@glsxtr@checkspch{%
6514 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6515 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6516 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6517 \@glsxtr@checkspch
6518 \let#1\@gls@checkedmkidx\relax
6519 \fi
6520 }

```

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`.

```

6521 \newcommand*{\@glsxtr@autoindex@at}{}

```

`trSetActualChar` Set the actual character.

```

6522 \newcommand*{\GlsXtrSetActualChar}[1]{%
6523 \gdef\@glsxtr@autoindex@at{#1}%
6524 \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6525 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6526 }%
6527 }
6528 \@onlypreamble\GlsXtrSetActualChar
6529 \makeatother
6530 \GlsXtrSetActualChar{@}
6531 \makeatletter

```

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

```

6532 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`XtrSetEncapChar` Set the encap character.

```

6533 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6534 \gdef\@glsxtr@autoindex@encap{#1}%
6535 \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
6536 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6537 }%
6538 }
6539 \GlsXtrSetEncapChar{|}
6540 \@onlypreamble\GlsXtrSetEncapChar

```

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

```

6541 \newcommand*{\@glsxtr@autoindex@level}{}

```

XtrSetLevelChar Set the encap character.

```
6542 \newcommand*\GlsXtrSetLevelChar}[1]{%
6543   \gdef\@glsxtr@autoindex@level{#1}%
6544   \def\@glsxtr@autoindex@esclevel##1##2##3\@glsxtr@endescspch{%
6545     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
6546   }%
6547 }
6548 \GlsXtrSetLevelChar{!}
6549 \@onlypreamble\GlsXtrSetLevelChar
```

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

```
6550 \newcommand*\@glsxtr@autoindex@esc{"}
```

lsXtrSetEscChar Set the escape character.

```
6551 \newcommand*\GlsXtrSetEscChar}[1]{%
6552   \gdef\@glsxtr@autoindex@esc{#1}%
6553   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6554     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6555   }%
6556 }
6557 \GlsXtrSetEscChar{"}
6558 \@onlypreamble\GlsXtrSetEscChar
```

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

```
6559 \ifdef\actualchar
6560 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6561 {}
```

Quote character \quotechar:

```
6562 \ifdef\quotechar
6563 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6564 {}
```

Level character \levelchar:

```
6565 \ifdef\levelchar
6566 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6567 {}
```

Encap character \encapchar:

```
6568 \ifdef\encapchar
6569 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6570 {}
```

leto@endescspch

```
6571 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

```
\@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```

6572 \newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
6573   \gls@tmpb=\expandafter{\gls@checkedmkidx}%
6574   \toks@={#3}%
6575   \ifx\@nnil#3\relax
6576     \def\@@glsxtr@checkspch{\glsxtr@gobbleto@endescspch#5\glsxtr@endescspch}%
6577   \else
6578     \ifx\@nnil#4\relax
6579       \edef\gls@checkedmkidx{\the\gls@tmpb\the\toks@}%
6580       \def\@@glsxtr@checkspch{\glsxtr@gobbleto@endescspch
6581         #4#5\glsxtr@endescspch}%
6582     \else
6583       \edef\gls@checkedmkidx{\the\gls@tmpb\the\toks@
6584         \glsxtr@autoindex@esc#1}%
6585       \def\@@glsxtr@checkspch{#2#5#1\@nnil#1\glsxtr@endescspch}%
6586     \fi
6587   \fi
6588   \@@glsxtr@checkspch
6589 }

```

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

```

6590 \renewcommand*{\Glossentrydesc}[1]{%
6591   \glsdoifexistsorwarn{#1}%
6592   {%
6593     \glssetabbrvfmt{\glscategory{#1}}%
6594     \Glsaccessdesc{#1}%
6595   }%
6596 }

```

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

```

6597 \renewcommand*{\Glossentrysymbol}[1]{%
6598   \glsdoifexistsorwarn{#1}%
6599   {%
6600     \glssetabbrvfmt{\glscategory{#1}}%
6601     \Glsaccesssymbol{#1}%
6602   }%
6603 }

```

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

```

6604 \renewcommand*{\Glossentrysymbol}[1]{%
6605   \glsdoifexistsorwarn{#1}%
6606   {%
6607     \glssetabbrvfmt{\glscategory{#1}}%
6608     \Glsaccesssymbol{#1}%
6609   }%
6610 }

```

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

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

```
6611 \newcommand*{\GlsXtrEnableInitialTagging}{%
6612   \ifstar\s@glstr@enabletagging\@glstr@enabletagging
6613 }
6614 \@onlypreamble\GlsXtrEnableInitialTagging
```

r@enabletagging Starred version undefines command.

```
6615 \newcommand*{\s@glstr@enabletagging}[2]{%
6616   \undef#2%
6617   \@glstr@enabletagging{#1}{#2}%
6618 }
```

r@enabletagging Internal command.

```
6619 \newcommand*{\@glstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6620   \@for\@glstr@cat:=#1\do
6621   {%
6622     \ifdefempty\@glstr@cat
6623     {}%
6624     {\glsssetcategoryattribute{\@glstr@cat}{tagging}{true}}%
6625   }%
6626   \newrobustcmd*#2[1]{##1}%
6627   \def\@glstr@taggingcs{#2}%
6628   \renewcommand*\@glstr@activate@initialtagging{%
6629     \let#2\@glstr@tag
6630   }%
6631   \ifundef\@gls@preglossaryhook
6632   {\GlossariesExtraWarning{Initial tagging requires at least
6633     glossaries.sty v4.19 to work correctly}}%
6634   {}%
6635 }
```

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

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

```
6636 \ifundef\mfu@checkword@do
6637 {
6638   \newcommand*{\mfu@checkword@do}[1]{%
6639     \ifdefstring{\mfu@checkword@arg}{#1}%
6640     {%
6641       \let\@mfu@domakefirstuc\@firstofone
6642       \listbreak
6643     }%
6644     {}%
6645   }
```

`\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.

```
6646 \ifundef\mfu@checkword
6647 {
6648   \newcommand{\@glxtr@do@titlecaps@warn}{%
6649     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6650       support not available}%
6651     \let\@glxtr@do@titlecaps@warn\relax
6652   }
6653 }
6654 {
6655   \renewcommand*\mfu@checkword}[1]{%
6656     \def\mfu@checkword@arg{#1}%
6657     \let\@mfu@domakefirstuc\makefirstuc
6658     \forlistloop\mfu@checkword@do\@mfu@nocaplist
6659   }
6660 }
6661 }
6662 {}% no patch required
```

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

```
6663 \newcommand*\@glxtr@do@titlecaps@warn{}
```

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

```
6664 \newcommand*\@glxtr@activate@initialtagging{}
```

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

```
6665 \newrobustcmd*\@glxtr@tag}[1]{%
6666   \glxtr@tagfont{#1}{#1}%
6667   \glxtr@tagfont{#1}{#1}%
6668 }
```

`\glxtrtagfont` Used in the glossary.

```
6669 \newcommand*\glxtrtagfont}[1]{\underline{#1}}
```

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

```
6670 \ifdef\@gls@preglossaryhook
6671 {
6672   \renewcommand*\@gls@preglossaryhook}{%
6673     \@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.

```

6674 \ifundef\@glstr@org@postdescription
6675 {%
6676 \let\@glstr@org@postdescription\glspostdescription
6677 \renewcommand*\glspostdescription{%
6678 \ifglstryexists{\glscurrententrylabel}%
6679 {%
6680 \glstrpostdescription
6681 \@glstr@org@postdescription
6682 }%
6683 }%
6684 }%
6685 }%
6686 {}%

```

Enable the options used by \@glstr:

```

6687 \glossxrsetopts
6688 }%
6689 }
6690 {}

```

`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 predefined styles that don't include it.

```

6691 \newcommand*\glstrpostdescription{%
6692 \csuse{glstrpostdesc\glscategory{\glscurrententrylabel}}%
6693 }

```

`postdescgeneral`

```

6694 \newcommand*\glstrpostdescgeneral{}

```

`trpostdescterm`

```

6695 \newcommand*\glstrpostdescterm{}

```

`postdescacronym`

```

6696 \newcommand*\glstrpostdescacronym{}

```

`descabbreviation`

```

6697 \newcommand*\glstrpostdescabbreviation{}

```

`\glsdefpostdesc` Provide a convenient command for defining the post-description hook for the given category.

```

6698 \newcommand*\glsdefpostdesc}[2]{%
6699 \csdef{glstrpostdesc#1}{#2}%
6700 }

```

`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.

```
6701 \renewcommand*{\glspostlinkhook}{%
6702 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6703 }
```

`xtrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
6704 \newcommand*{\glsxtrpostlinkhook}{%
6705 \glsxtrdiscardperiod{\glslabel}%
6706 {\glsxtrpostlinkendsentence}%
6707 {\glsxtrifcustomdiscardperiod
6708 {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}%
6709 {\glsxtrpostlink}%
6710 }%
6711 }
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6712 \newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

`\glsxtrpostlink`

```
6713 \newcommand*{\glsxtrpostlink}{%
6714 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
6715 }
```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glsxtrpostlink`.

```
6716 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6717 \ifthenelse{\equal{#1}{}}%
6718 {\PackageError{glossaries-extra}
6719   {Invalid empty category label in \string\glsdefpostlink}{}}%
6720 {\csdef{glsxtrpostlink#1}{#2}}%
6721 }
```

`linkendsentence` Done by `\glsxtrpostlinkhook` if a full stop is discarded.

```
6722 \newcommand*{\glsxtrpostlinkendsentence}{%
6723 \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
6724 {%
6725 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
  Put the full stop back.
6726 \spacefactor\sfcode'\. \relax
6727 }%
6728 }
```


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

```
6729 \spacefactor\sfcode‘\ . \relax
6730 }%
6731 }
```

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

```
6732 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6733 \glxtrifwasfirstuse{\space\glxtrparen{\glsaccessdesc{\glslabel}}}{}%
6734 }
```

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.

```
6735 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6736 \glxtrifwasfirstuse
6737 {%
6738 \ifglshassymbol{\glslabel}%
6739 {\space\glxtrparen{\glsaccesssymbol{\glslabel}}}%
6740 {}}%
6741 }%
6742 {}%
6743 }
```

LDescOnFirstUse 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.

```
6744 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6745 \glxtrifwasfirstuse
6746 {%
6747 \space\glxtrparen
6748 {%
6749 \ifglshassymbol{\glslabel}%
6750 {\glsaccesssymbol{\glslabel}, }%
6751 {}}%
6752 \glsaccessdesc{\glslabel}%
6753 }%
6754 }%
6755 {}%
6756 }
```

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).

```
6757 \newcommand*{\glxtrdiscardperiod}[3]{%
6758 \glxtrifwasfirstuse
6759 {%
6760 \glsifattribute{#1}{retainfirstuseperiod}{true}%
```

```

6761   {#3}%
6762   {%
6763     \glsifattribute{#1}{discardperiod}{true}%
6764     {%
6765       \glsifplural
6766       {%
6767         \glsifattribute{#1}{pluraldiscardperiod}{true}%
6768         {\glsxtrifperiod{#2}{#3}}%
6769         {#3}%
6770       }%
6771     }%
6772     \glsxtrifperiod{#2}{#3}%
6773   }%
6774 }%
6775 {#3}%
6776 }%
6777 }%
6778 {%
6779   \glsifattribute{#1}{discardperiod}{true}%
6780   {%
6781     \glsifplural
6782     {%
6783       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6784       {\glsxtrifperiod{#2}{#3}}%
6785       {#3}%
6786     }%
6787   }%
6788   \glsxtrifperiod{#2}{#3}%
6789 }%
6790 }%
6791 {#3}%
6792 }%
6793 }

```

`\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`

```
6794 \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 ' ').

```
6795 \newcommand*\glsxtr@punclist{. , ; ? !}
```

`\punctuationmark` Add character to punctuation list.

```
6796 \newcommand*\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`\punctuationmarks` Reset the punctuation list.

```
6797 \newcommand*\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}
```

`\glxtrifpunc` `\glxtrifnextpunc{<true part>}{<>false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```
6798 \newcommand*{\glxtrifnextpunc}[2]{%
6799   \def\reserved@a{#1}%
6800   \def\reserved@b{#2}%
6801   \futurelet\@glspunc@token\glxtr@ifnextpunc
6802 }
```

`sxtr@ifnextpunc`

```
6803 \newcommand*{\glxtr@ifnextpunc}{%
6804   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6805   \reserved@b
6806 }
```

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

```
6807 \newcommand*{\glxtr@ifpunctoken}[1]{%
6808   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6809 }
```

`xtr@ifpunctoken`

```
6810 \def\@glxtr@ifpunctoken#1#2{%
6811   \let\reserved@d=#2%
6812   \ifx\reserved@d\@nnil
6813     \let\glxtr@next\@glxtr@notfoundinlist
6814   \else
6815     \ifx#1\reserved@d
6816       \let\glxtr@next\@glxtr@foundinlist
6817     \else
6818       \let\glxtr@next\@glxtr@ifpunctoken
6819     \fi
6820   \fi
6821   \glxtr@next#1%
6822 }
```

`xtr@foundinlist`

```
6823 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

`@notfoundinlist`

```
6824 \def\@glxtr@notfoundinlist#1{\@secondoftwo}
```

`glxtrdopostpunc` `\glxtrdopostpunc{<code>}`

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```
6825 \newcommand{\glxtrdopostpunc}[1]{%
6826   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6827 }
```

@glxtr@swaptwo

```
6828 \newcommand{\@glxtr@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.

```
6829 \define@key{glxtrabbrv}{category}{%
6830   \edef\glscategorylabel{#1}%
6831   \ifcsdef{@glxtr@current@#1}%
6832     {%
```

Warning should already have been issued.

```
6833   \let\@glxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6834   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6835   \glxtr@applyabbrvstyle{\csname @glxtr@current@#1\endcsname}%
6836   \let\GlsXtrWarnDeprecatedAbbrStyle\@glxtr@orgwarndep
6837 }%
6838 {}%
6839 }
```

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

```
6840 \define@key{glxtrabbrv}{shortplural}{%
6841   \def\@gls@shortpl{#1}%
6842 }
```

Similarly for the long plural form.

```
6843 \define@key{glxtrabbrv}{longplural}{%
6844   \def\@gls@longpl{#1}%
6845 }
```

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

\glsshortpltok

```
6846 \newtoks\glsshortpltok
```

\glslongpltok

```
6847 \newtoks\glslongpltok
```

`gsxtr@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.

```
6848 \newcommand*{\@glsxtr@insertdots}[2]{%
6849   \def#1{}%
6850   \@glsxtr@insert@dots#1#2\@nnil
6851 }
```

`xtr@insert@dots`

```
6852 \newcommand*{\@glsxtr@insert@dots}[2]{%
6853   \ifx\@nnil#2\relax
6854   \let\@glsxtr@insert@dots@next\@gobble
6855   \else
6856   \ifx\relax#2\relax
6857   \else
6858     \appto#1{#2.}%
6859   \fi
6860   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6861   \fi
6862   \@glsxtr@insert@dots@next#1%
6863 }
```

Similarly provide a way of replacing spaces with `\glsxtrwordsep`, which first needs to be defined:

`\glsxtrwordsep`

```
6864 \newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

`\glsxtrword`

```
6865 \newcommand*{\glsxtrword}[1]{#1}
```

`tr@markwordseps`

```
6866 \newcommand*{\@glsxtr@markwordseps}[2]{%
6867   \def#1{}%
6868   \@glsxtr@mark@wordseps#1#2 \@nnil
6869 }
```

`r@mark@wordseps`

```
6870 \def\@glsxtr@mark@wordseps#1#2 #3{%
6871   \ifdefempty{#1}%
6872   {\def#1{\protect\glsxtrword{#2}}}%
6873   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
6874   \ifx\@nnil#3\relax
```

```

6875 \let\@glsxtr@mark@wordseps@next\relax
6876 \else
6877 \def\@glsxtr@mark@wordseps@next{%
6878 \@glsxtr@mark@wordseps#1#3}%
6879 \fi
6880 \@glsxtr@mark@wordseps@next
6881 }

```

`newabbreviation` Define a new generic abbreviation.

```

6882 \newcommand*{\newabbreviation}[4] []{%
6883 \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
6884 }

```

`newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This is just makes it easier to save and restore the original definition.)

```

6885 \newcommand*{\glsxtr@newabbreviation}[4] {%
6886 \glskeylisttok{#1}%
6887 \glslabeltok{#2}%
6888 \glsshorttok{#3}%
6889 \glslongtok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```

6890 \def\glsxtrorgshort{#3}%
6891 \def\glsxtrorglong{#4}%

```

Provide extra settings for hooks (if modified, this command must end with a comma).

```

6892 \def\ExtraCustomAbbreviationFields{}%

```

Initialise accessibility settings if required.

```

6893 \@gls@initaccesskeys

```

Get the category.

```

6894 \def\gls@categorylabel{abbreviation}%
6895 \glsxtr@applyabbrvstyle{\@gls@abbrv@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```

6896 \setkeys*{glsxtr@abbrv}[shortplural,longplural]{#1}%

```

Set the default long plural

```

6897 \def\@gls@longpl{#4\glspluralsuffix}%
6898 \let\@gls@default@longpl\@gls@longpl

```

Has the markwords attribute been set?

```

6899 \glsifcategoryattribute{\gls@categorylabel}{markwords}{true}%
6900 {%
6901 \@glsxtr@markwordseps\@gls@long{#4}%
6902 \expandafter\def\expandafter\@gls@longpl\expandafter
6903 {\@gls@long\glspluralsuffix}%
6904 \let\@gls@default@longpl\@gls@longpl

```

Update \glslongtok.

```

6905 \expandafter\glslongtok\expandafter{\@gls@long}%
6906 }%
6907 {}%

```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```

6908 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6909 {%
6910 \@glsxtr@markwordseps\@gls@short{#3}%
6911 }%
6912 {}%

```

Has the insertdots attribute been set?

```

6913 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6914 {%
6915 \@glsxtr@insertdots\@gls@short{#3}%
6916 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6917 }%
6918 {\def\@gls@short{#3}}%
6919 }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

6920 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6921 {%
6922 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6923 '\abbrvpluralsuffix}%
6924 }%
6925 {}%

```

Has the noshortplural attribute been set?

```

6926 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6927 {%
6928 \let\@gls@shortpl\@gls@short
6929 }%
6930 {%
6931 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6932 \abbrvpluralsuffix}%
6933 }%
6934 }%

```

Update \glsshorttok:

```

6935 \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

6936 \glsxtrnewabbrevpresetkeyhook{#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).

```

6937 \setkeys*{glsxtrabbrv}[category]{#1}%

```

Has the plural been explicitly set?

```

6938 \ifx\@gls@default@longpl\@gls@longpl
6939 \else

```

Has the markwords attribute been set?

```
6940 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6941 {%
6942   \expandafter\@glsxtr@markwordseps\expandafter\@gls@longpl\expandafter
6943     {\@gls@longpl}%
6944   }%
6945   {}%
6946 \fi
```

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

```
6947 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6948 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6949 \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6950 \newabbreviationhook
```

Define this entry:

```
6951 \protected@edef\@do@newglossaryentry{%
6952   \noexpand\newglossaryentry{\the\glslabeltok}%
6953   {%
6954     type=\glsxtrabbrvtype,%
6955     category=abbreviation,%
6956     short={\the\glsshorttok},%
6957     shortplural={\the\glsshortpltok},%
6958     long={\the\glslongtok},%
6959     longplural={\the\glslongpltok},%
6960     name={\the\glsshorttok},%
6961     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6962   \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6963   \the\glskeylisttok
6964   }%
6965 }%
6966 \@do@newglossaryentry
6967 \GlsXtrPostNewAbbreviation
6968 }
```

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

```
6969 \newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```

`NewAbbreviation` Hook used by abbreviation styles.

```
6970 \newcommand*{\GlsXtrPostNewAbbreviation}{}
```

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

```
6971 \newcommand*{\newabbreviationhook}{}
```


reivationFields

```
6972 \newcommand*\CustomAbbreviationFields{}
```

\glstrparen For the parenthetical styles.

```
6973 \newcommand*\glstrparen[1]{(#1)}
```

lsxtrfullformat Full format without case change.

```
6974 \newcommand*\glxtrfullformat[2]{%
```

```
6975 \glstfirstlongfont{\glstaccesslong{#1}}#2\glxtrfullsep{#1}%
```

```
6976 \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshort{#1}}}%
```

```
6977 }
```

lSxtrfullformat Full format with case change.

```
6978 \newcommand*\Glxtrfullformat[2]{%
```

```
6979 \glstfirstlongfont{\Glstaccesslong{#1}}#2\glxtrfullsep{#1}%
```

```
6980 \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshort{#1}}}%
```

```
6981 }
```

xtrfullplformat Plural full format without case change.

```
6982 \newcommand*\glxtrfullplformat[2]{%
```

```
6983 \glstfirstlongfont{\glstaccesslongpl{#1}}#2\glxtrfullsep{#1}%
```

```
6984 \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshortpl{#1}}}%
```

```
6985 }
```

xtrfullplformat Plural full format with case change.

```
6986 \newcommand*\Glxtrfullplformat[2]{%
```

```
6987 \glstfirstlongfont{\Glstaccesslongpl{#1}}#2\glxtrfullsep{#1}%
```

```
6988 \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshortpl{#1}}}%
```

```
6989 }
```

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

```
6990 \newcommand*\glxtrfullsep[1]{\space}
```

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

nlinefullformat Full format without case change.

```
6991 \newcommand*\glstinlinefullformat{\glxtrfullformat}
```

nlinefullformat Full format with case change.

```
6992 \newcommand*\Glxstinlinefullformat{\Glxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
6993 \newcommand*\glstinlinefullplformat{\glxtrfullplformat}
```

inefullplformat Plural full format with case change.

```
6994 \newcommand*\Glxstinlinefullplformat{\Glxtrfullplformat}
```

Redefine `\glsentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glsxtrfull` set of commands instead.

```

\glsentryfull
6995 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}

\Glsentryfull
6996 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}

\glsentryfullpl
6997 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}

\Glsentryfullpl
6998 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}

\glsfirstabbrvfont  Font changing command used for the abbreviation on first use or in the full format.
6999 \newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}

\glsabbrvdefaultfont  Font changing command used for the abbreviation on first use or in the full format.
7000 \newcommand*{\glsabbrvdefaultfont}[1]{\glsabbrvdefaultfont{#1}}

\glsabbrvfont  Font changing command used for the abbreviation on subsequent use.
7001 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}

\glsabbrvdefaultfont
7002 \newcommand*{\glsabbrvdefaultfont}[1]{#1}

\glslongfont  Font changing command used for the long form in commands like \glsxtrlong.
7003 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}

\glslongdefaultfont  Default font changing command used for the long form in commands like \glsxtrlong.
7004 \newcommand*{\glslongdefaultfont}[1]{#1}

\glsfirstlongfont  Font changing command used for the long form on first use or in the full format.
7005 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}

\glsfirstlongdefaultfont
7006 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}

\glsbrvpluralsuffix  Default plural suffix. Allow an alternative default suffix for abbreviations.
7007 \newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}

\glsbrvpluralsuffix  Default plural suffix.
7008 \newcommand*{\glsbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}

```

`\glxtrfull` Full form (no case-change).

```
7009 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\ns@glxtrfull}
7010 \newcommand*\ns@glxtrfull[2][ ]{%
7011 \new@ifnextchar[{\@glxtr@full{#1}{#2}}%
7012 {\@glxtr@full{#1}{#2}[]}%
7013 }
```

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

```
7014 \def\@glxtr@full#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).
7015 \@glxtr@record{#1}{#2}{glslink}%
7016 \glsdoifexists{#2}%
7017 {%
7018 \glssetabbrvfmt{\gls@category{#2}}%
7019 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7020 \let\glsifplural\@secondoftwo
7021 \let\gls@scaps@case\@firstofthree
7022 \let\glsinsert\@empty
7023 \def\gls@customtext{\gls@xtr@inline@full@format{#2}{#3}}%
```

What should `\gls@xtr@ifwasfirstuse` 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.

```
7024 \gls@xtr@setup@full@defs
7025 \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
7026 }%
7027 \gls@postlink@hook
7028 }
```

`tr@setup@full@defs`

```
7029 \newcommand*{\gls@xtr@setup@full@defs}{%
7030 \let\gls@xtr@ifwasfirstuse\@firstoftwo
7031 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
7032 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
7033 \newcommand*\ns@Glsxtrfull[2][ ]{%
7034 \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
7035 {\@Glsxtr@full{#1}{#2}[]}%
7036 }
```

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

```
7037 \def\@Glsxtr@full#1#2[#3]{%
7038 \glsdoifexists{#2}%
7039 {%
7040 \glssetabbrvfmt{\gls@category{#2}}%
```

```

7041 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7042 \let\gl@sifplural\@secondoftwo
7043 \let\gl@scapscase\@secondofthree
7044 \let\gl@sinsert\@empty
7045 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
7046 \glxtrsetupfulldefs
7047 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7048 }%
7049 \glspostlinkhook
7050 }

```

`\GLSxtrfull` Full form (all uppercase).

```

7051 \newrobustcmd*{\GLSxtrfull}{\@gl@hyp@opt\ns@GLSxtrfull}
7052 \newcommand*\ns@GLSxtrfull[2][{}]{%
7053 \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
7054 {\@GLSxtr@full{#1}{#2}[]}%
7055 }

```

`\@GLSxtr@full` Low-level macro:

```

7056 \def\@GLSxtr@full#1#2[#3]{%
7057 \gl@doifexists{#2}%
7058 {%
7059 \gl@setabbrvfmt{\gl@category{#2}}%
7060 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7061 \let\gl@sifplural\@secondoftwo
7062 \let\gl@scapscase\@thirdofthree
7063 \let\gl@sinsert\@empty
7064 \def\glscustomtext{\mfirstucMakeUppercase{\glxtrinlinefullformat{#2}{#3}}}%
7065 \glxtrsetupfulldefs
7066 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7067 }%
7068 \glspostlinkhook
7069 }

```

`\glxtrfullpl` Plural full form (no case-change).

```

7070 \newrobustcmd*{\glxtrfullpl}{\@gl@hyp@opt\ns@glxtrfullpl}
7071 \newcommand*\ns@glxtrfullpl[2][{}]{%
7072 \new@ifnextchar[{\@glxtr@fullpl{#1}{#2}}%
7073 {\@glxtr@fullpl{#1}{#2}[]}%
7074 }

```

`\@glxtr@fullpl` Low-level macro:

```

7075 \def\@glxtr@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).
7076 \@glxtr@record{#1}{#2}{glslink}%
7077 \gl@doifexists{#2}%
7078 {%

```

```

7079 \glssetabbrvfmt{\glscategory{#2}}%
7080 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7081 \let\glsifplural\@firstoftwo
7082 \let\gls caps case\@firstofthree
7083 \let\glsinsert\@empty
7084 \def\gls custom text{\glsxtr inline full pl format{#2}{#3}}%
7085 \glsxtr setup full defs
7086 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7087 }%
7088 \gls post link hook
7089 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

7090 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
7091 \newcommand*\ns@Glsxtrfullpl[2] []{%
7092 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
7093 {\@Glsxtr@fullpl{#1}{#2} []}%
7094 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

7095 \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).
7096 \@glsxtr@record{#1}{#2}{glslink}%
7097 \glsdoifexists{#2}%
7098 {%
7099 \glssetabbrvfmt{\glscategory{#2}}%
7100 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7101 \let\glsifplural\@firstoftwo
7102 \let\gls caps case\@secondofthree
7103 \let\glsinsert\@empty
7104 \def\gls custom text{\Glsxtr inline full pl format{#2}{#3}}%
7105 \glsxtr setup full defs
7106 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7107 }%
7108 \gls post link hook
7109 }

```

`\GLSxtrfullpl` Plural full form (all upper case).

```

7110 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\ns@GLSxtrfullpl}
7111 \newcommand*\ns@GLSxtrfullpl[2] []{%
7112 \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
7113 {\@GLSxtr@fullpl{#1}{#2} []}%
7114 }

```

`\@GLSxtr@fullpl` Low-level macro:

```

7115 \def\@GLSxtr@fullpl#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).

```

7116 \@glsxtr@record{#1}{#2}{glslink}%
7117 \glsdoifexists{#2}%
7118 {%
7119   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7120   \let\glsifplural\@firstoftwo
7121   \let\gls caps case\@thirdofthree
7122   \let\glsinsert\@empty
7123   \def\gls custom text{%
7124     \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
7125   \glsxtrsetupfulldefs
7126   \@gls@link[#1]{#2}{\c sname gls@\glstype @entryfmt\endcsname}%
7127 }%
7128 \gls post link hook
7129 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

7130 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}
  Define the un-starred form. Need to determine if there is a final optional argument
7131 \newcommand*{\ns@glsxtrshort}[2][ ]{%
7132   \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2}[]}%
7133 }

```

Read in the final optional argument:

```

7134 \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).

```

7135 \@glsxtr@record{#1}{#2}{glslink}%
7136 \glsdoifexists{#2}%
7137 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

7138   \glssetabbrvfmt{\gls category{#2}}%
7139   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7140   \let\glsxtrifwasfirstuse\@secondoftwo
7141   \let\glsifplural\@secondoftwo
7142   \let\gls caps case\@firstofthree
7143   \let\glsinsert\@empty
7144   \def\gls custom text{%
7145     \glsabbrvfont{\gls access short{#2}\ifglsxtrinsetinside#3\fi}%
7146     \ifglsxtrinsetinside\else#3\fi
7147   }%
7148   \@gls@link[#1]{#2}{\c sname gls@\glstype @entryfmt\endcsname}%
7149 }%
7150 \gls post link hook
7151 }

```

`\Glsxtrshort`

```
7152 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7153 \newcommand*{\ns@Glsxtrshort}[2] [] {%
```

```
7154   \new@ifnextchar [{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} [] }%
```

```
7155 }
```

Read in the final optional argument:

```
7156 \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).

```
7157   \@glsxtr@record{#1}{#2}{glslink}%
```

```
7158   \glsdoifexists{#2}%
```

```
7159   {%
```

```
7160     \glssetabbrvfmt{\glscategory{#2}}%
```

```
7161     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
7162     \let\glsxtrifwasfirstuse\@secondoftwo
```

```
7163     \let\glsifplural\@secondoftwo
```

```
7164     \let\glscapscase\@secondofthree
```

```
7165     \let\glsinsert\@empty
```

```
7166     \def\glscustomtext{%
```

```
7167       \glsabbrvfont{\Glsaccessshort{#2}}\ifglsxtrinsertinside#3\fi}%
```

```
7168       \ifglsxtrinsertinside\else#3\fi
```

```
7169     }%
```

```
7170     \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
```

```
7171   }%
```

```
7172   \glspostlinkhook
```

```
7173 }
```

`\GLSxtrshort`

```
7174 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7175 \newcommand*{\ns@GLSxtrshort}[2] [] {%
```

```
7176   \new@ifnextchar [{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} [] }%
```

```
7177 }
```

Read in the final optional argument:

```
7178 \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).

```
7179   \@glsxtr@record{#1}{#2}{glslink}%
```

```
7180   \glsdoifexists{#2}%
```

```
7181   {%
```

```
7182     \glssetabbrvfmt{\glscategory{#2}}%
```

```
7183     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
7184     \let\glsxtrifwasfirstuse\@secondoftwo
```

```
7185     \let\glsifplural\@secondoftwo
```

```

7186 \let\glscapscase\@thirdofthree
7187 \let\glsinsert\@empty
7188 \def\glscustomtext{%
7189 \mfirstucMakeUppercase
7190 {\glsabbrvfont{\glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
7191 \ifglxtrinsertinside\else#3\fi
7192 }%
7193 }%
7194 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7195 }%
7196 \glspostlinkhook
7197 }

```

\glxtrlong

```

7198 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\ns@glxtrlong}
  Define the un-starred form. Need to determine if there is a final optional argument
7199 \newcommand*{\ns@glxtrlong}[2] []{%
7200 \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
7201 }

  Read in the final optional argument:
7202 \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).
7203 \@glsxtr@record{#1}{#2}{glslink}%
7204 \glsdoifexists{#2}%
7205 {%
7206 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7207 \let\glsxtrifwasfirstuse\@secondoftwo
7208 \let\glsifplural\@secondoftwo
7209 \let\glscapscase\@firstofthree
7210 \let\glsinsert\@empty
7211 \def\glscustomtext{%
7212 \glslongfont{\glsaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7213 \ifglxtrinsertinside\else#3\fi
7214 }%
7215 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7216 }%
7217 \glspostlinkhook
7218 }

```

\Glsxtrlong

```

7219 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
  Define the un-starred form. Need to determine if there is a final optional argument
7220 \newcommand*{\ns@Glsxtrlong}[2] []{%
7221 \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
7222 }

```


Read in the final optional argument:

```
7223 \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).

```
7224 \@glsxtr@record{#1}{#2}{glslink}%
7225 \glsdoifexists{#2}%
7226 {%
7227 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7228 \let\glsxtrifwasfirstuse\@secondoftwo
7229 \let\glsifplural\@secondoftwo
7230 \let\gls caps case\@secondofthree
7231 \let\glsinsert\@empty
7232 \def\gls custom text{%
7233 \gls long font{\Gls access long{#2}\ifglsxtrinsetinside#3\fi}%
7234 \ifglsxtrinsetinside\else#3\fi
7235 }%
7236 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7237 }%
7238 \gls post link hook
7239 }
```

\GLSxtrlong

```
7240 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7241 \newcommand*{\ns@GLSxtrlong}[2][ ]{%
7242 \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[ ]}%
7243 }
```

Read in the final optional argument:

```
7244 \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).

```
7245 \@glsxtr@record{#1}{#2}{glslink}%
7246 \glsdoifexists{#2}%
7247 {%
7248 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7249 \let\glsxtrifwasfirstuse\@secondoftwo
7250 \let\glsifplural\@secondoftwo
7251 \let\gls caps case\@thirdofthree
7252 \let\glsinsert\@empty
7253 \def\gls custom text{%
7254 \mfirstucMakeUppercase
7255 {\gls long font{\Gls access long{#2}\ifglsxtrinsetinside#3\fi}%
7256 \ifglsxtrinsetinside\else#3\fi
7257 }%
7258 }%
7259 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

7260 }%
7261 \glspostlinkhook
7262 }

```

Plural short forms:

`\glsxtrshortpl`

```

7263 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7264 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
7265   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7266 }

```

Read in the final optional argument:

```

7267 \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).

```

7268   \@glsxtr@record{#1}{#2}{glslink}%
7269   \glsdoifexists{#2}%
7270   {%
7271     \glssetabbrvfmt{\glscategory{#2}}%
7272     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7273     \let\glsxtrifwasfirstuse\@secondoftwo
7274     \let\glsifplural\@firstoftwo
7275     \let\glscapscase\@firstofthree
7276     \let\glsinsert\@empty
7277     \def\glscustomtext{%
7278       \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7279       \ifglsxtrininsertinside\else#3\fi
7280     }%
7281     \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7282   }%
7283   \glspostlinkhook
7284 }

```

`\Glsxtrshortpl`

```

7285 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7286 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
7287   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7288 }

```

Read in the final optional argument:

```

7289 \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).

```

7290   \@glsxtr@record{#1}{#2}{glslink}%

```

```

7291 \glsdoifexists{#2}%
7292 {%
7293   \glssetabbrvfmt{\glscategory{#2}}%
7294   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7295   \let\glsxtrifwasfirstuse\@secondoftwo
7296   \let\glsifplural\@firstoftwo
7297   \let\glscapscase\@secondofthree
7298   \let\glsinsert\@empty
7299   \def\glscustomtext{%
7300     \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7301     \ifglsxtrininsertinside\else#3\fi
7302   }%
7303   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7304 }%
7305 \glspostlinkhook
7306 }

```

\GLSxtrshortpl

```

7307 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7308 \newcommand*\ns@GLSxtrshortpl}[2] [] {%
7309   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}]%
7310 }

```

Read in the final optional argument:

```

7311 \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).

```

7312   \@glsxtr@record{#1}{#2}{glslink}%
7313   \glsdoifexists{#2}%
7314   {%
7315     \glssetabbrvfmt{\glscategory{#2}}%
7316     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7317     \let\glsxtrifwasfirstuse\@secondoftwo
7318     \let\glsifplural\@firstoftwo
7319     \let\glsapsacscase\@thirdofthree
7320     \let\glsinsert\@empty
7321     \def\glscustomtext{%
7322       \mfirstucMakeUppercase
7323       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7324       \ifglsxtrininsertinside\else#3\fi
7325     }%
7326     }%
7327     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7328   }%
7329   \glspostlinkhook
7330 }

```

Plural long forms:

`\glxtrlongpl`

```
7331 \newrobustcmd*{\glxtrlongpl}{\@gls@hyp@opt\ns@glxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7332 \newcommand*{\ns@glxtrlongpl}[2] [] {%
```

```
7333 \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
```

```
7334 }
```

Read in the final optional argument:

```
7335 \def\@glxtrlongpl#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).

```
7336 \@glxtr@record{#1}{#2}{glslink}%
```

```
7337 \glsdoifexists{#2}%
```

```
7338 {%
```

```
7339 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
7340 \let\glxtrifwasfirstuse\@secondoftwo
```

```
7341 \let\glsifplural\@firstoftwo
```

```
7342 \let\glscapscase\@firstofthree
```

```
7343 \let\glsinsert\@empty
```

```
7344 \def\glscustomtext{%
```

```
7345 \glsfont{\glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
```

```
7346 \ifglxtrinsertinside\else#3\fi
```

```
7347 }%
```

```
7348 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
7349 }%
```

```
7350 \glspostlinkhook
```

```
7351 }
```

`\Glsxtrlongpl`

```
7352 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\ns@Glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7353 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
```

```
7354 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
```

```
7355 }
```

Read in the final optional argument:

```
7356 \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).

```
7357 \@glxtr@record{#1}{#2}{glslink}%
```

```
7358 \glsdoifexists{#2}%
```

```
7359 {%
```

```
7360 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
7361 \let\glxtrifwasfirstuse\@secondoftwo
```

```
7362 \let\glsifplural\@firstoftwo
```

```
7363 \let\glscapscase\@secondofthree
```

```
7364 \let\glsinsert\@empty
```

```

7365 \def\glscustomtext{%
7366 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
7367 \ifglxtrinsertinside\else#3\fi
7368 }%
7369 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7370 }%
7371 \glspostlinkhook
7372 }

```

\GLSxtrlongpl

```

7373 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
  Define the un-starred form. Need to determine if there is a final optional argument
7374 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7375 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7376 }

```

Read in the final optional argument:

```

7377 \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).

```

7378 \@glsxtr@record{#1}{#2}{glslink}%
7379 \glsdoifexists{#2}%
7380 {%
7381 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7382 \let\glsxtrifwasfirstuse\@secondoftwo
7383 \let\glsifplural\@firstoftwo
7384 \let\glscapscase\@thirdofthree
7385 \let\glsinsert\@empty
7386 \def\glscustomtext{%
7387 \mfirstucMakeUppercase
7388 {\glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
7389 \ifglxtrinsertinside\else#3\fi
7390 }%
7391 }%
7392 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7393 }%
7394 \glspostlinkhook
7395 }

```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

7396 \newcommand*{\glssetabbrvfmt}[1] {%
7397 \ifcsdef{@glsabbrv@current@#1}%
7398 {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7399 {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7400 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7401 \newrobustcmd*{\glsuseabbrvfont}[2] {{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

```


7433 }%
7434 }%
7435 }%
7436 {%

First use:

7437 \glsifplural
7438 {%

First use plural form:

7439 \glscapscase
7440 {%

First use plural form, don't adjust case:

7441 \glsxtrfullplformat{\glslabel}{\glsinsert}%
7442 }%
7443 {%

First use plural form, make first letter upper case:

7444 \Glsxtrfullplformat{\glslabel}{\glsinsert}%
7445 }%
7446 {%

First use plural form, all caps:

7447 \mfirstucMakeUppercase
7448 {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
7449 }%
7450 }%
7451 {%

First use singular form

7452 \glscapscase
7453 {%

First use singular form, don't adjust case:

7454 \glsxtrfullformat{\glslabel}{\glsinsert}%
7455 }%
7456 {%

First use singular form, make first letter upper case:

7457 \Glsxtrfullformat{\glslabel}{\glsinsert}%
7458 }%
7459 {%

First use singular form, all caps:

7460 \mfirstucMakeUppercase
7461 {\glsxtrfullformat{\glslabel}{\glsinsert}}%
7462 }%
7463 }%
7464 }%
7465 }%
7466 {%

User supplied text.

```
7467 \glscustomtext
7468 }%
7469 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7470 \newcommand*\glxtrsubsequentfmt}[2]{%
7471 \glsabbrvfont{\glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7472 \ifglxtrinsertinside \else#2\fi
7473 }
7474 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
7475 \newcommand*\glxtrsubsequentplfmt}[2]{%
7476 \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7477 \ifglxtrinsertinside \else#2\fi
7478 }
7479 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt
```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```
7480 \newcommand*\Glsxtrsubsequentfmt}[2]{%
7481 \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7482 \ifglxtrinsertinside \else#2\fi
7483 }
7484 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```
7485 \newcommand*\Glsxtrsubsequentplfmt}[2]{%
7486 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7487 \ifglxtrinsertinside \else#2\fi
7488 }
7489 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

abbreviationstyle

```
7490 \newcommand*\setabbreviationstyle}[2][abbreviation]{%
7491 \ifcsundef{@glsabbrv@dispstyle@setup@#2}
7492 {%
7493 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7494 }%
7495 }
```

Have abbreviations already been defined for this category?

```
7496 \ifcsstring{@glsabbrv@current@#1}{#2}%
7497 }
```


Style already set.

```
7498 }%
7499 {%
7500 \def\@glsxtr@dostylewarn{%
7501 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7502 {%
7503 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
7504 style has been switched \MessageBreak
7505 for category ‘#1’, \MessageBreak
7506 but there have already been entries \MessageBreak
7507 defined for this category. Unwanted \MessageBreak
7508 side-effects may result}}%
7509 \@endfortrue
7510 }%
7511 \@glsxtr@dostylewarn
```

Set up the style for the given category.

```
7512 \csdef{@glsabbrv@current@#1}{#2}%
7513 \glsxtr@applyabbrvstyle{#2}%
7514 }%
7515 }%
7516 }
```

`\glsxtr@applyabbrvstyle` Apply the abbreviation style without existence check.

```
7517 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7518 \csuse{@glsabbrv@dispstyle@setup@#1}%
7519 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7520 }
```

`\glsxtr@applyabbrvfmt` Only apply the style formats.

```
7521 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7522 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7523 }
```

`\glsxtr@newabbreviationstyle` This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
7524 \newcommand*{\newabbreviationstyle}[3]{%
7525 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7526 {%
7527 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7528 defined}{}%
7529 }%
7530 {%
7531 \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
7532 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7533 #2}%
7534 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
7535 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7536 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7537 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7538 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

Reset `\glxtrsubsequentfmt` etc in case a style changes this.

```
7539 \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
7540 \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
7541 \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7542 \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7543 #3}%
7544 }%
7545 }
```

breivationstyle

```
7546 \newcommand*{\renewabbreviationstyle}[3]{%
7547 \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7548 {%
7549 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7550 }%
7551 {%
7552 \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
7553 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7554 #2}%
7555 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
7556 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7557 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7558 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7559 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7560 #3}%
7561 }%
7562 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
7563 \newcommand*{\letabbreviationstyle}[2]{%
7564 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7565 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7566 }
```

ecated@abbrstyle

```
\@glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```
7567 \newcommand*{\@GlsXtrDeprecatedAbbrStyle}[2]{%
7568   \csdef{@Glsabbrv@dispstyle@setup@#1}{%
7569     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7570     \csuse{@Glsabbrv@dispstyle@setup@#2}%
7571   }%
7572   \csletcs{@Glsabbrv@dispstyle@fmts@#1}{@Glsabbrv@dispstyle@fmts@#2}%
7573 }
```

`ecatedAbbrStyle` Generate warning for deprecated style use.

```
7574 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7575   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7576   use ‘#2’ instead}%
7577 }
```

`eAbbrStyleSetup`

```
7578 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7579   \ifcsundef{@Glsabbrv@dispstyle@setup@#1}%
7580   {%
7581     \PackageError{glossaries-extra}%
7582     {Unknown abbreviation style definitions ‘#1’}{}%
7583   }%
7584   {%
7585     \csname @Glsabbrv@dispstyle@setup@#1\endcsname
7586   }%
7587 }
```

`seAbbrStyleFmts`

```
7588 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7589   \ifcsundef{@Glsabbrv@dispstyle@fmts@#1}%
7590   {%
7591     \PackageError{glossaries-extra}%
7592     {Unknown abbreviation style formats ‘#1’}{}%
7593   }%
7594   {%
7595     \csname @Glsabbrv@dispstyle@fmts@#1\endcsname
7596   }%
7597 }
```

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.

```
7598 \newif\ifglxtrinsertinside
7599 \glxtrinsertinsidefalse
```

trlongshortname

```
7600 \newcommand*\glxtrlongshortname{%
7601   \protect\glxtrlongshortfont{\the\glsshorttok}%
7602 }
```

long-short

```
7603 \newabbreviationstyle{long-short}%
7604 {%
7605   \renewcommand*\CustomAbbreviationFields{%
7606     name={\glxtrlongshortname},
7607     sort={\the\glsshorttok},
7608     first={\protect\glxtrlongfont{\the\glslongtok}%
7609       \protect\glxtrfullsep{\the\glslabeltok}%
7610       \glxtrparen{\protect\glxtrshortfont{\the\glsshorttok}}},%
7611     firstplural={\protect\glxtrlongfont{\the\glslongtok}%
7612       \protect\glxtrfullsep{\the\glslabeltok}%
7613       \glxtrparen{\protect\glxtrshortfont{\the\glsshortptok}}},%
7614     plural={\protect\glxtrshortfont{\the\glsshortptok}},%
7615     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
7616 \renewcommand*\GlsXtrPostNewAbbreviation{%
7617   \glshasattribute{\the\glslabeltok}{regular}%
7618   {%
7619     \glissetattribute{\the\glslabeltok}{regular}{false}%
7620   }%
7621   {}%
7622 }%
7623 }%
7624 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7625 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7626 \renewcommand*\glxtrshortfont[1]{\glxtrshortfontdefaultfont{##1}}%
7627 \renewcommand*\glxtrlongfont[1]{\glxtrlongfontdefaultfont{##1}}%
7628 \renewcommand*\glxtrfullfont[1]{\glxtrfullfontdefaultfont{##1}}%
7629 \renewcommand*\glxtrfullfont[1]{\glxtrfullfontdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7630 \renewcommand*\glxtrfullfont[2]{%
7631   \glxtrlongfont{\glxtrfullfont{##1}\ifglxtrinsertinside##2\fi}%
7632   \ifglxtrinsertinsideelse##2\fi
7633   \glxtrfullfont{##1}}%
```

```

7634 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7635 }%
7636 \renewcommand*{\glsxtrfullplformat}[2]{%
7637 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7638 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7639 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7640 }%
7641 \renewcommand*{\Glsxtrfullformat}[2]{%
7642 \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7643 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7644 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7645 }%
7646 \renewcommand*{\Glsxtrfullplformat}[2]{%
7647 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7648 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7649 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7650 }%
7651 }

```

Set this as the default style for general abbreviations:

```
7652 \setabbreviationstyle{long-short}
```

ngshortdescsort

```

7653 \newcommand*{\glsxtrlongshortdescsort}{%
7654 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7655 }

```

ngshortdescname

```

7656 \newcommand*{\glsxtrlongshortdescname}{%
7657 \protect\glslongfont{\the\glslongtok}
7658 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7659 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7660 \newabbreviationstyle{long-short-desc}%
7661 {%
7662 \renewcommand*{\CustomAbbreviationFields}{%
7663 name={\glsxtrlongshortdescname},
7664 sort={\glsxtrlongshortdescsort},%
7665 first={\protect\glsfirstlongfont{\the\glslongtok}%
7666 \protect\glsxtrfullsep{\the\glslabeltok}%
7667 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7668 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7669 \protect\glsxtrfullsep{\the\glslabeltok}%
7670 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```
7671 text={\protect\glsabbrvfont{\the\glsshorttok}},%
```

```

7672   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7673 }%

```

Unset the regular attribute if it has been set.

```

7674 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7675   \glsattribute{\the\glslabeltok}{regular}%
7676   {%
7677     \glssetattribute{\the\glslabeltok}{regular}{false}%
7678   }%
7679   {}%
7680 }%
7681 }%
7682 {%
7683   \GlsXtrUseAbbrStyleFmts{long-short}%
7684 }

```

trshortlongname

```

7685 \newcommand*{\glsxtrshortlongname}{%
7686   \protect\glsabbrvfont{\the\glsshorttok}%
7687 }

```

short-long Short form followed by long form in parenthesis on first use.

```

7688 \newabbreviationstyle{short-long}%
7689 {%
7690   \renewcommand*{\CustomAbbreviationFields}{%
7691     name={\glsxtrshortlongname},
7692     sort={\the\glsshorttok},
7693     description={\the\glslongtok},%
7694     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7695       \protect\glsxtrfullsep{\the\glslabeltok}%
7696       \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7697     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7698       \protect\glsxtrfullsep{\the\glslabeltok}%
7699       \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7700     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7701 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7702   \glsattribute{\the\glslabeltok}{regular}%
7703   {%
7704     \glssetattribute{\the\glslabeltok}{regular}{false}%
7705   }%
7706   {}%
7707 }%
7708 }%
7709 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7710 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%

```

```

7711 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7712 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7713 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7714 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7715 \renewcommand*\glsxtrfullformat}[2]{%
7716   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7717   \ifglsxtrininsertinside\else##2\fi
7718   \glsxtrfullsep{##1}%
7719   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7720 }%
7721 \renewcommand*\glsxtrfullplformat}[2]{%
7722   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7723   \ifglsxtrininsertinside\else##2\fi
7724   \glsxtrfullsep{##1}%
7725   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7726 }%
7727 \renewcommand*\Glsxtrfullformat}[2]{%
7728   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7729   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7730   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7731 }%
7732 \renewcommand*\Glsxtrfullplformat}[2]{%
7733   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7734   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7735   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7736 }%
7737 }

```

ortlongdescsort

```

7738 \newcommand*\glsxtrshortlongdescsort{\the\glsshorttok}

```

ortlongdescname

```

7739 \newcommand*\glsxtrshortlongdescname{%
7740   \protect\glsabbrvfont{\the\glsshorttok}
7741   \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
7742 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7743 \newabbreviationstyle{short-long-desc}%
7744 {%
7745   \renewcommand*\CustomAbbreviationFields{%
7746     name={\glsxtrshortlongdescname},
7747     sort={\glsxtrshortlongdescsort},
7748     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7749     \protect\glsxtrfullsep{\the\glslabeltok}}%
7750     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7751     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%

```

```

7752 \protect\glxtrfullsep{\the\glslabeltok}%
7753 \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7754 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7755 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7756 }%

```

Unset the regular attribute if it has been set.

```

7757 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7758 \glshasattribute{\the\glslabeltok}{regular}%
7759 {%
7760 \glissetattribute{\the\glslabeltok}{regular}{false}%
7761 }%
7762 {}%
7763 }%
7764 }%
7765 {%
7766 \GlsXtrUseAbbrStyleFmts{short-long}%
7767 }

```

`\longfootnotefont` Only used by the “footnote” styles.

```
7768 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

`\longfootnotefont` Only used by the “footnote” styles.

```
7769 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

`\glxtrabbrvfootnote` `\glxtrabbrvfootnote{<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`).

```
7770 \newcommand*{\glxtrabbrvfootnote}[2]{\footnote{#2}}
```

`\glxtrfootnotename`

```

7771 \newcommand*{\glxtrfootnotename}{%
7772 \protect\glsabbrvfont{\the\glsshorttok}}%
7773 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

7774 \newabbreviationstyle{footnote}%
7775 {%
7776 \renewcommand*{\CustomAbbreviationFields}{%
7777 name={\glxtrfootnotename},
7778 sort={\the\glsshorttok},
7779 description={\the\glslongtok},%

```



```

7780 first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7781 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7782 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7783 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7784 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7785 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

7786 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.

```

7787 \renewcommand*\GlsXtrPostNewAbbreviation}{%
7788 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7789 \glsattribute{\the\glslabeltok}{regular}%
7790 {%
7791 \glssetattribute{\the\glslabeltok}{regular}{false}%
7792 }%
7793 {}%
7794 }%
7795 }%
7796 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7797 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7798 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7799 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7800 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7801 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7802 \renewcommand*\glsxtrfullformat[2]{%
7803 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinertinside##2\fi}%
7804 \ifglsxtrinertinside\else##2\fi
7805 \protect\glsxtrabbrvfootnote{##1}%
7806 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7807 }%
7808 \renewcommand*\glsxtrfullplformat[2]{%
7809 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinertinside##2\fi}%
7810 \ifglsxtrinertinside\else##2\fi
7811 \protect\glsxtrabbrvfootnote{##1}%
7812 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7813 }%
7814 \renewcommand*\Glsxtrfullformat[2]{%
7815 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinertinside##2\fi}%
7816 \ifglsxtrinertinside\else##2\fi
7817 \protect\glsxtrabbrvfootnote{##1}%
7818 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7819 }%
7820 \renewcommand*\Glsxtrfullplformat[2]{%
7821 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinertinside##2\fi}%

```

```

7822 \ifglxtrinsertinside\else##2\fi
7823 \protect\glxtrabbrvfootnote{##1}%
7824 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7825 }%

```

The first use full form and the inline full form use the short (long) style.

```

7826 \renewcommand*{\glxtrinlinefullformat}[2]{%
7827 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7828 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7829 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7830 }%
7831 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7832 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7833 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7834 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7835 }%
7836 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7837 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7838 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7839 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7840 }%
7841 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7842 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7843 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7844 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7845 }%
7846 }

```

short-footnote

```
7847 \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.

```

7848 \newabbreviationstyle{postfootnote}%
7849 {%
7850 \renewcommand*{\CustomAbbreviationFields}{%
7851 name={\glxtrfootnotename},
7852 sort={\the\glsshorttok},
7853 description={\the\glslongtok},%
7854 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7855 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7856 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.

```

7857 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7858 \csdef{glxtrpostlink\glscategorylabel}{%
7859 \glxtrifwasfirstuse

```

7860 {%

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
7861        \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
7862        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7863        }%
7864        {%
7865        }%
7866        \glshasattribute{\the\glslabeltok}{regular}%
7867        {%
7868        \glissetattribute{\the\glslabeltok}{regular}{false}%
7869        }%
7870        {%
7871        }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7872 \renewcommand*\glxtrsetupfulldefs{%
7873   \let\glxtrifwasfirstuse\@secondoftwo
7874 }%
7875 }%
7876 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7877 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7878 \renewcommand*\glssabrvfont[1]{\glssabrvdefaultfont{##1}}%
7879 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7880 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7881 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7882 \renewcommand*\glxtrfullformat[2]{%
7883   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7884   \ifglxtrininsertinside\else##2\fi
7885 }%
7886 \renewcommand*\glxtrfullplformat[2]{%
7887   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7888   \ifglxtrininsertinside\else##2\fi
7889 }%
7890 \renewcommand*\Glsxtrfullformat[2]{%
7891   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7892   \ifglxtrininsertinside\else##2\fi
7893 }%
7894 \renewcommand*\Glsxtrfullplformat[2]{%
7895   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7896   \ifglxtrininsertinside\else##2\fi
7897 }%
```

The first use full form and the inline full form use the short (long) style.

```
7898 \renewcommand*\glxtrininlinefullformat[2]{%
```

```

7899   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7900   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7901   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7902 }%
7903 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7904   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7905   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7906   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7907 }%
7908 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7909   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7910   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7911   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7912 }%
7913 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7914   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7915   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7916   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7917 }%
7918 }

```

short-postfootnote

```
7919 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```

7920 \newcommand*{\glxtrshortnolongname}{%
7921   \protect\glsabbrvfont{\the\glsshorttok}%
7922 }

```

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.

```

7923 \newabbreviationstyle{short}{%
7924 {%
7925   \renewcommand*{\CustomAbbreviationFields}{%
7926     name={\glxtrshortnolongname},
7927     sort={\the\glsshorttok},
7928     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7929     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7930     text={\protect\glsabbrvfont{\the\glsshorttok}},
7931     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7932     description={\the\glslongtok}}%
7933 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7934   \glssetAttribute{\the\glslabeltok}{regular}{true}}%
7935 }%
7936 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7937 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7938 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7939 \renewcommand*\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7940 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7941 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7942 \renewcommand*\glxtrinlinefullformat}[2]{%
7943   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7944   \ifglxtrinsertinside##2\fi}%
7945   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7946   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7947 }%
7948 \renewcommand*\glxtrinlinefullplformat}[2]{%
7949   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7950   \ifglxtrinsertinside##2\fi}%
7951   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7952   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7953 }%
7954 \renewcommand*\Glsxtrinlinefullformat}[2]{%
7955   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7956   \ifglxtrinsertinside##2\fi}%
7957   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7958   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7959 }%
7960 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
7961   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7962   \ifglxtrinsertinside##2\fi}%
7963   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7964   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7965 }%

```

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.

```

7966 \renewcommand*\glxtrfullformat}[2]{%
7967   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7968   \ifglxtrinsertinside\else##2\fi
7969 }%
7970 \renewcommand*\glxtrfullplformat}[2]{%
7971   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7972   \ifglxtrinsertinside\else##2\fi
7973 }%
7974 \renewcommand*\Glsxtrfullformat}[2]{%
7975   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7976   \ifglxtrinsertinside\else##2\fi
7977 }%
7978 \renewcommand*\Glsxtrfullplformat}[2]{%
7979   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7980   \ifglxtrinsertinside\else##2\fi
7981 }%

```

7982 }

Set this as the default style for acronyms:

7983 \setabbreviationstyle[acronym]{short}

short-nolong

7984 \letabbreviationstyle{short-nolong}{short}

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

7985 \newabbreviationstyle{short-nolong-noreg}%

7986 {%

7987 \GlsXtrUseAbbrStyleSetup{short-nolong}%

Unset the regular attribute if it has been set.

7988 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7989 \glshasattribute{\the\glslabeltok}{regular}%

7990 {%

7991 \glissetattribute{\the\glslabeltok}{regular}{false}%

7992 }%

7993 {}%

7994 }%

7995 }%

7996 {%

7997 \GlsXtrUseAbbrStyleFmts{short-nolong}%

7998 }

trshortdescname

7999 \newcommand*{\glxtrshortdescname}{%

8000 \protect\glssabrvfont{\the\glsshorttok}%

8001 \protect\glsxtrfullsep{\the\glslabeltok}%

8002 \protect\glsxtrparen{\protect\glslongfont{\the\glslongtok}}%

8003 }

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.

8004 \newabbreviationstyle{short-desc}%

8005 {%

8006 \renewcommand*{\CustomAbbreviationFields}{%

8007 name={\glxtrshortdescname},

8008 sort={\the\glsshorttok},

8009 first={\protect\glssabrvfont{\the\glsshorttok}},

8010 firstplural={\protect\glssabrvfont{\the\glsshortpltok}},

8011 text={\protect\glssabrvfont{\the\glsshorttok}},

8012 plural={\protect\glssabrvfont{\the\glsshortpltok}}}%

8013 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

8014 \glissetattribute{\the\glslabeltok}{regular}{true}}%

8015 }%

8016 {%

In case the user wants to mix and match font styles, these are redefined here.

```
8017 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
8018 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
8019 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
8020 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8021 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8022 \renewcommand*\glsxtrinlinefullformat[2]{%
8023   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8024   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8025   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8026 }%
8027 \renewcommand*\glsxtrinlinefullplformat[2]{%
8028   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8029   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8030   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8031 }%
8032 \renewcommand*\Glsxtrinlinefullformat[2]{%
8033   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8034   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8035   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8036 }%
8037 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8038   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8039   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8040   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8041 }%
```

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.

```
8042 \renewcommand*\glsxtrfullformat[2]{%
8043   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8044   \ifglsxtrininsertinside\else##2\fi
8045 }%
8046 \renewcommand*\glsxtrfullplformat[2]{%
8047   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8048   \ifglsxtrininsertinside\else##2\fi
8049 }%
8050 \renewcommand*\Glsxtrfullformat[2]{%
8051   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8052   \ifglsxtrininsertinside\else##2\fi
8053 }%
8054 \renewcommand*\Glsxtrfullplformat[2]{%
8055   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8056   \ifglsxtrininsertinside\else##2\fi
8057 }%
8058 }
```

ort-nolong-desc

```
8059 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```
8060 \newabbreviationstyle{short-nolong-desc-noreg}%
```

```
8061 {%
```

```
8062 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
```

Unset the regular attribute if it has been set.

```
8063 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
8064 \glshasattribute{\the\glslabeltok}{regular}%
```

```
8065 {%
```

```
8066 \glissetattribute{\the\glslabeltok}{regular}{false}%
```

```
8067 }%
```

```
8068 {}%
```

```
8069 }%
```

```
8070 }%
```

```
8071 {%
```

```
8072 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
```

```
8073 }
```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```
8074 \newabbreviationstyle{nolong-short}%
```

```
8075 {%
```

```
8076 \GlsXtrUseAbbrStyleSetup{short-nolong}%
```

```
8077 }%
```

```
8078 {%
```

```
8079 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8080 \renewcommand*{\glxtrinlinefullformat}[2]{%
```

```
8081 \protect\glsfirstlongfont{\glssaccesslong{##1}}%
```

```
8082 \ifglxtrininsertinside##2\fi}%
```

```
8083 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
8084 \glxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}}%
```

```
8085 }%
```

```
8086 \renewcommand*{\glxtrinlinefullplformat}[2]{%
```

```
8087 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%
```

```
8088 \ifglxtrininsertinside##2\fi}%
```

```
8089 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
8090 \glxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}}%
```

```
8091 }%
```

```
8092 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
```

```
8093 \protect\glsfirstlongfont{\glssaccesslong{##1}}%
```

```
8094 \ifglxtrininsertinside##2\fi}%
```

```
8095 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
8096 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}}%
```

```
8097 }%
```

```
8098 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
```

```
8099 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%
```



```

8100     \ifglxtrinsertinside##2\fi}%
8101     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8102     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8103 }%
8104 }

```

`nolong-short-noreg` Like `nolong-short` but doesn't set the regular attribute.

```

8105 \newabbreviationstyle{nolong-short-noreg}%
8106 {%
8107   \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

8108   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8109     \glshasattribute{\the\glslabelltok}{regular}%
8110     {%
8111       \glissetattribute{\the\glslabelltok}{regular}{false}%
8112     }%
8113   }%
8114 }%
8115 }%
8116 {%
8117   \GlsXtrUseAbbrStyleFmts{nolong-short}%
8118 }

```

`noshortdescname`

```

8119 \newcommand*{\glxtrlongnoshortdescname}{%
8120   \protect\glslongfont{\the\glslongtok}%
8121 }

```

`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.

```

8122 \newabbreviationstyle{long-desc}%
8123 {%
8124   \renewcommand*{\CustomAbbreviationFields}{%
8125     name={\glxtrlongnoshortdescname},
8126     sort={\the\glslongtok},
8127     first={\protect\glsfirstlongfont{\the\glslongtok}},
8128     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8129     text={\glslongfont{\the\glslongtok}},
8130     plural={\glslongfont{\the\glslongpltok}}%
8131   }%
8132   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8133     \glissetattribute{\the\glslabelltok}{regular}{true}}%
8134 }%
8135 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8136   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

```

```

8137 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
8138 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
8139 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8140 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8141 \renewcommand*\glsxtrsubsequentfmt[2]{%
8142   \glslongfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8143   \ifglsxtrininsertinside \else##2\fi
8144 }%
8145 \renewcommand*\glsxtrsubsequentplfmt[2]{%
8146   \glslongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8147   \ifglsxtrininsertinside \else##2\fi
8148 }%
8149 \renewcommand*\Glsxtrsubsequentfmt[2]{%
8150   \glslongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8151   \ifglsxtrininsertinside \else##2\fi
8152 }%
8153 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
8154   \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8155   \ifglsxtrininsertinside \else##2\fi
8156 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8157 \renewcommand*\glsxtrinlinefullformat[2]{%
8158   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8159   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8160   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8161 }%
8162 \renewcommand*\glsxtrinlinefullplformat[2]{%
8163   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8164   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8165   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8166 }%
8167 \renewcommand*\Glsxtrinlinefullformat[2]{%
8168   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8169   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8170   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8171 }%
8172 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8173   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8174   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8175   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8176 }%

```

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.

```

8177 \renewcommand*\glsxtrfullformat[2]{%
8178   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8179   \ifglsxtrininsertinside\else##2\fi
8180 }%

```

```

8181 \renewcommand*{\glxtrfullplformat}[2]{%
8182   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8183   \ifglxtrinsertinside\else##2\fi
8184 }%
8185 \renewcommand*{\Glsxtrfullformat}[2]{%
8186   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8187   \ifglxtrinsertinside\else##2\fi
8188 }%
8189 \renewcommand*{\Glsxtrfullplformat}[2]{%
8190   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8191   \ifglxtrinsertinside\else##2\fi
8192 }%
8193 }

```

`long-noshort-desc` Provide a synonym that matches similar styles.

```
8194 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

`short-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```

8195 \newabbreviationstyle{long-noshort-desc-noreg}%
8196 {%
8197   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
      Unset the regular attribute if it has been set.
8198   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8199     \glsattribute{\the\glslabeltok}{regular}%
8200     {%
8201       \glssetattribute{\the\glslabeltok}{regular}{false}%
8202     }%
8203   }%
8204 }%
8205 }%
8206 {%
8207   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8208 }

```

`longnoshortname`

```

8209 \newcommand*{\glxtrlongnoshortname}{%
8210   \protect\glsabbrvfont{\the\glsshorttok}%
8211 }

```

`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.

```

8212 \newabbreviationstyle{long}%
8213 {%
8214   \renewcommand*{\CustomAbbreviationFields}{%
8215     name={\glxtrlongnoshortname},
8216     sort={\the\glsshorttok},
8217     first={\protect\glsfirstlongfont{\the\glslongtok}},

```

```

8218   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8219   text={\glslongfont{\the\glslongtok}},
8220   plural={\glslongfont{\the\glslongpltok}},%
8221   description={\the\glslongtok}%
8222 }%
8223 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8224   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8225 }%
8226 {%
8227   \GlsXtrUseAbbrStyleFmts{long-desc}%
8228 }

```

`long-noshort` Provide a synonym that matches similar styles.

```
8229 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```

8230 \newabbreviationstyle{long-noshort-noreg}%
8231 {%
8232   \GlsXtrUseAbbrStyleSetup{long-noshort}%

```

Unset the regular attribute if it has been set.

```

8233 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8234   \glsattribute{\the\glslabeltok}{regular}%
8235   {%
8236     \glssetattribute{\the\glslabeltok}{regular}{false}%
8237   }%
8238   {}}%
8239 }%
8240 }%
8241 {%
8242   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8243 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
8244 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
8245 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
8246 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`firstabbrvscfont` Added for consistent naming.

```
8247 \newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glxtrscsuffix`

```
8248 \newcommand*{\glxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
8249 \newabbreviationstyle{long-short-sc}%
8250 {%
8251   \renewcommand*{\CustomAbbreviationFields}{%
8252     name={\glxtrlongshortname},
8253     sort={\the\glsshorttok},
8254     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
8255       \protect\glxtrfullsep{\the\glslabeltok}%
8256       \glxtrparen{\protect\glfirstabbrvscfont{\the\glsshorttok}}},%
8257     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
8258       \protect\glxtrfullsep{\the\glslabeltok}%
8259       \glxtrparen{\protect\glfirstabbrvscfont{\the\glsshortpltok}}},%
8260     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}},%
8261     description={\the\glslongtok}}%
8262   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8263     \glshasattribute{\the\glslabeltok}{regular}%
8264     {%
8265       \glissetattribute{\the\glslabeltok}{regular}{false}%
8266     }%
8267   }%
8268 }%
8269 }%
8270 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8271 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8272 \renewcommand*{\glsabbrvfnt}[1]{\glsabbrvscfont{##1}}%
8273 \renewcommand*{\glfirstabbrvfnt}[1]{\glfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
8274 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8275 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8276 \renewcommand*{\glxtrfullformat}[2]{%
8277   \glfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8278   \ifglxtrininsertinside\else##2\fi
8279   \glxtrfullsep{##1}%
8280   \glxtrparen{\glfirstabbrvscfont{\glsaccessshort{##1}}}%
8281 }%
8282 \renewcommand*{\glxtrfullplformat}[2]{%
8283   \glfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8284   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8285   \glxtrparen{\glfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8286 }%
8287 \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

8288 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8289 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8290 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8291 }%
8292 \renewcommand*{\Glsxtrfullplformat}[2]{%
8293 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8294 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8295 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8296 }%
8297 }

```

g-short-sc-desc

```

8298 \newabbreviationstyle{long-short-sc-desc}%
8299 {%
8300 \renewcommand*{\CustomAbbreviationFields}{%
8301 name={\glxtrlongshortdescname},
8302 sort={\glxtrlongshortdescsort},%
8303 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8304 \protect\glxtrfullsep{\the\glslabeltok}%
8305 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8306 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8307 \protect\glxtrfullsep{\the\glslabeltok}%
8308 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8309 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8310 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8311 }%

```

Unset the regular attribute if it has been set.

```

8312 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8313 \glshasattribute{\the\glslabeltok}{regular}%
8314 {%
8315 \glissetattribute{\the\glslabeltok}{regular}{false}%
8316 }%
8317 {}%
8318 }%
8319 }%
8320 {%

```

As long-short-sc style:

```

8321 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8322 }

```

Now the short (long) version

```

8323 \newabbreviationstyle{short-sc-long}%
8324 {%
8325 \renewcommand*{\CustomAbbreviationFields}{%
8326 name={\glxtrshortlongname},
8327 sort={\the\glsshorttok},
8328 description={\the\glslongtok},%
8329 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%

```

```

8330 \protect\glxtrfullsep{\the\glslabeltok}%
8331 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8332 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8333 \protect\glxtrfullsep{\the\glslabeltok}%
8334 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8335 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8336 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8337 \glshasattribute{\the\glslabeltok}{regular}%
8338 {%
8339 \glssetattribute{\the\glslabeltok}{regular}{false}%
8340 }%
8341 {}%
8342 }%
8343 }%
8344 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8345 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8346 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8347 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8348 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8349 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8350 \renewcommand*\glxtrfullformat}[2]{%
8351 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8352 \ifglxtrininsertinside\else##2\fi
8353 \glxtrfullsep{##1}%
8354 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8355 }%
8356 \renewcommand*\glxtrfullplformat}[2]{%
8357 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8358 \ifglxtrininsertinside\else##2\fi
8359 \glxtrfullsep{##1}%
8360 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8361 }%
8362 \renewcommand*\Glsxtrfullformat}[2]{%
8363 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8364 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8365 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8366 }%
8367 \renewcommand*\Glsxtrfullplformat}[2]{%
8368 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8369 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8370 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8371 }%
8372 }

```

As before but user provides description

```

8373 \newabbreviationstyle{short-sc-long-desc}%
8374 {%
8375   \renewcommand*{\CustomAbbreviationFields}{%
8376     name={\glxtrshortlongdescname},
8377     sort={\glxtrshortlongdescsort},
8378     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
8379     \protect\glxtrfullsep{\the\glslabeltok}}%
8380     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8381     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
8382     \protect\glxtrfullsep{\the\glslabeltok}}%
8383     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8384     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8385     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8386   }%

```

Unset the regular attribute if it has been set.

```

8387 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8388   \glsattribute{\the\glslabeltok}{regular}%
8389   {%
8390     \glssetattribute{\the\glslabeltok}{regular}{false}%
8391     }%
8392   }%
8393 }%
8394 }%
8395 {%

```

As short-sc-long style:

```

8396 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8397 }

```

short-sc

```

8398 \newabbreviationstyle{short-sc}%
8399 {%
8400   \renewcommand*{\CustomAbbreviationFields}{%
8401     name={\glxtrshortnolongname},
8402     sort={\the\glsshorttok},
8403     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8404     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8405     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8406     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8407     description={\the\glslongtok}}%
8408   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8409     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8410   }%
8411 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8412 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8413 \renewcommand*{\glsabbrvfont[1]}{\glsabbrvscfont{##1}}%
8414 \renewcommand*{\glsfirstabbrvfont[1]}{\glsfirstabbrvscfont{##1}}%

```



```

8415 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8416 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8417 \renewcommand*\glsxtrinlinefullformat}[2]{%
8418   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8419   \ifglsxtrinsertinside##2\fi}%
8420   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8421   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8422 }%
8423 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8424   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8425   \ifglsxtrinsertinside##2\fi}%
8426   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8427   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8428 }%

8429 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8430   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
8431   \ifglsxtrinsertinside##2\fi}%
8432   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8433   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8434 }%
8435 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8436   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}%
8437   \ifglsxtrinsertinside##2\fi}%
8438   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8439   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8440 }%

```

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.

```

8441 \renewcommand*\glsxtrfullformat}[2]{%
8442   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrinsertinside##2\fi}%
8443   \ifglsxtrinsertinside\else##2\fi
8444 }%
8445 \renewcommand*\glsxtrfullplformat}[2]{%
8446   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
8447   \ifglsxtrinsertinside\else##2\fi
8448 }%
8449 \renewcommand*\Glsxtrfullformat}[2]{%
8450   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrinsertinside##2\fi}%
8451   \ifglsxtrinsertinside\else##2\fi
8452 }%
8453 \renewcommand*\Glsxtrfullplformat}[2]{%
8454   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
8455   \ifglsxtrinsertinside\else##2\fi
8456 }%
8457 }

```

short-sc-nolong

```
8458 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
8459 \newabbreviationstyle{short-sc-desc}{%
8460 {%
8461   \renewcommand*{\CustomAbbreviationFields}{%
8462     name={\glxtrshortdescname},
8463     sort={\the\glsshorttok},
8464     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8465     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8466     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8467     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8468   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8469     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8470 }%
8471 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8472 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8473 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8474 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8475 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8476 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8477 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8478   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8479   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8480   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8481 }%
8482 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8483   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8484   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8485   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8486 }%
8487 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8488   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8489   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8490   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8491 }%
8492 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8493   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8494   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8495   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8496 }%
```

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.

```
8497 \renewcommand*{\glsxtrfullformat}[2]{%
```

```

8498   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8499   \ifglxtrinsertinside\else##2\fi
8500 }%
8501 \renewcommand*{\glxtrfullplformat}[2]{%
8502   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8503   \ifglxtrinsertinside\else##2\fi
8504 }%
8505 \renewcommand*{\Glsxtrfullformat}[2]{%
8506   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8507   \ifglxtrinsertinside\else##2\fi
8508 }%
8509 \renewcommand*{\Glsxtrfullplformat}[2]{%
8510   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8511   \ifglxtrinsertinside\else##2\fi
8512 }%
8513 }

```

-sc-nolong-desc

```
8514 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```

8515 \newabbreviationstyle{nolong-short-sc}%
8516 {%
8517   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8518 }%
8519 {%
8520   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8521 \renewcommand*{\glxtrinlinelinefullformat}[2]{%
8522   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8523     \ifglxtrinsertinside##2\fi}%
8524   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8525   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8526 }%
8527 \renewcommand*{\glxtrinlinelinefullplformat}[2]{%
8528   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8529     \ifglxtrinsertinside##2\fi}%
8530   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8531   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8532 }%
8533 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%
8534   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8535     \ifglxtrinsertinside##2\fi}%
8536   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8537   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8538 }%
8539 \renewcommand*{\Glsxtrinlinelinefullplformat}[2]{%
8540   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8541     \ifglxtrinsertinside##2\fi}%

```

```

8542     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8543     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8544 }%
8545 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

8546 \newabbreviationstyle{long-noshort-sc}%
8547 {%
8548   \renewcommand*{\CustomAbbreviationFields}{%
8549     name={\glxtrlongnoshortname},
8550     sort={\the\glsshorttok},
8551     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8552     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8553     text={\protect\glslongdefaultfont{\the\glslongtok}},
8554     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8555     description={\the\glslongtok}%
8556   }%
8557   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8558     \glssetAttribute{\the\glslabeltok}{regular}{true}}%
8559 }%
8560 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8561   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8562   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8563   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8564   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8565   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8566   \renewcommand*{\glxtrsubsequentfmt}[2]{%
8567     \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8568     \ifglxtrinsertinside \else##2\fi
8569   }%
8570   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8571     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8572     \ifglxtrinsertinside \else##2\fi
8573   }%
8574   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8575     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8576     \ifglxtrinsertinside \else##2\fi
8577   }%
8578   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8579     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8580     \ifglxtrinsertinside \else##2\fi
8581   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8582   \renewcommand*{\glxtrininlinefullformat}[2]{%

```

```

8583 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8584 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8585 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8586 }%
8587 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8588 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8589 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8590 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8591 }%
8592 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8593 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8594 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8595 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8596 }%
8597 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8598 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8599 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8600 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8601 }%

```

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.

```

8602 \renewcommand*{\glsxtrfullformat}[2]{%
8603 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8604 \ifglsxtrinsertinside\else##2\fi
8605 }%
8606 \renewcommand*{\glsxtrfullplformat}[2]{%
8607 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8608 \ifglsxtrinsertinside\else##2\fi
8609 }%
8610 \renewcommand*{\Glsxtrfullformat}[2]{%
8611 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8612 \ifglsxtrinsertinside\else##2\fi
8613 }%
8614 \renewcommand*{\Glsxtrfullplformat}[2]{%
8615 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8616 \ifglsxtrinsertinside\else##2\fi
8617 }%
8618 }

```

long-sc Backward compatibility:

```
8619 \@glsxtr@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.

```

8620 \newabbreviationstyle{long-noshort-sc-desc}%
8621 {%
8622 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8623 }%
8624 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8625 \renewcommand*{\abbrvpluralsuffix}{\protect\glstrscsuffix}%
8626 \renewcommand*{glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8627 \renewcommand*{glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8628 \renewcommand*{glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8629 \renewcommand*{glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8630 \renewcommand*{glsxtrsubsequentfmt}[2]{%
8631   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8632   \ifglxtrinsertinside \else##2\fi
8633 }%
8634 \renewcommand*{glsxtrsubsequentplfmt}[2]{%
8635   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8636   \ifglxtrinsertinside \else##2\fi
8637 }%
8638 \renewcommand*{Glsxtrsubsequentfmt}[2]{%
8639   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8640   \ifglxtrinsertinside \else##2\fi
8641 }%
8642 \renewcommand*{Glsxtrsubsequentplfmt}[2]{%
8643   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8644   \ifglxtrinsertinside \else##2\fi
8645 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
8646 \renewcommand*{glsxtrinelinefullformat}[2]{%
8647   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8648   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8649   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8650 }%
8651 \renewcommand*{glsxtrinelinefullplformat}[2]{%
8652   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8653   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8654   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8655 }%
8656 \renewcommand*{Glsxtrinelinefullformat}[2]{%
8657   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8658   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8659   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8660 }%
8661 \renewcommand*{Glsxtrinelinefullplformat}[2]{%
8662   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8663   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8664   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8665 }%
```

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.

```
8666 \renewcommand*{glsxtrfullformat}[2]{%
```

```

8667 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8668 \ifglxtrinsertinside\else##2\fi
8669 }%
8670 \renewcommand*{\glxtrfullplformat}[2]{%
8671 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8672 \ifglxtrinsertinside\else##2\fi
8673 }%
8674 \renewcommand*{\Glsxtrfullformat}[2]{%
8675 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8676 \ifglxtrinsertinside\else##2\fi
8677 }%
8678 \renewcommand*{\Glsxtrfullplformat}[2]{%
8679 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8680 \ifglxtrinsertinside\else##2\fi
8681 }%
8682 }

```

long-desc-sc Backward compatibility:

```
8683 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```

8684 \newabbreviationstyle{short-sc-footnote}%
8685 {%
8686 \renewcommand*{\CustomAbbreviationFields}{%
8687 name={\glxtrfootnotename},
8688 sort={\the\glsshorttok},
8689 description={\the\glslongtok},%
8690 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8691 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8692 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8693 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8694 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8695 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8696 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

8697 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8698 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8699 \glsattribute{\the\glslabeltok}{regular}%
8700 {%
8701 \glssetattribute{\the\glslabeltok}{regular}{false}%
8702 }%
8703 {}%
8704 }%
8705 }%
8706 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8707 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
```

```

8708 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8709 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8710 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8711 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8712 \renewcommand*\glsxtrfullformat}[2]{%
8713   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8714   \ifglsxtrininsertinside\else##2\fi
8715   \protect\glsxtrabbrvfootnote{##1}%
8716   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8717 }%
8718 \renewcommand*\glsxtrfullplformat}[2]{%
8719   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8720   \ifglsxtrininsertinside\else##2\fi
8721   \protect\glsxtrabbrvfootnote{##1}%
8722   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8723 }%
8724 \renewcommand*\Glsxtrfullformat}[2]{%
8725   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8726   \ifglsxtrininsertinside\else##2\fi
8727   \protect\glsxtrabbrvfootnote{##1}%
8728   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8729 }%
8730 \renewcommand*\Glsxtrfullplformat}[2]{%
8731   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8732   \ifglsxtrininsertinside\else##2\fi
8733   \protect\glsxtrabbrvfootnote{##1}%
8734   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8735 }%

```

The first use full form and the inline full form use the short (long) style.

```

8736 \renewcommand*\glsxtrinlinefullformat}[2]{%
8737   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8738   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8739   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8740 }%
8741 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8742   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8743   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8744   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8745 }%
8746 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8747   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8748   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8749   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8750 }%
8751 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8752   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8753   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```



```

8754   \glxtrparen{\glsfirstlongfootnotefont{\glaccesslongpl{##1}}}%
8755 }%
8756 }

```

footnote-sc Backward compatibility:

```
8757 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

```

8758 \newabbreviationstyle{short-sc-postfootnote}%
8759 {%
8760   \renewcommand*{\CustomAbbreviationFields}{%
8761     name={\glxtrfootnotename},
8762     sort={\the\glsshorttok},
8763     description={\the\glslongtok},%
8764     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8765     firstplural={\protect\glsfirstabbrvscfont{\the\glshortpltok}},%
8766     plural={\protect\glabbrvscfont{\the\glshortpltok}}}%

```

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.

```

8767   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8768     \csdef{glxtrpostlink\glscategorylabel}{%
8769       \glxtrifwasfirstuse
8770       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8771         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8772         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
8773       }%
8774     }%
8775   }%
8776   \glshasattribute{\the\glslabeltok}{regular}%
8777   {%
8778     \glsetattribute{\the\glslabeltok}{regular}{false}%
8779   }%
8780   {}%
8781 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8782   \renewcommand*{\glxtrsetupfulldefs}{%
8783     \let\glxtrifwasfirstuse\@secondoftwo
8784   }%
8785 }%
8786 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8787   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8788   \renewcommand*{\glabbrvfont[1]{\glabbrvscfont{##1}}}%

```

```

8789 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8790 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8791 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8792 \renewcommand*\glsxtrfullformat}[2]{%
8793   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8794   \ifglsxtrininsertinside\else##2\fi
8795 }%
8796 \renewcommand*\glsxtrfullplformat}[2]{%
8797   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8798   \ifglsxtrininsertinside\else##2\fi
8799 }%
8800 \renewcommand*\Glsxtrfullformat}[2]{%
8801   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8802   \ifglsxtrininsertinside\else##2\fi
8803 }%
8804 \renewcommand*\Glsxtrfullplformat}[2]{%
8805   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8806   \ifglsxtrininsertinside\else##2\fi
8807 }%

```

The first use full form and the inline full form use the short (long) style.

```

8808 \renewcommand*\glsxtrinlinefullformat}[2]{%
8809   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8810   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8811   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8812 }%
8813 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8814   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8815   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8816   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8817 }%
8818 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8819   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8820   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8821   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8822 }%
8823 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8824   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8825   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8826   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8827 }%
8828 }

```

postfootnote-sc Backward compatibility:

```

8829 \@glsxtr@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.

```
8830 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}
```

`\glsabbrvsmfont` Added for consistent naming.

```
8831 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}
```

`sxtrfirstsmfont` Maintained for backward compatibility.

```
8832 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}
```

`irstabbrvsmfont` Added for consistent naming.

```
8833 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}
```

and for the default short form suffix:

`\glxtrsmsuffix`

```
8834 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

`long-short-sm`

```
8835 \newabbreviationstyle{long-short-sm}%
```

```
8836 {%
```

```
8837   \renewcommand*{\CustomAbbreviationFields}{%
```

```
8838     name={\glxtrlongshortname},
```

```
8839     sort={\the\glsshorttok},
```

```
8840     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
```

```
8841     \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
8842     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
```

```
8843     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}}%
```

```
8844     \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
8845     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
```

```
8846     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
```

```
8847     description={\the\glslongtok}}%
```

```
8848   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
8849     \glsattribute{\the\glslabeltok}{regular}}%
```

```
8850     {%
```

```
8851       \glssetattribute{\the\glslabeltok}{regular}{false}}%
```

```
8852     }%
```

```
8853   }%
```

```
8854 }%
```

```
8855 }%
```

```
8856 {%
```

```
8857   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
```

```
8858   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
```

```
8859   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}}%
```

Use the default long fonts.

```
8860 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8861 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8862 \renewcommand*{\glsxtrfullformat}[2]{%
8863   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8864   \ifglsxtrinsertinside\else##2\fi
8865   \glsxtrfullsep{##1}}%
8866   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8867 }%
8868 \renewcommand*{\glsxtrfullplformat}[2]{%
8869   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8870   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8871   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8872 }%
8873 \renewcommand*{\Glsxtrfullformat}[2]{%
8874   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8875   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8876   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8877 }%
8878 \renewcommand*{\Glsxtrfullplformat}[2]{%
8879   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8880   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8881   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8882 }%
8883 }
```

g-short-sm-desc

```
8884 \newabbreviationstyle{long-short-sm-desc}%
8885 {%
8886   \renewcommand*{\CustomAbbreviationFields}{%
8887     name={\glsxtrlongshortdescname},
8888     sort={\glsxtrlongshortdescsort},%
8889     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8890       \protect\glsxtrfullsep{\the\glslabeltok}}%
8891     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8892     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8893       \protect\glsxtrfullsep{\the\glslabeltok}}%
8894     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8895     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8896     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8897   }%
```

Unset the regular attribute if it has been set.

```
8898 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8899   \glsattribute{\the\glslabeltok}{regular}%
8900   {%
8901     \glsattribute{\the\glslabeltok}{regular}{false}%
8902   }%
```

```

8903   {}%
8904 }%
8905 }%
8906 {%

```

As long-short-sm style:

```

8907 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8908 }

```

short-sm-long Now the short (long) version

```

8909 \newabbreviationstyle{short-sm-long}%
8910 {%
8911   \renewcommand*{\CustomAbbreviationFields}{%
8912     name={\glxtrshortlongname},
8913     sort={\the\glsshorttok},
8914     description={\the\gslongtok},%
8915     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}%
8916     \protect\glxtrfullsep{\the\glslabeltok}}%
8917     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongtok}}},%
8918     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}%
8919     \protect\glxtrfullsep{\the\glslabeltok}}%
8920     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongpltok}}},%
8921     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8922 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8923   \glshasattribute{\the\glslabeltok}{regular}%
8924   {%
8925     \glissetattribute{\the\glslabeltok}{regular}{false}%
8926   }%
8927   {}%
8928 }%
8929 }%
8930 {%

```

```

8931 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8932 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8933 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrrmsuffix}%
8934 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8935 \renewcommand*\gslongfont[1]{\gslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8936 \renewcommand*{\glxtrfullformat}[2]{%
8937   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8938   \ifglxtrininsertinside\else##2\fi
8939   \glxtrfullsep{##1}}%
8940   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}}%
8941 }%
8942 \renewcommand*{\glxtrfullplformat}[2]{%
8943   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8944   \ifglxtrininsertinside\else##2\fi

```

```

8945 \glxtrfullsep{##1}%
8946 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8947 }%
8948 \renewcommand*{\Glsxtrfullformat}[2]{%
8949 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8950 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8951 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8952 }%
8953 \renewcommand*{\Glsxtrfullplformat}[2]{%
8954 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8955 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8956 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8957 }%
8958 }

```

rt-sm-long-desc As before but user provides description

```

8959 \newabbreviationstyle{short-sm-long-desc}%
8960 {%
8961 \renewcommand*{\CustomAbbreviationFields}{%
8962 name={\glxtrshortlongdescname},
8963 sort={\glxtrshortlongdescsort},
8964 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8965 \protect\glxtrfullsep{\the\glslabeltok}%
8966 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8967 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8968 \protect\glxtrfullsep{\the\glslabeltok}%
8969 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8970 text={\protect\glssabbrvsmfont{\the\glsshorttok}},%
8971 plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}%
8972 }%

```

Unset the regular attribute if it has been set.

```

8973 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8974 \glshasattribute{\the\glslabeltok}{regular}%
8975 {%
8976 \glissetattribute{\the\glslabeltok}{regular}{false}%
8977 }%
8978 {}%
8979 }%
8980 }%
8981 {%

```

As short-sm-long style:

```

8982 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8983 }

```

short-sm

```

8984 \newabbreviationstyle{short-sm}%
8985 {%
8986 \renewcommand*{\CustomAbbreviationFields}{%

```

```

8987   name={\glxtrshortnolongname},
8988   sort={\the\glsshorttok},
8989   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8990   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8991   text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8992   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8993   description={\the\glslongtok}}%
8994 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8995   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8996 }%
8997 {%
8998 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8999 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9000 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrmsuffix}%
9001 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9002 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9003 \renewcommand*{\glxtrinlinelinefullformat}[2]{%
9004   \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%
9005   \ifglxtrininsertinside##2\fi}%
9006 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9007 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9008 }%
9009 \renewcommand*{\glxtrinlinelinefullplformat}[2]{%
9010   \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}%
9011   \ifglxtrininsertinside##2\fi}%
9012 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9013 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9014 }%
9015 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%
9016   \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}%
9017   \ifglxtrininsertinside##2\fi}%
9018 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9019 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9020 }%
9021 \renewcommand*{\Glsxtrinlinelinefullplformat}[2]{%
9022   \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}%
9023   \ifglxtrininsertinside##2\fi}%
9024 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9025 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9026 }%

```

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.

```

9027 \renewcommand*{\glxtrfullformat}[2]{%
9028   \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9029   \ifglxtrininsertinside\else##2\fi
9030 }%

```

```

9031 \renewcommand*\glxtrfullplformat}[2]{%
9032   \glsfirstabbrvsmfont{\glsaccessshortp{##1}\ifglxtrininsertinside##2\fi}%
9033   \ifglxtrininsertinside\else##2\fi
9034 }%
9035 \renewcommand*\Glsxtrfullformat}[2]{%
9036   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9037   \ifglxtrininsertinside\else##2\fi
9038 }%
9039 \renewcommand*\Glsxtrfullplformat}[2]{%
9040   \glsfirstabbrvsmfont{\glsaccessshortp{##1}\ifglxtrininsertinside##2\fi}%
9041   \ifglxtrininsertinside\else##2\fi
9042 }%
9043 }

```

short-sm-nolong

```

9044 \letabbreviationstyle{short-sm-nolong}{short-sm}

```

short-sm-desc

```

9045 \newabbreviationstyle{short-sm-desc}%
9046 {%
9047   \renewcommand*\CustomAbbreviationFields{%
9048     name={\glsxtrshortdescname},
9049     sort={\the\glsshorttok},
9050     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
9051     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
9052     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
9053     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
9054   \renewcommand*\GlsXtrPostNewAbbreviation{%
9055     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9056 }%
9057 {%
9058   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9059   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9060   \renewcommand*\abbrvpluralsuffix{\protect\glsxtrrmsuffix}%
9061   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9062   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9063 \renewcommand*\glxtrininlinefullformat}[2]{%
9064   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9065   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9066   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9067 }%
9068 \renewcommand*\Glsxtrininlinefullplformat}[2]{%
9069   \glsfirstabbrvsmfont{\glsaccessshortp{##1}\ifglxtrininsertinside##2\fi}%
9070   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9071   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongp{##1}}}%
9072 }%
9073 \renewcommand*\Glsxtrininlinefullformat}[2]{%
9074   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```



```

9075 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9076 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9077 }%
9078 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9079 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9080 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9081 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9082 }%

```

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.

```

9083 \renewcommand*{\glxtrfullformat}[2]{%
9084 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9085 \ifglxtrinsertinside\else##2\fi
9086 }%
9087 \renewcommand*{\glxtrfullplformat}[2]{%
9088 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9089 \ifglxtrinsertinside\else##2\fi
9090 }%
9091 \renewcommand*{\Glsxtrfullformat}[2]{%
9092 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9093 \ifglxtrinsertinside\else##2\fi
9094 }%
9095 \renewcommand*{\Glsxtrfullplformat}[2]{%
9096 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9097 \ifglxtrinsertinside\else##2\fi
9098 }%
9099 }

```

-sm-nolong-desc

```
9100 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

nolong-short-sm

```

9101 \newabbreviationstyle{nolong-short-sm}%
9102 {%
9103 \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9104 }%
9105 {%
9106 \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9107 \renewcommand*{\glxtrinlinefullformat}[2]{%
9108 \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
9109 \ifglxtrinsertinside##2\fi}%
9110 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9111 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
9112 }%
9113 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9114 \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
9115 \ifglxtrinsertinside##2\fi}%

```

```

9116 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9117 \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9118 }%
9119 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9120 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9121 \ifglxtrinsertinside##2\fi}%
9122 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9123 \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9124 }%
9125 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9126 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9127 \ifglxtrinsertinside##2\fi}%
9128 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9129 \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9130 }%
9131 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9132 \newabbreviationstyle{long-noshort-sm}%
9133 {%
9134 \renewcommand*{\CustomAbbreviationFields}{%
9135 name={\glxtrlongnoshortname},
9136 sort={\the\glsshorttok},
9137 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9138 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9139 text={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9140 plural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},%
9141 description={\the\glslongtok}%
9142 }%
9143 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9144 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9145 }%
9146 {%
9147 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9148 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9149 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9150 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9151 \renewcommand*{\glslongfont}[1]{\glsfirstlongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9152 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9153 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9154 \ifglxtrinsertinside \else##2\fi
9155 }%
9156 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9157 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9158 \ifglxtrinsertinside \else##2\fi
9159 }%
9160 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

9161 \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
9162 \ifglsxtrinsertinside \else##2\fi
9163 }%
9164 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9165 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
9166 \ifglsxtrinsertinside \else##2\fi
9167 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9168 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9169 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9170 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9171 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9172 }%
9173 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9174 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9175 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9176 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9177 }%
9178 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9179 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9180 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9181 \glsxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
9182 }%
9183 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9184 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9185 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9186 \glsxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
9187 }%

```

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.

```

9188 \renewcommand*{\glsxtrfullformat}[2]{%
9189 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9190 \ifglsxtrinsertinside\else##2\fi
9191 }%
9192 \renewcommand*{\glsxtrfullplformat}[2]{%
9193 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9194 \ifglsxtrinsertinside\else##2\fi
9195 }%
9196 \renewcommand*{\Glsxtrfullformat}[2]{%
9197 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9198 \ifglsxtrinsertinside\else##2\fi
9199 }%
9200 \renewcommand*{\Glsxtrfullplformat}[2]{%
9201 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9202 \ifglsxtrinsertinside\else##2\fi
9203 }%
9204 }

```

long-sm Backward compatibility:

```
9205 \@glsxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}
```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
9206 \newabbreviationstyle{long-noshort-sm-desc}%
```

```
9207 {%
```

```
9208 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

```
9209 }%
```

```
9210 {%
```

```
9211 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
```

```
9212 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
```

```
9213 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrrmsuffix}%
```

```
9214 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
```

```
9215 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9216 \renewcommand*\glsxtrsubsequentfmt}[2]{%
```

```
9217 \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
9218 \ifglsxtrininsertinside \else##2\fi
```

```
9219 }%
```

```
9220 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
```

```
9221 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
9222 \ifglsxtrininsertinside \else##2\fi
```

```
9223 }%
```

```
9224 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
```

```
9225 \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
9226 \ifglsxtrininsertinside \else##2\fi
```

```
9227 }%
```

```
9228 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
```

```
9229 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
9230 \ifglsxtrininsertinside \else##2\fi
```

```
9231 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9232 \renewcommand*\glsxtrinlinefullformat}[2]{%
```

```
9233 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
9234 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
9235 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
```

```
9236 }%
```

```
9237 \renewcommand*\glsxtrinlinefullplformat}[2]{%
```

```
9238 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
```

```
9239 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
9240 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
```

```
9241 }%
```

```
9242 \renewcommand*\Glsxtrinlinefullformat}[2]{%
```

```
9243 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
9244 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
9245 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
```

```
9246 }%
```

```

9247 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9248   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9249   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9250   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9251 }%

```

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.

```

9252 \renewcommand*\glsxtrfullformat}[2]{%
9253   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9254   \ifglsxtrininsertinside\else##2\fi
9255 }%
9256 \renewcommand*\glsxtrfullplformat}[2]{%
9257   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9258   \ifglsxtrininsertinside\else##2\fi
9259 }%
9260 \renewcommand*\Glsxtrfullformat}[2]{%
9261   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9262   \ifglsxtrininsertinside\else##2\fi
9263 }%
9264 \renewcommand*\Glsxtrfullplformat}[2]{%
9265   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9266   \ifglsxtrininsertinside\else##2\fi
9267 }%
9268 }

```

long-desc-sm Backward compatibility:

```

9269 \@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9270 \newabbreviationstyle{short-sm-footnote}%
9271 {%
9272   \renewcommand*\CustomAbbreviationFields{%
9273     name={\glsxtrfootnotename},
9274     sort={\the\glsshorttok},
9275     description={\the\glslongtok},%
9276     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9277       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
9278       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9279     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9280       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
9281       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9282     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.

```

9283 \renewcommand*\GlsXtrPostNewAbbreviation{%
9284   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9285   \glsattribute{\the\glslabeltok}{regular}%
9286 }%

```

```

9287     \glssetattribute{\the\glslabeltok}{regular}{false}%
9288     }%
9289     {}%
9290     }%
9291 }%
9292 {%
9293 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9294 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9295 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrmsuffix}%
9296 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9297 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9298 \renewcommand*\glsxtrfullformat}[2]{%
9299   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9300   \ifglsxtrininsertinside\else##2\fi
9301   \protect\glsxtrabbrvfootnote{##1}%
9302   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9303 }%
9304 \renewcommand*\glsxtrfullplformat}[2]{%
9305   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9306   \ifglsxtrininsertinside\else##2\fi
9307   \protect\glsxtrabbrvfootnote{##1}%
9308   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9309 }%
9310 \renewcommand*\Glsxtrfullformat}[2]{%
9311   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9312   \ifglsxtrininsertinside\else##2\fi
9313   \protect\glsxtrabbrvfootnote{##1}%
9314   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9315 }%
9316 \renewcommand*\Glsxtrfullplformat}[2]{%
9317   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9318   \ifglsxtrininsertinside\else##2\fi
9319   \protect\glsxtrabbrvfootnote{##1}%
9320   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9321 }%

```

The first use full form and the inline full form use the short (long) style.

```

9322 \renewcommand*\glsxtrinlinefullformat}[2]{%
9323   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9324   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9325   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9326 }%
9327 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9328   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9329   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9330   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9331 }%
9332 \renewcommand*\Glsxtrinlinefullformat}[2]{%

```

```

9333   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9334   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9335   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9336 }%
9337 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9338   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9339   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9340   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9341 }%
9342 }

```

footnote-sm Backward compatibility:

```

9343 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

9344 \newabbreviationstyle{short-sm-postfootnote}%
9345 {%
9346   \renewcommand*{\CustomAbbreviationFields}{%
9347     name={\glxtrfootnotename},
9348     sort={\the\glsshorttok},
9349     description={\the\glslongtok},%
9350     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9351     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9352     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.

```

9353 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9354   \csdef{glxtrpostlink\glscategorylabel}{%
9355     \glxtrifwasfirstuse
9356     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9357     \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9358     {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
9359   }%
9360   {}%
9361   }%
9362   \glshasattribute{\the\glslabeltok}{regular}%
9363   {%
9364     \glissetattribute{\the\glslabeltok}{regular}{false}%
9365     }%
9366   {}%
9367   }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9368 \renewcommand*{\glxtrsetupfulldefs}{%
9369   \let\glxtrifwasfirstuse\@secondoftwo

```

```

9370 }%
9371 }%
9372 {%
9373 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9374 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9375 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrmsuffix}%
9376 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9377 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9378 \renewcommand*\glsxtrfullformat}[2]{%
9379 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9380 \ifglsxtrininsertinside\else##2\fi
9381 }%
9382 \renewcommand*\glsxtrfullplformat}[2]{%
9383 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9384 \ifglsxtrininsertinside\else##2\fi
9385 }%
9386 \renewcommand*\Glsxtrfullformat}[2]{%
9387 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9388 \ifglsxtrininsertinside\else##2\fi
9389 }%
9390 \renewcommand*\Glsxtrfullplformat}[2]{%
9391 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9392 \ifglsxtrininsertinside\else##2\fi
9393 }%

```

The first use full form and the inline full form use the short (long) style.

```

9394 \renewcommand*\glsxtrinlinefullformat}[2]{%
9395 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9396 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9397 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9398 }%
9399 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9400 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9401 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9402 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9403 }%
9404 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9405 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9406 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9407 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9408 }%
9409 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9410 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9411 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9412 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9413 }%
9414 }

```


postfootnote-sm Backward compatibility:
 9415 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`
 9416 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

`irstabbrvemfont`
 9417 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

The default short form suffix:

`\glxtremsuffix`
 9418 \newcommand*{\glxtremsuffix}{\glsxtrabbrvpluralsuffix}

`firstlongemfont` Only used by the “long-em” styles.
 9419 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%

`\glslongemfont` Only used by the “long-em” styles.
 9420 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

`long-short-em` The long form is just set in the default long font.
 9421 \newabbreviationstyle{long-short-em}%
 9422 {%
 9423 \renewcommand*{\CustomAbbreviationFields}{%
 9424 name={\glxtrlongshortname},
 9425 sort={\the\glsshorttok},
 9426 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
 9427 \protect\glxtrfullsep{\the\glslabeltok}}%
 9428 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
 9429 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}}%
 9430 \protect\glxtrfullsep{\the\glslabeltok}}%
 9431 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
 9432 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
 9433 description={\the\glslongtok}}%
 9434 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
 9435 \glshasattribute{\the\glslabeltok}{regular}}%
 9436 {%
 9437 \glssetattribute{\the\glslabeltok}{regular}{false}}%
 9438 }%
 9439 {}%
 9440 }%
 9441 }%
 9442 {%
 9443 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
 9444 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
 9445 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}}%

Use the default long fonts.

```
9446 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9447 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9448 \renewcommand*{\glsxtrfullformat}[2]{%
9449   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinertinside##2\fi}%
9450   \ifglsxtrinertinside\else##2\fi
9451   \glsxtrfullsep{##1}}%
9452   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9453 }%
9454 \renewcommand*{\glsxtrfullplformat}[2]{%
9455   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinertinside##2\fi}%
9456   \ifglsxtrinertinside\else##2\fi\glsxtrfullsep{##1}}%
9457   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9458 }%
9459 \renewcommand*{\Glsxtrfullformat}[2]{%
9460   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinertinside##2\fi}%
9461   \ifglsxtrinertinside\else##2\fi\glsxtrfullsep{##1}}%
9462   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9463 }%
9464 \renewcommand*{\Glsxtrfullplformat}[2]{%
9465   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinertinside##2\fi}%
9466   \ifglsxtrinertinside\else##2\fi\glsxtrfullsep{##1}}%
9467   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9468 }%
9469 }
```

g-short-em-desc

```
9470 \newabbreviationstyle{long-short-em-desc}%
9471 {%
9472   \renewcommand*{\CustomAbbreviationFields}{%
9473     name={\glsxtrlongshortdescname},
9474     sort={\glsxtrlongshortdescsort},%
9475     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9476       \protect\glsxtrfullsep{\the\glslabeltok}%
9477       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9478     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9479       \protect\glsxtrfullsep{\the\glslabeltok}%
9480       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9481     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9482     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9483   }%
```

Unset the regular attribute if it has been set.

```
9484 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9485   \glsattribute{\the\glslabeltok}{regular}%
9486   {%
9487     \glssetattribute{\the\glslabeltok}{regular}{false}%
9488   }%
```

```

9489   {}%
9490   }%
9491 }%
9492 {%

```

As long-short-em style:

```

9493 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9494 }

```

ong-em-short-em

```

9495 \newabbreviationstyle{long-em-short-em}%
9496 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9497 \renewcommand*{\CustomAbbreviationFields}{%
9498   name={\glsxtrlongshortname},
9499   sort={\the\glsshorttok},
9500   first={\protect\glsfirstlongemfont{\the\glslongtok}}%
9501   \protect\glsxtrfullsep{\the\glslabeltok}}%
9502   \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9503   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}}%
9504   \protect\glsxtrfullsep{\the\glslabeltok}}%
9505   \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9506   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9507   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9508 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9509   \glsattribute{\the\glslabeltok}{regular}}%
9510   {%
9511     \glssetattribute{\the\glslabeltok}{regular}{false}}%
9512   }%
9513   {}%
9514 }%
9515 }%
9516 {%
9517 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9518 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9519 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9520 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9521 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9522 \renewcommand*{\glsxtrfullformat}[2]{%
9523   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinertinside##2\fi}%
9524   \ifglsxtrinertinside\else##2\fi
9525   \glsxtrfullsep{##1}}%
9526   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}}%
9527 }%
9528 \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

9529 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9530 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9531 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9532 }%
9533 \renewcommand*{\Glsxtrfullformat}[2]{%
9534 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9535 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9536 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9537 }%
9538 \renewcommand*{\Glsxtrfullplformat}[2]{%
9539 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9540 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9541 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9542 }%
9543 }

```

m-short-em-desc

```

9544 \newabbreviationstyle{long-em-short-em-desc}%
9545 {%
9546 \renewcommand*{\CustomAbbreviationFields}{%
9547 name={\glxtrlongshortdescname},
9548 sort={\glxtrlongshortdescsort},%
9549 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9550 \protect\glxtrfullsep{\the\glslabeltok}%
9551 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9552 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9553 \protect\glxtrfullsep{\the\glslabeltok}%
9554 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9555 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9556 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9557 }%

```

Unset the regular attribute if it has been set.

```

9558 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9559 \glsattribute{\the\glslabeltok}{regular}%
9560 {%
9561 \glssetattribute{\the\glslabeltok}{regular}{false}%
9562 }%
9563 }%
9564 }%
9565 }%
9566 {%
9567 \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9568 }

```

short-em-long Now the short (long) version

```

9569 \newabbreviationstyle{short-em-long}%
9570 {%
9571 \renewcommand*{\CustomAbbreviationFields}{%
9572 name={\glxtrshortlongname},

```

```

9573 sort={\the\glsshorttok},
9574 description={\the\glslongtok},%
9575 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9576 \protect\glxtrfullsep{\the\glslabeltok}%
9577 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9578 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9579 \protect\glxtrfullsep{\the\glslabeltok}%
9580 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9581 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9582 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9583 \glsattribute{\the\glslabeltok}{regular}%
9584 {%
9585 \glsattribute{\the\glslabeltok}{regular}{false}%
9586 }%
9587 {}%
9588 }%
9589 }%
9590 {%

```

Mostly as short-long style:

```

9591 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsuffix}%
9592 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9593 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9594 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9595 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9596 \renewcommand*\glxtrfullformat}[2]{%
9597 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9598 \ifglxtrininsertinside\else##2\fi
9599 \glxtrfullsep{##1}%
9600 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9601 }%
9602 \renewcommand*\glxtrfullplformat}[2]{%
9603 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9604 \ifglxtrininsertinside\else##2\fi
9605 \glxtrfullsep{##1}%
9606 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9607 }%
9608 \renewcommand*\Glsxtrfullformat}[2]{%
9609 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9610 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9611 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9612 }%
9613 \renewcommand*\Glsxtrfullplformat}[2]{%
9614 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9615 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9616 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9617 }%

```

9618 }

rt-em-long-desc As before but user provides description

```
9619 \newabbreviationstyle{short-em-long-desc}%
9620 {%
9621   \renewcommand*{\CustomAbbreviationFields}{%
9622     name={\glxtrshortlongdescname},
9623     sort={\glxtrshortlongdescsort},
9624     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9625       \protect\glxtrfullsep{\the\glslabeltok}%
9626       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9627     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9628       \protect\glxtrfullsep{\the\glslabeltok}%
9629       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9630     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9631     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9632   }%
```

Unset the regular attribute if it has been set.

```
9633 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9634   \glsattribute{\the\glslabeltok}{regular}%
9635   {%
9636     \glssetattribute{\the\glslabeltok}{regular}{false}%
9637   }%
9638   {}}%
9639 }%
9640 }%
9641 {%
9642   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9643 }
```

hort-em-long-em

```
9644 \newabbreviationstyle{short-em-long-em}%
9645 {%
9646   \glslongemfont is used in the description since \glsdesc doesn't set the style.
9647   \renewcommand*{\CustomAbbreviationFields}{%
9648     name={\glxtrshortlongname},
9649     sort={\the\glsshorttok},
9650     description={\protect\glslongemfont{\the\glslongtok}},%
9651     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9652       \protect\glxtrfullsep{\the\glslabeltok}%
9653       \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9654     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9655       \protect\glxtrfullsep{\the\glslabeltok}%
9656       \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9657     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9658   }%
```

Unset the regular attribute if it has been set.

```

9657 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9658   \glsattribute{\the\glslabeltok}{regular}%
9659   {%
9660     \glssetattribute{\the\glslabeltok}{regular}{false}%
9661   }%
9662   {}%
9663 }%
9664 }%
9665 {%
9666 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9667 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9668 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9669 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
9670 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9671 \renewcommand*\glsxtrfullformat[2]{%
9672   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9673   \ifglsxtrininsertinside\else##2\fi
9674   \glsxtrfullsep{##1}%
9675   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9676 }%
9677 \renewcommand*\glsxtrfullplformat[2]{%
9678   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9679   \ifglsxtrininsertinside\else##2\fi
9680   \glsxtrfullsep{##1}%
9681   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9682 }%
9683 \renewcommand*\Glsxtrfullformat[2]{%
9684   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9685   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9686   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9687 }%
9688 \renewcommand*\Glsxtrfullplformat[2]{%
9689   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9690   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9691   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9692 }%
9693 }

```

em-long-em-desc

```

9694 \newabbreviationstyle{short-em-long-em-desc}%
9695 {%
9696 \renewcommand*\CustomAbbreviationFields{%
9697   name={\glsxtrshortlongdescname},%
9698   sort={\glsxtrshortlongdescsort},%
9699   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9700   \protect\glsxtrfullsep{\the\glslabeltok}}%
9701   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9702   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%

```

```

9703     \protect\glxtrfullsep{\the\glslabelltok}%
9704     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9705     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9706     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9707 }%

```

Unset the regular attribute if it has been set.

```

9708 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9709   \glshasattribute{\the\glslabelltok}{regular}%
9710   {%
9711     \glissetattribute{\the\glslabelltok}{regular}{false}%
9712   }%
9713   {}%
9714 }%
9715 }%
9716 {%
9717   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9718 }

```

short-em

```

9719 \newabbreviationstyle{short-em}%
9720 {%
9721   \renewcommand*\CustomAbbreviationFields{%
9722     name={\glxtrshortnolongname},
9723     sort={\the\glsshorttok},
9724     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9725     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9726     text={\protect\glsabbrvemfont{\the\glsshorttok}},
9727     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9728     description={\the\glslongtok}}%
9729   \renewcommand*\GlsXtrPostNewAbbreviation}{%
9730     \glissetattribute{\the\glslabelltok}{regular}{true}}%
9731 }%
9732 {%
9733   \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
9734   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9735   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9736   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9737   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9738 \renewcommand*\glxtrinlinefullformat}[2]{%
9739   \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
9740   \ifglxtrininsertinside##2\fi}%
9741   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9742   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9743 }%
9744 \renewcommand*\glxtrinlinefullplformat}[2]{%
9745   \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}%
9746   \ifglxtrininsertinside##2\fi}%

```



```

9747 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9748 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9749 }%

9750 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9751 \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
9752 \ifglxtrinsertinside##2\fi}%
9753 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9754 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9755 }%

9756 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9757 \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
9758 \ifglxtrinsertinside##2\fi}%
9759 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9760 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9761 }%

```

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.

```

9762 \renewcommand*{\glxtrfullformat}[2]{%
9763 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9764 \ifglxtrinsertinside\else##2\fi
9765 }%

9766 \renewcommand*{\glxtrfullplformat}[2]{%
9767 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9768 \ifglxtrinsertinside\else##2\fi
9769 }%

9770 \renewcommand*{\Glsxtrfullformat}[2]{%
9771 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9772 \ifglxtrinsertinside\else##2\fi
9773 }%

9774 \renewcommand*{\Glsxtrfullplformat}[2]{%
9775 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9776 \ifglxtrinsertinside\else##2\fi
9777 }%
9778 }

```

short-em-nolong

```
9779 \letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```

9780 \newabbreviationstyle{short-em-desc}%
9781 {%
9782 \renewcommand*{\CustomAbbreviationFields}{%
9783 name={\glxtrshortdescname},
9784 sort={\the\glssshorttok},
9785 first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9786 firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9787 text={\protect\glssabbrvemfont{\the\glssshorttok}},

```

```

9788 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9789 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9790 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9791 }%
9792 {%
9793 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9794 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9795 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9796 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9797 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9798 \renewcommand*\glsxtrinlinefullformat}[2]{%
9799 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9800 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9801 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9802 }%
9803 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9804 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9805 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9806 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9807 }%
9808 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9809 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9810 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9811 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9812 }%
9813 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9814 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9815 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9816 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9817 }%

```

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.

```

9818 \renewcommand*\glsxtrfullformat}[2]{%
9819 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9820 \ifglsxtrininsertinside\else##2\fi
9821 }%
9822 \renewcommand*\glsxtrfullplformat}[2]{%
9823 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9824 \ifglsxtrininsertinside\else##2\fi
9825 }%
9826 \renewcommand*\Glsxtrfullformat}[2]{%
9827 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9828 \ifglsxtrininsertinside\else##2\fi
9829 }%
9830 \renewcommand*\Glsxtrfullplformat}[2]{%
9831 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9832 \ifglsxtrininsertinside\else##2\fi

```

```
9833 }%
9834 }
```

-em-nolong-desc

```
9835 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}
```

nolong-short-em

```
9836 \newabbreviationstyle{nolong-short-em}%
9837 {%
9838   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9839 }%
9840 {%
9841   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
9842 \renewcommand*{\glxtrinlinefullformat}[2]{%
9843   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9844     \ifglxtrinsertinside##2\fi}%
9845   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9846   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9847 }%
9848 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9849   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9850     \ifglxtrinsertinside##2\fi}%
9851   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9852   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9853 }%
9854 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9855   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9856     \ifglxtrinsertinside##2\fi}%
9857   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9858   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9859 }%
9860 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9861   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9862     \ifglxtrinsertinside##2\fi}%
9863   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9864   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9865 }%
9866 }
```

long-noshort-em The short form is explicitly invoked through commands like `\glsshort`.

```
9867 \newabbreviationstyle{long-noshort-em}%
9868 {%
9869   \renewcommand*{\CustomAbbreviationFields}{%
9870     name={\glxtrlongnoshortname},
9871     sort={\the\glsshorttok},
9872     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9873     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
```

```

9874     text={\protect\glslongdefaultfont{\the\glslongtok}},
9875     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9876     description={\the\glslongtok}%
9877 }%
9878 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9879   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9880 }%
9881 {%
9882 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9883 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9884 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9885 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9886 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9887 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9888   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9889   \ifglsxtrininsertinside \else##2\fi
9890 }%
9891 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9892   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9893   \ifglsxtrininsertinside \else##2\fi
9894 }%
9895 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9896   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9897   \ifglsxtrininsertinside \else##2\fi
9898 }%
9899 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9900   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9901   \ifglsxtrininsertinside \else##2\fi
9902 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9903 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9904   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9905   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9906   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9907 }%
9908 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9909   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9910   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9911   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9912 }%
9913 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9914   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9915   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9916   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9917 }%
9918 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9919   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

9920     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9921     \glxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9922 }%

```

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.

```

9923 \renewcommand*\glxtrfullformat}[2]{%
9924   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9925   \ifglxtrinsertinside\else##2\fi
9926 }%
9927 \renewcommand*\glxtrfullplformat}[2]{%
9928   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9929   \ifglxtrinsertinside\else##2\fi
9930 }%
9931 \renewcommand*\Glsxtrfullformat}[2]{%
9932   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9933   \ifglxtrinsertinside\else##2\fi
9934 }%
9935 \renewcommand*\Glsxtrfullplformat}[2]{%
9936   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9937   \ifglxtrinsertinside\else##2\fi
9938 }%
9939 }

```

long-em Backward compatibility:

```

9940 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like `\glssshort`.

```

9941 \newabbreviationstyle{long-em-noshort-em}%
9942 {%
9943   \renewcommand*\CustomAbbreviationFields{%
9944     name={\glxtrlongnoshortname},
9945     sort={\the\glssshorttok},
9946     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9947     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9948     text={\protect\glslongemfont{\the\glslongtok}},
9949     plural={\protect\glslongemfont{\the\glslongpltok}},%
9950     description={\protect\glslongemfont{\the\glslongtok}}%
9951   }%
9952   \renewcommand*\GlsXtrPostNewAbbreviation{%
9953     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9954 }%
9955 {%
9956   \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
9957   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9958   \renewcommand*\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9959   \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9960   \renewcommand*\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9961 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9962   \glslongemfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9963   \ifglxtrininsertinside \else##2\fi
9964 }%
9965 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9966   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9967   \ifglxtrininsertinside \else##2\fi
9968 }%
9969 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9970   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9971   \ifglxtrininsertinside \else##2\fi
9972 }%
9973 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9974   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9975   \ifglxtrininsertinside \else##2\fi
9976 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9977 \renewcommand*{\glxtrininlinefullformat}[2]{%
9978   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9979   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9980   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9981 }%
9982 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9983   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9984   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9985   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9986 }%
9987 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9988   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9989   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9990   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9991 }%
9992 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9993   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9994   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9995   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9996 }%

```

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.

```

9997 \renewcommand*{\glxtrfullformat}[2]{%
9998   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9999   \ifglxtrininsertinside\else##2\fi
10000 }%
10001 \renewcommand*{\glxtrfullplformat}[2]{%
10002   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10003   \ifglxtrininsertinside\else##2\fi
10004 }%
10005 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

10006   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10007   \ifglxtrinsertinside\else##2\fi
10008 }%
10009 \renewcommand*{\Glsxtrfullplformat}[2]{%
10010   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10011   \ifglxtrinsertinside\else##2\fi
10012 }%
10013 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

10014 \newabbreviationstyle{long-em-noshort-em-noreg}%
10015 {%
10016   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
      Unset the regular attribute if it has been set.
10017   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10018     \glshasattribute{\the\glslabeltok}{regular}%
10019     {%
10020       \glissetattribute{\the\glslabeltok}{regular}{false}%
10021     }%
10022   }%
10023 }%
10024 }%
10025 {%
10026   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
10027 }

```

`noshort-em-desc` The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

10028 \newabbreviationstyle{long-noshort-em-desc}%
10029 {%
10030   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
10031 }%
10032 {%
10033   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
10034   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10035   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10036   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10037   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
      The format for subsequent use (not used when the regular attribute is set).
10038   \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10039     \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10040     \ifglxtrinsertinside \else##2\fi
10041   }%
10042   \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10043     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10044     \ifglxtrinsertinside \else##2\fi
10045   }%
10046   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

10047 \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
10048 \ifglsxtrinsertinside \else##2\fi
10049 }%
10050 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10051 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
10052 \ifglsxtrinsertinside \else##2\fi
10053 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10054 \renewcommand*{\glsxtrinlinelinefullformat}[2]{%
10055 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10056 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10057 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10058 }%
10059 \renewcommand*{\glsxtrinlinelinefullplformat}[2]{%
10060 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10061 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10062 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10063 }%
10064 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%
10065 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10066 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10067 \glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
10068 }%
10069 \renewcommand*{\Glsxtrinlinelinefullplformat}[2]{%
10070 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10071 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10072 \glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10073 }%

```

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.

```

10074 \renewcommand*{\glsxtrfullformat}[2]{%
10075 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10076 \ifglsxtrinsertinside\else##2\fi
10077 }%
10078 \renewcommand*{\glsxtrfullplformat}[2]{%
10079 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10080 \ifglsxtrinsertinside\else##2\fi
10081 }%
10082 \renewcommand*{\Glsxtrfullformat}[2]{%
10083 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10084 \ifglsxtrinsertinside\else##2\fi
10085 }%
10086 \renewcommand*{\Glsxtrfullplformat}[2]{%
10087 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10088 \ifglsxtrinsertinside\else##2\fi
10089 }%
10090 }

```


long-desc-em Backward compatibility:

```
10091 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glxtrshort`. The long form is emphasized.

```
10092 \newabbreviationstyle{long-em-noshort-em-desc}%
10093 {%
10094   \renewcommand*{\CustomAbbreviationFields}{%
10095     name={\glxtrlongnoshortdescname},
10096     sort={\the\glslongtok},
10097     first={\protect\glsfirstlongemfont{\the\glslongtok}},
10098     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10099     text={\glslongemfont{\the\glslongtok}},
10100     plural={\glslongemfont{\the\glslongpltok}}}%
10101   }%
10102   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10103     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
10104 }%
10105 {%
10106   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
10107   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10108   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10109   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10110   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
10111 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10112   \glslongemfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10113   \ifglxtrininsertinside \else##2\fi
10114 }%
10115 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10116   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10117   \ifglxtrininsertinside \else##2\fi
10118 }%
10119 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10120   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10121   \ifglxtrininsertinside \else##2\fi
10122 }%
10123 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10124   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10125   \ifglxtrininsertinside \else##2\fi
10126 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
10127 \renewcommand*{\glxtrinlinefullformat}[2]{%
10128   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10129   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10130   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10131 }%
10132 \renewcommand*{\glxtrinlinefullplformat}[2]{%
```

```

10133 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10134 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10135 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10136 }%
10137 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10138 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10139 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10140 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10141 }%
10142 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10143 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10144 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10145 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10146 }%

```

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.

```

10147 \renewcommand*{\glxtrfullformat}[2]{%
10148 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10149 \ifglxtrinsertinside\else##2\fi
10150 }%
10151 \renewcommand*{\glxtrfullplformat}[2]{%
10152 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10153 \ifglxtrinsertinside\else##2\fi
10154 }%
10155 \renewcommand*{\Glsxtrfullformat}[2]{%
10156 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10157 \ifglxtrinsertinside\else##2\fi
10158 }%
10159 \renewcommand*{\Glsxtrfullplformat}[2]{%
10160 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10161 \ifglxtrinsertinside\else##2\fi
10162 }%
10163 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

10164 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10165 {%
10166 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

10167 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10168 \glshasattribute{\the\glslabeltok}{regular}%
10169 {%
10170 \glssetattribute{\the\glslabeltok}{regular}{false}%
10171 }%
10172 {}%
10173 }%
10174 }%
10175 {%

```

```

10176 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10177 }

```

ort-em-footnote

```

10178 \newabbreviationstyle{short-em-footnote}%
10179 {%
10180 \renewcommand*{\CustomAbbreviationFields}{%
10181   name={\glxtrfootnotename},
10182   sort={\the\glsshorttok},
10183   description={\the\gslongtok},%
10184   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10185     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10186     {\protect\glsfirstlongfootnotefont{\the\gslongtok}}},%
10187   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10188     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10189     {\protect\glsfirstlongfootnotefont{\the\gslongpltok}}},%
10190   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.

```

10191 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10192   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
10193   \glsattribute{\the\glslabeltok}{regular}%
10194   {%
10195     \glssetattribute{\the\glslabeltok}{regular}{false}%
10196   }%
10197   {}%
10198 }%
10199 }%
10200 {%
10201 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
10202 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10203 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10204 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
10205 \renewcommand*{\gslongfont}[1]{\gslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10206 \renewcommand*{\glsxtrfullformat}[2]{%
10207   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinertinside##2\fi}%
10208   \ifglxtrinertinside\else##2\fi
10209   \protect\glxtrabbrvfootnote{##1}%
10210     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10211 }%
10212 \renewcommand*{\glsxtrfullplformat}[2]{%
10213   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinertinside##2\fi}%
10214   \ifglxtrinertinside\else##2\fi
10215   \protect\glxtrabbrvfootnote{##1}%
10216     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10217 }%
10218 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

10219 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10220 \ifglxtrinsertinside\else##2\fi
10221 \protect\glxtrabbrvfootnote{##1}%
10222 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10223 }%
10224 \renewcommand*{\Glsxtrfullplformat}[2]{%
10225 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10226 \ifglxtrinsertinside\else##2\fi
10227 \protect\glxtrabbrvfootnote{##1}%
10228 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10229 }%

```

The first use full form and the inline full form use the short (long) style.

```

10230 \renewcommand*{\glsxtrinlinelinefullformat}[2]{%
10231 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10232 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10233 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10234 }%
10235 \renewcommand*{\glsxtrinlinelinefullplformat}[2]{%
10236 \glsfirstabbrvemfont{\glsaccessshorttpl{##1}\ifglxtrinsertinside##2\fi}%
10237 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10238 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10239 }%
10240 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%
10241 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10242 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10243 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10244 }%
10245 \renewcommand*{\Glsxtrinlinelinefullplformat}[2]{%
10246 \glsfirstabbrvemfont{\Glsaccessshorttpl{##1}\ifglxtrinsertinside##2\fi}%
10247 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10248 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10249 }%
10250 }

```

footnote-em Backward compatibility:

```

10251 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

10252 \newabbreviationstyle{short-em-postfootnote}%
10253 {%
10254 \renewcommand*{\CustomAbbreviationFields}{%
10255 name={\glxtrfootnotename},
10256 sort={\the\glsshorttok},
10257 description={\the\glslongtok},%
10258 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
10259 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
10260 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
10261 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10262   \csdef{glxstrpostlink\glscategorylabel}{%
10263     \glxstrifwasfirstuse
10264     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
10265       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10266       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
10267     }%
10268   {}%
10269 }%
10270 \glshasattribute{\the\glslabeltok}{regular}%
10271 {}%
10272 \glissetattribute{\the\glslabeltok}{regular}{false}%
10273 }%
10274 {}%
10275 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
10276 \renewcommand*{\glxtrsetupfulldefs}{%
10277   \let\glxstrifwasfirstuse\@secondoftwo
10278 }%
10279 }%
10280 {%
10281 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
10282 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10283 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10284 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
10285 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
10286 \renewcommand*{\glxtrfullformat}[2]{%
10287   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10288   \ifglxtrinsertinside\else##2\fi
10289 }%
10290 \renewcommand*{\glxtrfullplformat}[2]{%
10291   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10292   \ifglxtrinsertinside\else##2\fi
10293 }%
10294 \renewcommand*{\Glsxtrfullformat}[2]{%
10295   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10296   \ifglxtrinsertinside\else##2\fi
10297 }%
10298 \renewcommand*{\Glsxtrfullplformat}[2]{%
10299   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10300   \ifglxtrinsertinside\else##2\fi
```

10301 }%

The first use full form and the inline full form use the short (long) style.

```
10302 \renewcommand*\glsxtrinlinefullformat}[2]{%
10303   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10304   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10305   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10306 }%
10307 \renewcommand*\glsxtrinlinefullplformat}[2]{%
10308   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10309   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10310   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10311 }%
10312 \renewcommand*\Glsxtrinlinefullformat}[2]{%
10313   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10314   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10315   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10316 }%
10317 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
10318   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10319   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10320   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10321 }%
10322 }
```

postfootnote-em Backward compatibility:

```
10323 \@glsxtr@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:

glsxtruserfield Default is the useri field.

```
10324 \newcommand*\glsxtruserfield}{useri}
```

glsxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If `\glscurrentfieldvalue` has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```
10325 \ifdef\glscurrentfieldvalue
10326 {
10327   \newcommand*\glsxtruserparen}[2]{%
10328     \glsxtrfullsep{#2}%
10329     \glsxtrparen
10330     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}}}%
10331   }
10332 }
10333 {
10334   \newcommand*\glsxtruserparen}[2]{%
```

```

10335 \glxtrfullsep{#2}%
10336 \glxtrparen
10337   {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
10338 }
10339 }

```

Font used for short form:

lsabbrvuserfont

```
10340 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
10341 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
10342 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10343 \newcommand*{\glfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10344 \newcommand*{\glxtrusersuffix}{\glxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10345 \newcommand*{\gluserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10346 \newabbreviationstyle{long-short-user}%
```

```
10347 {%
```

```
10348 \renewcommand*{\CustomAbbreviationFields}{%
```

```
10349   name={\glxtrlongshortname},
```

```
10350   sort={\the\glsshorttok},
```

```
10351   first={\protect\glfirstlonguserfont{\the\glslongtok}}%
```

```
10352   \protect\glxtruserparen{\protect\glfirstabbrvuserfont{\the\glsshorttok}}%
```

```
10353   {\the\glslabeltok}},%
```

```
10354   firstplural={\protect\glfirstlonguserfont{\the\glslongpltok}}%
```

```
10355   \protect\glxtruserparen
```

```
10356   {\protect\glfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
```

```
10357   plural={\protect\glabbrvuserfont{\the\glsshortpltok}},%
```

```
10358   description={\protect\gluserdescription{\the\glslongtok}}%
```

```
10359   {\the\glslabeltok}}}%

```

Unset the regular attribute if it has been set.

```
10360 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10361   \glshasattribute{\the\glslabeltok}{regular}%
10362   {%
10363     \glsselattribute{\the\glslabeltok}{regular}{false}%
10364   }%
10365   {}%
10366 }%
10367 }%
10368 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10369 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10370 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10371 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10372 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10373 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
10374 \renewcommand*{\glsxtrfullformat}[2]{%
10375   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsetinside##2\fi}%
10376   \ifglsxtrinsetinside\else##2\fi
10377   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10378 }%
10379 \renewcommand*{\glsxtrfullplformat}[2]{%
10380   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsetinside##2\fi}%
10381   \ifglsxtrinsetinside\else##2\fi
10382   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10383 }%
10384 \renewcommand*{\Glsxtrfullformat}[2]{%
10385   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsetinside##2\fi}%
10386   \ifglsxtrinsetinside\else##2\fi
10387   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10388 }%
10389 \renewcommand*{\Glsxtrfullplformat}[2]{%
10390   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsetinside##2\fi}%
10391   \ifglsxtrinsetinside\else##2\fi
10392   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10393 }%
10394 }
```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```
10395 \newabbreviationstyle{long-postshort-user}%
10396 {%
10397   \renewcommand*{\CustomAbbreviationFields}{%
10398     name={\glsxtrlongshortname},
10399     sort={\the\glsshorttok},
10400     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10401     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
```



```

10402 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10403 description={\protect\glsuserdescription{\the\glslongtok}%
10404   {\the\glslabeltok}}}%
10405 \renewcommand*\GlsXtrPostNewAbbreviation}{%
10406   \csdef{glsxtrpostlink\glscategorylabel}{%
10407     \glsxtrifwasfirstuse
10408     {%
10409       \glsxtruserparen
10410       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10411       {\glslabel}%
10412     }%
10413   }%
10414 }%
10415 \glsattribute{\the\glslabeltok}{regular}%
10416 {%
10417   \glssetattribute{\the\glslabeltok}{regular}{false}%
10418 }%
10419 {}%
10420 }%
10421 }%
10422 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10423 \renewcommand*\abbrvpluralsuffix{\glsxtrusersuffix}%
10424 \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
10425 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvuserfont{##1}}%
10426 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonguserfont{##1}}%
10427 \renewcommand*\glslongfont[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10428 \renewcommand*\glsxtrfullformat[2]{%
10429   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10430   \ifglsxtrininsertinside\else##2\fi
10431 }%
10432 \renewcommand*\glsxtrfullplformat[2]{%
10433   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10434   \ifglsxtrininsertinside\else##2\fi
10435 }%
10436 \renewcommand*\Glsxtrfullformat[2]{%
10437   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10438   \ifglsxtrininsertinside\else##2\fi
10439 }%
10440 \renewcommand*\Glsxtrfullplformat[2]{%
10441   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10442   \ifglsxtrininsertinside\else##2\fi
10443 }%

```

In-line format:

```

10444 \renewcommand*\glsxtrinlinefullformat[2]{%
10445   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%

```

```

10446   \ifglxtrinsertinside\else##2\fi
10447   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10448 }%
10449 \renewcommand*{\glxtrinlinelinefullplformat}[2]{%
10450   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10451   \ifglxtrinsertinside\else##2\fi
10452   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10453 }%
10454 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%
10455   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10456   \ifglxtrinsertinside\else##2\fi
10457   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10458 }%
10459 \renewcommand*{\Glsxtrinlinelinefullplformat}[2]{%
10460   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10461   \ifglxtrinsertinside\else##2\fi
10462   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10463 }%
10464 }

```

ortuserdescname

```

10465 \newcommand*{\glxtrlongshortuserdescname}{%
10466   \protect\glslonguserfont{\the\glslongtok}%
10467   \protect\glxtruserparen
10468     {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10469 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10470 \newabbreviationstyle{long-postshort-user-desc}%
10471 {%
10472   \renewcommand*{\CustomAbbreviationFields}{%
10473     name={\glxtrlongshortuserdescname},
10474     sort={\the\glslongtok},
10475     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10476     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10477     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10478     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
10479   }%
10480   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10481     \csdef{glxtrpostlink\glscategorylabel}{%
10482       \glxtrifwasfirstuse
10483       {%
10484         \glxtruserparen
10485           {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10486           {\glslabel}%
10487       }%
10488     }%
10489   }%

```

```

10490   \glshasattribute{\the\glslabelltok}{regular}%
10491   {%
10492     \glissetattribute{\the\glslabelltok}{regular}{false}%
10493   }%
10494   {}%
10495 }%
10496 }%
10497 {%
10498   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10499 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10500 \newabbreviationstyle{short-postlong-user}%
10501 {%
10502   \renewcommand*{\CustomAbbreviationFields}{%
10503     name={\glstrshortlongname},
10504     sort={\the\glsshorttok},
10505     first={\protect\glfirstlonguserfont{\the\glslongtok}},%
10506     firstplural={\protect\glfirstlonguserfont{\the\glslongpltok}},%
10507     plural={\protect\glabbrvuserfont{\the\glsshortpltok}},%
10508     description={\protect\gluserdescription{\the\glslongtok}%
10509       {\the\glslabelltok}}}%
10510   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10511     \csdef{glstrpostlink\glscategorylabel}{%
10512       \glstrifwasfirstuse
10513       {%
10514         \glstruserparen
10515         {\glfirstlonguserfont{\glentrylong{\glslabell}}}%
10516         {\glslabell}%
10517       }%
10518     }%
10519   }%
10520   \glshasattribute{\the\glslabelltok}{regular}%
10521   {%
10522     \glissetattribute{\the\glslabelltok}{regular}{false}%
10523   }%
10524   {}%
10525 }%
10526 }%
10527 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10528 \renewcommand*{\abbrvpluralsuffix}{\glstrusersuffix}%
10529 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
10530 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvuserfont{##1}}%
10531 \renewcommand*{\glfirstlongfont}[1]{\glfirstlonguserfont{##1}}%
10532 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10533 \renewcommand*{\glxtrfullformat}[2]{%
10534   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10535   \ifglxtrininsertinside\else##2\fi
10536 }%
10537 \renewcommand*{\glxtrfullplformat}[2]{%
10538   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10539   \ifglxtrininsertinside\else##2\fi
10540 }%
10541 \renewcommand*{\Glsxtrfullformat}[2]{%
10542   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10543   \ifglxtrininsertinside\else##2\fi
10544 }%
10545 \renewcommand*{\Glsxtrfullplformat}[2]{%
10546   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10547   \ifglxtrininsertinside\else##2\fi
10548 }%

```

In-line format:

```

10549 \renewcommand*{\glxtrinlinefullformat}[2]{%
10550   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10551   \ifglxtrininsertinside\else##2\fi
10552   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10553 }%
10554 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10555   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10556   \ifglxtrininsertinside\else##2\fi
10557   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10558 }%
10559 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10560   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10561   \ifglxtrininsertinside\else##2\fi
10562   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10563 }%
10564 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10565   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10566   \ifglxtrininsertinside\else##2\fi
10567   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10568 }%
10569 }

```

onguserdesname

```

10570 \newcommand*{\glxtrshortlonguserdesname}{%
10571   \protect\glsabbrvuserfont{\the\glsshorttok}%
10572   \protect\glxtruserparen
10573     {\protect\glslonguserfont{\the\glslongpltok}}%
10574     {\the\glslabeltok}%
10575 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10576 \newabbreviationstyle{short-postlong-user-desc}%
10577 {%
10578   \renewcommand*{\CustomAbbreviationFields}{%
10579     name={\glxtrshortlonguserdescname},
10580     sort={\the\glsshorttok},
10581     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10582     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10583     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10584     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
10585   }%
10586   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10587     \csdef{glxtrpostlink\glscategorylabel}{%
10588       \glxtrifwasfirstuse
10589       {%
10590         \glxtruserparen
10591         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10592         {\glslabel}%
10593       }%
10594     }%
10595   }%
10596   \glshasattribute{\the\glslabeltok}{regular}%
10597   {%
10598     \glssetattribute{\the\glslabeltok}{regular}{false}%
10599   }%
10600   {}%
10601 }%
10602 }%
10603 {%
10604   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10605 }

```

short-user-desc

```

10606 \newabbreviationstyle{long-short-user-desc}%
10607 {%
10608   \renewcommand*{\CustomAbbreviationFields}{%
10609     name={\glxtrlongshortuserdescname},
10610     sort={\glxtrlongshortdescsort},%

10611     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10612       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
10613       {\the\glslabeltok}},%
10614     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10615       \protect\glxtruserparen
10616       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10617     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10618     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
10619   }%

```

Unset the regular attribute if it has been set.

```

10620 \renewcommand*\GlsXtrPostNewAbbreviation}{%
10621   \glshasattribute{\the\glslabelltok}{regular}%
10622   {%
10623     \glissetattribute{\the\glslabelltok}{regular}{false}%
10624   }%
10625   {}%
10626 }%
10627 }%
10628 {%
10629   \GlsXtrUseAbbrStyleFmts{long-short-user}%
10630 }

```

short-long-user

```

10631 \newabbreviationstyle{short-long-user}%
10632 {%
   \glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in
   \glsuserdescription.)
10633 \renewcommand*\CustomAbbreviationFields{%
10634   name={\glsxtrshortlongname},
10635   sort={\the\glsshorttok},
10636   description={\protect\glsuserdescription{\the\glslongtok}%
10637     {\the\glslabelltok}},%
10638   first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10639     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
10640     {\the\glslabelltok}},%
10641   firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10642     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
10643     {\the\glslabelltok}},%
10644   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10645 \renewcommand*\GlsXtrPostNewAbbreviation}{%
10646   \glshasattribute{\the\glslabelltok}{regular}%
10647   {%
10648     \glissetattribute{\the\glslabelltok}{regular}{false}%
10649   }%
10650   {}%
10651 }%
10652 }%
10653 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10654 \renewcommand*\abbrvpluralsuffix{\glsxtrusersuffix}%
10655 \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
10656 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvuserfont{##1}}%
10657 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonguserfont{##1}}%
10658 \renewcommand*\glslongfont[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10659 \renewcommand*\glxtrfullformat}[2]{%
10660   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10661   \ifglxtrinsertinside\else##2\fi
10662   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10663 }%
10664 \renewcommand*\glxtrfullplformat}[2]{%
10665   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10666   \ifglxtrinsertinside\else##2\fi
10667   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10668 }%
10669 \renewcommand*\Glsxtrfullformat}[2]{%
10670   \Glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10671   \ifglxtrinsertinside\else##2\fi
10672   \glxtruserparen{\Glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10673 }%
10674 \renewcommand*\Glsxtrfullplformat}[2]{%
10675   \Glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10676   \ifglxtrinsertinside\else##2\fi
10677   \glxtruserparen{\Glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10678 }%
10679 }

```

-long-user-desc

```

10680 \newabbreviationstyle{short-long-user-desc}%
10681 {%
10682   \renewcommand*\CustomAbbreviationFields{%
10683     name={\glxtrshortlonguserdescname},
10684     sort={\glxtrshortlongdescsort},%
10685     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10686       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
10687       {\the\glslabeltok}},%
10688     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10689       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
10690       {\the\glslabeltok}},%
10691     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10692     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
10693   }%

```

Unset the regular attribute if it has been set.

```

10694 \renewcommand*\GlsXtrPostNewAbbreviation{%
10695   \glsattribute{\the\glslabeltok}{regular}%
10696   {%
10697     \glssetattribute{\the\glslabeltok}{regular}{false}%
10698   }%
10699 }%
10700 }%
10701 }%
10702 {%

```

```
10703 \GlsXtrUseAbbrStyleFmts{short-long-user}%
10704 }
```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `keywords` 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 `\glsxtrlong` 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.

```
10705 \newrobustcmd*{\glsxtrifhyphenstart}[3]{%
10706   \ifx\glsinsert#1\relax
10707   \expandafter\@glsxtrifhyphenstart#1\relax\relax
10708   \@end@glsxtrifhyphenstart{#2}{#3}%
10709   \else
10710   \@glsxtrifhyphenstart#1\relax\relax\@end@glsxtrifhyphenstart{#2}{#3}%
10711   \fi
10712 }
```

`\trifhyphenstart`

```
10713 \def\@glsxtrifhyphenstart#1#2\@end@glsxtrifhyphenstart#3#4{%
10714   \ifx-#1\relax#3\else #4\fi
10715 }
```

`\rlonghyphenshort`

```
\glsxtrlonghyphenshort{<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.

```
10716 \newcommand*{\glsxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10717 {%
```

If `<insert>` starts with a hyphen, redefine `\glsxtrwordsep` 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 `\glsxtrwordsep` if `<insert>` doesn't start with a hyphen.

```
10718   \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{ }%
10719   \glsfirstlonghyphenfont{#2\ifglsxtrininsertinside{#4}\fi}%
10720   \ifglsxtrininsertinside\else{#4}\fi
10721   \glsxtrfullsep{#1}%
10722   \glsxtrparen{\glsfirstabbrvhyphenfont{#3\ifglsxtrininsertinside{#4}\fi}%
10723   \ifglsxtrininsertinside\else{#4}\fi}%
10724 }%
10725 }
```


abbrvhyphenfont

```
10726 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10727 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10728 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

tlonghyphenfont

```
10729 \newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10730 \newcommand*{\glsxtrhyphensuffix}{\glsxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the markwords attribute.

```
10731 \newabbreviationstyle{long-hyphen-short-hyphen}%
```

```
10732 {%
```

```
10733 \renewcommand*{\CustomAbbreviationFields}{%
```

```
10734   name={\glsxtrlongshortname},
```

```
10735   sort={\the\glsshorttok},
```

```
10736   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}}%
```

```
10737   \protect\glsxtrfullsep{\the\glslabeltok}}%
```

```
10738   \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
```

```
10739   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}}%
```

```
10740   \protect\glsxtrfullsep{\the\glslabeltok}}%
```

```
10741   \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
```

```
10742   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
```

```
10743   description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10744 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
10745   \glsasattribute{\the\glslabeltok}{regular}}%
```

```
10746   {%
```

```
10747     \glssetattribute{\the\glslabeltok}{regular}{false}}%
```

```
10748   }%
```

```
10749   }%}
```

```
10750 }%
```

```
10751 }%
```

```
10752 {%
```

```
10753 \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}}%
```

```
10754 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
```

```
10755 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
```

```
10756 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
```

```
10757 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```

10758 \renewcommand*\glxstrfullformat}[2]{%
10759   \glxstrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10760 }%
10761 \renewcommand*\glxstrfullplformat}[2]{%
10762   \glxstrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
10763   {\glsaccessshortpl{##1}}{##2}%
10764 }%
10765 \renewcommand*\Glsxstrfullformat}[2]{%
10766   \glxstrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10767 }%
10768 \renewcommand*\Glsxstrfullplformat}[2]{%
10769   \glxstrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10770   {\glsaccessshortpl{##1}}{##2}%
10771 }%
10772 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10773 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10774 {%
10775   \renewcommand*\CustomAbbreviationFields{%
10776     name={\glxstrlongshortdescname},
10777     sort={\glxstrlongshortdescsort},
10778     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}}%
10779     \protect\glxstrfullsep{\the\glslabeltok}}%
10780     \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10781     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}}%
10782     \protect\glxstrfullsep{\the\glslabeltok}}%
10783     \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10784     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
10785     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10786 }%

```

Unset the regular attribute if it has been set.

```

10787 \renewcommand*\GlsXtrPostNewAbbreviation{%
10788   \glsattribute{\the\glslabeltok}{regular}%
10789   {%
10790     \glsattribute{\the\glslabeltok}{regular}{false}%
10791   }%
10792   {}%
10793 }%
10794 }%
10795 {%
10796   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10797 }

```

onghyphennoshort `\glxstrlonghyphennoshort{<label>}{<long>}{<insert>}`

```
10798 \newcommand*{\glxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
10799 {%
```

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.

```
10800 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
```

```
10801 \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
```

```
10802 \ifglxtrininsertinside\else{#3}\fi
```

```
10803 }%
```

```
10804 }
```

`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.

```
10805 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
```

```
10806 {%
```

```
10807 \renewcommand*{\CustomAbbreviationFields}{%
```

```
10808 name={\glxtrlongnoshortdescname},
```

```
10809 sort={\expandonce\glxtrorglong},
```

```
10810 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
```

```
10811 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
```

```
10812 plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
```

```
10813 }%
```

Unset the regular attribute if it has been set.

```
10814 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
10815 \glshasattribute{\the\glslabeltok}{regular}}%
```

```
10816 {%
```

```
10817 \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
10818 }%
```

```
10819 {}%
```

```
10820 }%
```

```
10821 }%
```

```
10822 {%
```

```
10823 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10824 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
```

```
10825 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
```

```
10826 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
```

```
10827 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
```

```
10828 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
10829 \renewcommand*{\glxtrsubsequentfmt}[2]{%
```

```
10830 \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}}%
```

```
10831 }%
```

```

10832 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
10833   \glsxtrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10834 }%
10835 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
10836   \glsxtrlonghyphennohshort{##1}{\Glsaccesslong{##1}}{##2}%
10837 }%
10838 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
10839   \glsxtrlonghyphennohshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10840 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10841 \renewcommand*\glsxtrinilinefullformat}[2]{%
10842   \glsxtrlonghyphennohshort{##1}{\glsaccesslong{##1}}{##2}%
10843   \glsxtrfullsep{##1}%
10844   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10845 }%
10846 \renewcommand*\glsxtrinilinefullplformat}[2]{%
10847   \glsxtrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10848   \glsxtrfullsep{##1}%
10849   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10850 }%
10851 \renewcommand*\Glsxtrinilinefullformat}[2]{%
10852   \glsxtrlonghyphennohshort{##1}{\Glsaccesslong{##1}}{##2}%
10853   \glsxtrfullsep{##1}%
10854   \glsxtrparen{\protect\Glsfirstabbrvfont{\Glsaccessshort{##1}}}%
10855 }%
10856 \renewcommand*\Glsxtrinilinefullplformat}[2]{%
10857   \glsxtrlonghyphennohshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10858   \glsxtrfullsep{##1}%
10859   \glsxtrparen{\protect\Glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
10860 }%

```

The first use full form only displays the long form.

```

10861 \renewcommand*\glsxtrfullformat}[2]{%
10862   \glsxtrlonghyphennohshort{##1}{\glsaccesslong{##1}}{##2}%
10863 }%
10864 \renewcommand*\glsxtrfullplformat}[2]{%
10865   \glsxtrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10866 }%
10867 \renewcommand*\Glsxtrfullformat}[2]{%
10868   \glsxtrlonghyphennohshort{##1}{\Glsaccesslong{##1}}{##2}%
10869 }%
10870 \renewcommand*\Glsxtrfullplformat}[2]{%
10871   \glsxtrlonghyphennohshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10872 }%
10873 }

```

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.

```

10874 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10875 {%
10876   \renewcommand*{\CustomAbbreviationFields}{%
10877     name={\glxtrlongnoshortname},
10878     sort={\the\glssshorttok},
10879     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10880     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10881     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10882     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10883     description={\the\glslongtok}%
10884   }%

```

Unset the regular attribute if it has been set.

```

10885 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10886   \glshasattribute{\the\glslabeltok}{regular}%
10887   {%
10888     \glssetattribute{\the\glslabeltok}{regular}{false}%
10889   }%
10890   {}}%
10891 }%
10892 }%
10893 {%

10894 \GlsXtrUseAbbrStyleFmts{long-hyphen-noshort-desc-noreg}%
10895 }

```

glxtrlonghyphen

```
\glxtrlonghyphen{<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.

```
10896 \newcommand*{\glxtrlonghyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

10897   {%
10898     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
10899     \glsfirstlonghyphenfont{#1}%
10900   }%
10901 }

```

rposthyphenshort

```
\glxtrposthyphenshort{<label>}{<insert>}
```

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glxtrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```
10902 \newcommand*{\glxtrposthyphenshort}[2]{%
```

```

10903 {%
10904   \glstrifhyphenstart{#2}{\def\glstrwordsep{-}}{}%
10905   \ifglstrinsertinside{\glstrfirstlonghyphenfont{#2}}\else{#2}\fi
10906   \glstrfullsep{#1}%
10907   \glstrparen
10908   {\glstrfirstabbrvhyphenfont{\glstrshort{#1}}\ifglstrinsertinside{#2}\fi}%
10909   \ifglstrinsertinside\else{#2}\fi
10910 }%
10911 }%
10912 }

```

hyphensubsequent `\glstrposthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10913 \newcommand*{\glstrposthyphensubsequent}[2]{%
10914   \glstrfont{\ifglstrinsertinside {#2}\fi}%
10915   \ifglstrinsertinside \else{#2}\fi
10916 }

```

postshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10917 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10918 {%
10919   \renewcommand*{\CustomAbbreviationFields}{%
10920     name={\glstrlongshortname},
10921     sort={\the\glsshorttok},
10922     first={\protect\glstrfirstlonghyphenfont{\the\glslongtok}},%
10923     firstplural={\protect\glstrfirstlonghyphenfont{\the\glslongpltok}},%
10924     plural={\protect\glstrabbrvhyphenfont{\the\glsshortpltok}},%
10925     description={\protect\glstrlonghyphenfont{\the\glslongtok}}}%
10926 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10927   \csdef{glstrpostlink\glscategorylabel}{%
10928     \glstrifwasfirstuse
10929     {%
10930       \glstrposthyphenshort{\glslabel}{\glsinsert}%
10931     }%
10932   }%

```

Put the insertion into the post-link:

```

10933     \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
10934   }%
10935 }%
10936 \glshasattribute{\the\glslabeltok}{regular}%
10937 {%
10938   \glssetattribute{\the\glslabeltok}{regular}{false}%
10939 }%
10940 }%

```

```

10941 }%
10942 }%
10943 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10944 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
10945 \renewcommand*\glsabbrvfont[1]{\glsabbrvhyphenfont{##1}}%
10946 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvhyphenfont{##1}}%
10947 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
10948 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10949 \renewcommand*\glsxtrsubsequentfmt[2]{%
10950   \glsabbrvfont{\glsaccessshort{##1}}%
10951 }%
10952 \renewcommand*\glsxtrsubsequentplfmt[2]{%
10953   \glsabbrvfont{\glsaccessshortpl{##1}}%
10954 }%
10955 \renewcommand*\Glsxtrsubsequentfmt[2]{%
10956   \glsabbrvfont{\Glsaccessshort{##1}}%
10957 }%
10958 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
10959   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10960 }%

```

First use full form:

```

10961 \renewcommand*\glsxtrfullformat[2]{%
10962   \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10963 }%
10964 \renewcommand*\glsxtrfullplformat[2]{%
10965   \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10966 }%
10967 \renewcommand*\Glsxtrfullformat[2]{%
10968   \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10969 }%
10970 \renewcommand*\Glsxtrfullplformat[2]{%
10971   \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10972 }%

```

In-line format.

```

10973 \renewcommand*\glsxtrinilinefullformat[2]{%
10974   \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10975   \ifglsxtrininsertinside{##2}\fi}%
10976   \ifglsxtrininsertinside \else{##2}\fi
10977 }%
10978 \renewcommand*\glsxtrinilinefullplformat[2]{%
10979   \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10980   \ifglsxtrininsertinside{##2}\fi}%
10981   \ifglsxtrininsertinside \else{##2}\fi
10982 }%
10983 \renewcommand*\Glsxtrinilinefullformat[2]{%

```

```

10984   \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10985     \ifglxtrinsertinside{##2}\fi}%
10986   \ifglxtrinsertinside \else{##2}\fi
10987 }%
10988 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10989   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10990     \ifglxtrinsertinside{##2}\fi}%
10991   \ifglxtrinsertinside \else{##2}\fi
10992 }%
10993 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10994 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10995 {%
10996   \renewcommand*{\CustomAbbreviationFields}{%
10997     name={\glxtrlongshortdescname},
10998     sort={\glxtrlongshortdescsort},%
10999     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11000     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11001     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11002     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11003 }%
11004 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11005   \csdef{glxtrpostlink\glscategorylabel}{%
11006     \glxtrifwasfirstuse
11007     {%
11008       \glxtrposthyphenshort{\glslabel}{\glsinsert}%
11009     }%
11010     {%

```

Put the insertion into the post-link:

```

11011       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11012     }%
11013   }%
11014   \glshasattribute{\the\glslabeltok}{regular}%
11015   {%
11016     \glssetattribute{\the\glslabeltok}{regular}{false}%
11017   }%
11018   {}%
11019 }%
11020 }%
11021 {%
11022   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
11023 }

```

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.

```
11024 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
11025 {%
```

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.)

```
11026 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{)%
```

```
11027 \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
```

```
11028 \ifglxtrininsertinsideelse{#4}\fi
```

```
11029 \glxtrfullsep{#1}%
```

```
11030 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
```

```
11031 \ifglxtrininsertinsideelse{#4}\fi}%
```

```
11032 }%
```

```
11033 }
```

hen-long-hyphen Designed for use with the markwords attribute.

```
11034 \newabbreviationstyle{short-hyphen-long-hyphen}%
```

```
11035 {%
```

```
11036 \renewcommand*{\CustomAbbreviationFields}{%
```

```
11037 name={\glxtrshortlongname},
```

```
11038 sort={\the\glsshorttok},
```

```
11039 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
```

```
11040 \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
11041 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
```

```
11042 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
```

```
11043 \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
11044 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
```

```
11045 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
```

```
11046 description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
11047 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
11048 \glshasattribute{\the\glslabeltok}{regular}}%
```

```
11049 {%
```

```
11050 \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
11051 }%
```

```
11052 {}%
```

```
11053 }%
```

```
11054 }%
```

```
11055 {%
```

```
11056 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}}%
```

```
11057 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
```

```
11058 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
```

```
11059 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
```

```
11060 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```

11061 \renewcommand*\glxstrfullformat}[2]{%
11062   \glxstrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
11063 }%
11064 \renewcommand*\glxstrfullplformat}[2]{%
11065   \glxstrshorthyphenlong{##1}%
11066   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
11067 }%
11068 \renewcommand*\Glsxtrfullformat}[2]{%
11069   \glxstrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
11070 }%
11071 \renewcommand*\Glsxtrfullplformat}[2]{%
11072   \glxstrshorthyphenlong{##1}%
11073   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
11074 }%
11075 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

11076 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
11077 {%
11078   \renewcommand*\CustomAbbreviationFields{%
11079     name={\glxstrshortlongdescname},
11080     sort={\glxstrshortlongdescsort},
11081     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
11082     \protect\glxstrfullsep{\the\glslabeltok}}%
11083     \glxstrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11084     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
11085     \protect\glxstrfullsep{\the\glslabeltok}}%
11086     \glxstrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11087     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
11088     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11089 }%

```

Unset the regular attribute if it has been set.

```

11090 \renewcommand*\GlsXtrPostNewAbbreviation{%
11091   \glsattribute{\the\glslabeltok}{regular}%
11092   {%
11093     \glssetattribute{\the\glslabeltok}{regular}{false}%
11094   }%
11095   {}%
11096 }%
11097 }%
11098 {%
11099   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11100 }

```

lsxtrshorthyphen `\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.

```
11101 \newcommand*\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
11102 {%
11103   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
11104   \glsfirstabbrvhyphenfont{#1}%
11105 }%
11106 }
```

```
trposthyphenlong \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.

```
11107 \newcommand*\glxtrposthyphenlong}[2]{%
11108 {%
11109   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11110   \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11111   \glxtrfullsep{#1}%
11112   \glxtrparen
11113   {\glsfirstlonghyphenfont{\gl Sentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
11114   \ifglxtrininsertinside\else{#2}\fi
11115 }%
11116 }%
11117 }
```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
11118 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11119 {%
11120   \renewcommand*\CustomAbbreviationFields{%
11121     name={\glxtrshortlongname},
11122     sort={\the\glsshorttok},
11123     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11124     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11125     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
11126     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11127   \renewcommand*\GlsXtrPostNewAbbreviation{%
11128     \csdef{glxtrpostlink\glscategorylabel}{%
11129       \glxtrifwasfirstuse
11130       {%
11131         \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11132       }%
11133     }%
```

Put the insertion into the post-link:

```
11134     \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11135     }%
11136     }%
11137     \glsattribute{\the\glslabeltok}{regular}%
11138     {%
11139     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
11140     }%
11141     {}%
11142 }%
11143 }%
11144 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
11145 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
11146 \renewcommand*\glsabbrvfont[1]{\glsabbrvhyphenfont{##1}}%
11147 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvhyphenfont{##1}}%
11148 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
11149 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
11150 \renewcommand*\glxtrsubsequentfmt[2]{%
11151     \glsabbrvfont{\glsaccessshort{##1}}%
11152 }%
11153 \renewcommand*\glxtrsubsequentplfmt[2]{%
11154     \glsabbrvfont{\glsaccessshortpl{##1}}%
11155 }%
11156 \renewcommand*\Glsxtrsubsequentfmt[2]{%
11157     \glsabbrvfont{\Glsaccessshort{##1}}%
11158 }%
11159 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
11160     \glsabbrvfont{\Glsaccessshortpl{##1}}%
11161 }%
```

First use full form:

```
11162 \renewcommand*\glxtrfullformat[2]{%
11163     \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
11164 }%
11165 \renewcommand*\glxtrfullplformat[2]{%
11166     \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
11167 }%
11168 \renewcommand*\Glsxtrfullformat[2]{%
11169     \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
11170 }%
11171 \renewcommand*\Glsxtrfullplformat[2]{%
11172     \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
11173 }%
```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```
11174 \renewcommand*\glxtrinlinefullformat[2]{%
```

```

11175 \glsfirstabbrvhyphenfont{\glsaccessshort{##1}%
11176 \ifglsxtrinsertinside{##2}\fi}%
11177 \ifglsxtrinsertinside \else{##2}\fi
11178 }%
11179 \renewcommand*\glsxtrinlinefullplformat}[2]{%
11180 \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}%
11181 \ifglsxtrinsertinside{##2}\fi}%
11182 \ifglsxtrinsertinside \else{##2}\fi
11183 }%
11184 \renewcommand*\Glsxtrinlinefullformat}[2]{%
11185 \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
11186 \ifglsxtrinsertinside{##2}\fi}%
11187 \ifglsxtrinsertinside \else{##2}\fi
11188 }%
11189 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
11190 \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
11191 \ifglsxtrinsertinside{##2}\fi}%
11192 \ifglsxtrinsertinside \else{##2}\fi
11193 }%
11194 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

11195 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
11196 {%
11197 \renewcommand*\CustomAbbreviationFields{%
11198 name={\glsxtrshortlongdescname},
11199 sort={\glsxtrshortlongdescsort},%
11200 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11201 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11202 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11203 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11204 }%
11205 \renewcommand*\GlsXtrPostNewAbbreviation{%
11206 \csdef{glsxtrpostlink\glscategorylabel}{%
11207 \glsxtrifwasfirstuse
11208 {%
11209 \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
11210 }%
11211 }%

```

Put the insertion into the post-link:

```

11212 \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11213 }%
11214 }%
11215 \glsattribute{\the\glslabeltok}{regular}%
11216 {%
11217 \glssetattribute{\the\glslabeltok}{regular}{false}%
11218 }%
11219 {}%
11220 }%

```

```

11221 }%
11222 {%
11223   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
11224 }

```

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`

```
11225 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

`\stabbrvonlyfont`

```
11226 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

`\glslongonlyfont`

```
11227 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

`\rstlongonlyfont`

```
11228 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

`\lsxtronlysuffix`

```
11229 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

`\glsxtronlyname` The default name format for this style.

```

11230 \newcommand*{\glsxtronlyname}{%
11231   \protect\glsabbrvonlyfont{\the\glsshorttok}%
11232 }

```

`only-short-only`

```

11233 \newabbreviationstyle{long-only-short-only}%
11234 {%
11235   \renewcommand*{\CustomAbbreviationFields}{%
11236     name={\glsxtronlyname},
11237     sort={\the\glsshorttok},
11238     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11239     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11240     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
11241     description={\protect\glslongonlyfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

11242 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11243   \glsattribute{\the\glslabeltok}{regular}%
11244   {%
11245     \glssetattribute{\the\glslabeltok}{regular}{false}%
11246   }}%
11247 {}%

```

```

11248 }%
11249 }%
11250 {%
11251 \renewcommand*{\abbrvpluralsuffix}{\protect\glstronlysuffix}%
11252 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
11253 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
11254 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
11255 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%

```

The first use full form doesn't show the short form.

```

11256 \renewcommand*{\glsxtrfullformat}[2]{%
11257   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
11258   \ifglstrinsertinside\else##2\fi
11259 }%
11260 \renewcommand*{\glsxtrfullplformat}[2]{%
11261   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
11262   \ifglstrinsertinside\else##2\fi
11263 }%
11264 \renewcommand*{\Glsxtrfullformat}[2]{%
11265   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
11266   \ifglstrinsertinside\else##2\fi
11267 }%
11268 \renewcommand*{\Glsxtrfullplformat}[2]{%
11269   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
11270   \ifglstrinsertinside\else##2\fi
11271 }%

```

The inline full form does show the short form.

```

11272 \renewcommand*{\glsxtrinlinefullformat}[2]{%
11273   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
11274   \ifglstrinsertinside\else##2\fi
11275   \glsxtrfullsep{##1}%
11276   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
11277 }%
11278 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
11279   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
11280   \ifglstrinsertinside\else##2\fi
11281   \glsxtrfullsep{##1}%
11282   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
11283 }%
11284 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11285   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
11286   \ifglstrinsertinside\else##2\fi
11287   \glsxtrfullsep{##1}%
11288   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
11289 }%
11290 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11291   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
11292   \ifglstrinsertinside\else##2\fi
11293   \glsxtrfullsep{##1}%

```

```

11294   \glstrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11295   }%
11296 }

```

xtronlydescsort

```

11297 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

11298 \newcommand*{\glxtronlydescname}{%
11299   \protect\glslongfont{\the\glslongtok}}%
11300 }

```

short-only-desc

```

11301 \newabbreviationstyle{long-only-short-only-desc}%
11302 {%
11303   \renewcommand*{\CustomAbbreviationFields}{%
11304     name={\glxtronlydescname},
11305     sort={\glxtronlydescsort},%
11306     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11307     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11308     text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
11309     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
11310 }%

```

Unset the regular attribute if it has been set.

```

11311 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11312   \glshasattribute{\the\glslabeltok}{regular}}%
11313   {%
11314     \glissetattribute{\the\glslabeltok}{regular}{false}}%
11315   }%
11316   {}%
11317 }%
11318 }%
11319 {%
11320   \GlsXtrUseAbbrStyleFmts{long-only-short-only}}%
11321 }

```

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:

```
11322 \let\@glsxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
11323 \renewcommand*{\markright}[1]{%
```

```
11324 \glsxtrmarkhook
```

```
11325 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
```

```
11326 \glsxtrrestoremarkhook
```

```
11327 }
```

`\markboth` Save original definition:

```
11328 \let\@glsxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
11329 \renewcommand*{\markboth}[2]{%
```

```
11330 \glsxtrmarkhook
```

```
11331 \@glsxtr@org@markboth
```

```
11332 {\@glsxtrinmark#1\@glsxtrnotinmark}%
```

```
11333 {\@glsxtrinmark#2\@glsxtrnotinmark}%
```

```
11334 \glsxtrrestoremarkhook
```

```
11335 }
```

Also do this for `\@starttoc`

`\@starttoc` Save original definition:

```
11336 \let\@glsxtr@org@@starttoc\@starttoc
```

Redefine:

```
11337 \renewcommand*{\@starttoc}[1]{%
```

```
11338 \glsxtrmarkhook
```

```
11339 \@glsxtrinmark
```

```
11340 \@glsxtr@org@@starttoc{#1}%
```

```
11341 \@glsxtrnotinmark
```

```
11342 \glsxtrrestoremarkhook
```

```
11343 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```
11344 \newcommand*\gsxtrRevertMarks}{%
11345 \let\markright\@gsxtr@org@markright
11346 \let\markboth\@gsxtr@org@markboth
11347 \let\@starttoc\@gsxtr@org@starttoc
11348 }
```

rRevertTocMarks Just restores \@starttoc.

```
11349 \newcommand*\gsxtrRevertTocMarks}{%
11350 \let\@starttoc\@gsxtr@org@starttoc
11351 }
```

\gsxtrifinmark

```
11352 \newcommand*\gsxtrifinmark}[2]{#2}
```

\@gsxtrinmark

```
11353 \newrobustcmd*\@gsxtrinmark}{%
11354 \let\gsxtrifinmark\@firstoftwo
11355 }
```

gsxtrnotinmark

```
11356 \newrobustcmd*\@gsxtrnotinmark}{%
11357 \let\gsxtrifinmark\@secondoftwo
11358 }
```

eorpdforheading

```
11359 \ifdef\texorpdfstring
11360 {
11361 \newcommand*\gsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11362 }
11363 {
11364 \newcommand*\gsxtrtitleorpdforheading}[3]{#1}
11365 }
```

\gsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11366 \newcommand*\gsxtrmarkhook}{%
Save current definitions:
11367 \let\@gsxtr@org@MakeUppercase\MakeUppercase
11368 \let\@gsxtr@org@gsxtrtitleorpdforheading\gsxtrtitleorpdforheading
11369 \let\@gsxtr@org@gsxtrtitleshort\gsxtrtitleshort
11370 \let\@gsxtr@org@gsxtrtitleshortpl\gsxtrtitleshortpl
11371 \let\@gsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11372 \let\@gsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11373 \let\@gsxtr@org@gsxtrtitlename\gsxtrtitlename
11374 \let\@gsxtr@org@Glsxtrtitlename\Glsxtrtitlename
```

```

11375 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11376 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11377 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11378 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11379 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11380 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11381 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11382 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11383 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11384 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11385 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11386 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11387 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11388 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11389 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11390 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

11391 \let\glsxtrifinmark\@firstoftwo
11392 \let\MakeUppercase\MakeTextUppercase
11393 \let\glsxtrtitleorpdforheading\@thirdofthree
11394 \let\glsxtrtitleshort\glsxtrheadshort
11395 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11396 \let\Glsxtrtitleshort\Glsxtrheadshort
11397 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11398 \let\glsxtrtitlename\glsxtrheadname
11399 \let\Glsxtrtitlename\Glsxtrheadname
11400 \let\glsxtrtitletext\glsxtrheadtext
11401 \let\Glsxtrtitletext\Glsxtrheadtext
11402 \let\glsxtrtitleplural\glsxtrheadplural
11403 \let\Glsxtrtitleplural\Glsxtrheadplural
11404 \let\glsxtrtitlefirst\glsxtrheadfirst
11405 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11406 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11407 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11408 \let\glsxtrtitlelong\glsxtrheadlong
11409 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11410 \let\Glsxtrtitlelong\Glsxtrheadlong
11411 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11412 \let\glsxtrtitlefull\glsxtrheadfull
11413 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11414 \let\Glsxtrtitlefull\Glsxtrheadfull
11415 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11416 }

```

`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.)

```

11417 \newcommand*{\glxtrrestoremarkhook}{%
11418   \let\glxtrifinmark\@secondoftwo
11419   \let\MakeUppercase\@glxtr@org@MakeUppercase
11420   \let\glxtrtitleorpdforheading\@glxtr@org@glxtrtitleorpdforheading
11421   \let\glxtrtitleshort\@glxtr@org@glxtrtitleshort
11422   \let\glxtrtitleshortpl\@glxtr@org@glxtrtitleshortpl
11423   \let\Glsxtrtitleshort\@glxtr@org@Glsxtrtitleshort
11424   \let\Glsxtrtitleshortpl\@glxtr@org@Glsxtrtitleshortpl
11425   \let\glxtrtitlename\@glxtr@org@glxtrtitlename
11426   \let\Glsxtrtitlename\@glxtr@org@Glsxtrtitlename
11427   \let\glxtrtitletext\@glxtr@org@glxtrtitletext
11428   \let\Glsxtrtitletext\@glxtr@org@Glsxtrtitletext
11429   \let\glxtrtitleplural\@glxtr@org@glxtrtitleplural
11430   \let\Glsxtrtitleplural\@glxtr@org@Glsxtrtitleplural
11431   \let\glxtrtitlefirst\@glxtr@org@glxtrtitlefirst
11432   \let\Glsxtrtitlefirst\@glxtr@org@Glsxtrtitlefirst
11433   \let\glxtrtitlefirstplural\@glxtr@org@glxtrtitlefirstplural
11434   \let\Glsxtrtitlefirstplural\@glxtr@org@Glsxtrtitlefirstplural
11435   \let\glxtrtitlelong\@glxtr@org@glxtrtitlelong
11436   \let\glxtrtitlelongpl\@glxtr@org@glxtrtitlelongpl
11437   \let\Glsxtrtitlelong\@glxtr@org@Glsxtrtitlelong
11438   \let\Glsxtrtitlelongpl\@glxtr@org@Glsxtrtitlelongpl
11439   \let\glxtrtitlefull\@glxtr@org@glxtrtitlefull
11440   \let\glxtrtitlefullpl\@glxtr@org@glxtrtitlefullpl
11441   \let\Glsxtrtitlefull\@glxtr@org@Glsxtrtitlefull
11442   \let\Glsxtrtitlefullpl\@glxtr@org@Glsxtrtitlefullpl
11443 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glxtrheadshort Command used to display short form in the page header.

```

11444 \newcommand*{\glxtrheadshort}[1]{%
11445   \protect\NoCaseChange
11446   {%
11447     \glusifattribute{#1}{headuc}{true}%
11448     {%
11449       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11450     }%
11451     {%
11452       \glxtrshort[noindex,hyper=false]{#1}[]%
11453     }%
11454   }%
11455 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11456 \newrobustcmd*{\glxtrtitleshort}[1]{%
11457   \glxtrshort[noindex,hyper=false]{#1}[]%
11458 }

```

`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.

```
11459 \newcommand*{\glxtrheadshortpl}[1]{%
11460   \protect\NoCaseChange
11461   {%
11462     \glsifattribute{#1}{headuc}{true}%
11463     {%
11464       \GLSxtrshortpl [noindex,hyper=false] {#1} []%
11465     }%
11466     {%
11467       \glxtrshortpl [noindex,hyper=false] {#1} []%
11468     }%
11469   }%
11470 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11471 \newrobustcmd*{\glxtrtitleshortpl}[1]{%
11472   \glxtrshortpl [noindex,hyper=false] {#1} []%
11473 }
```

`GLsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11474 \newcommand*{\GLsxtrheadshort}[1]{%
11475   \protect\NoCaseChange
11476   {%
11477     \glsifattribute{#1}{headuc}{true}%
11478     {%
11479       \GLSxtrshort [noindex,hyper=false] {#1} []%
11480     }%
11481     {%
11482       \GLsxtrshort [noindex,hyper=false] {#1} []%
11483     }%
11484   }%
11485 }
```

`lSxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11486 \newrobustcmd*{\GLsxtrtitleshort}[1]{%
11487   \GLsxtrshort [noindex,hyper=false] {#1} []%
11488 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11489 \newcommand*{\GLsxtrheadshortpl}[1]{%
11490   \protect\NoCaseChange
11491   {%
11492     \glsifattribute{#1}{headuc}{true}%
```

```

11493  {%
11494    \GLSxtrshortpl [noindex,hyper=false] {#1} []%
11495  }%
11496  {%
11497    \GLSxtrshortpl [noindex,hyper=false] {#1} []%
11498  }%
11499 }%
11500 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11501 \newrobustcmd*{\GLSxtrtitleshortpl}[1]{%
11502   \GLSxtrshortpl [noindex,hyper=false] {#1} []%
11503 }

```

`\glxtrheadname` As above but for the name value.

```

11504 \newcommand*{\glxtrheadname}[1]{%
11505   \protect\NoCaseChange
11506   {%
11507     \gl@ifattribute{#1}{headuc}{true}%
11508     {%
11509       \GLSname [noindex,hyper=false] {#1} []%
11510     }%
11511     {%
11512       \gl$name [noindex,hyper=false] {#1} []%
11513     }%
11514   }%
11515 }

```

`glxtrtitlename` Command to display name value in section title and table of contents.

```

11516 \newrobustcmd*{\glxtrtitlename}[1]{%
11517   \gl$name [noindex,hyper=false] {#1} []%
11518 }

```

`\GLSxtrheadname` First letter converted to upper case

```

11519 \newcommand*{\GLSxtrheadname}[1]{%
11520   \protect\NoCaseChange
11521   {%
11522     \gl@ifattribute{#1}{headuc}{true}%
11523     {%
11524       \GLSname [noindex,hyper=false] {#1} []%
11525     }%
11526     {%
11527       \GL$name [noindex,hyper=false] {#1} []%
11528     }%
11529   }%
11530 }

```

`Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```
11531 %\changes{1.21}{2017-11-03}{new}
11532 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11533   \Glsname[noindex,hyper=false]{#1}[]%
11534 }
```

`\glsxtrheadtext` As above but for the text value.

```
11535 \newcommand*{\glsxtrheadtext}[1]{%
11536   \protect\NoCaseChange
11537   {%
11538     \glsifattribute{#1}{headuc}{true}%
11539     {%
11540       \GLStext[noindex,hyper=false]{#1}[]%
11541     }%
11542     {%
11543       \glstext[noindex,hyper=false]{#1}[]%
11544     }%
11545   }%
11546 }
```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
11547 \newrobustcmd*{\glsxtrtitletext}[1]{%
11548   \glstext[noindex,hyper=false]{#1}[]%
11549 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
11550 \newcommand*{\Glsxtrheadtext}[1]{%
11551   \protect\NoCaseChange
11552   {%
11553     \glsifattribute{#1}{headuc}{true}%
11554     {%
11555       \GLStext[noindex,hyper=false]{#1}[]%
11556     }%
11557     {%
11558       \Glstext[noindex,hyper=false]{#1}[]%
11559     }%
11560   }%
11561 }
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
11562 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11563   \Glstext[noindex,hyper=false]{#1}[]%
11564 }
```

`lgsxtrheadplural` As above but for the plural value.

```
11565 \newcommand*{\lgsxtrheadplural}[1]{%
```

```

11566 \protect\NoCaseChange
11567 {%
11568   \glsifattribute{#1}{headuc}{true}%
11569   {%
11570     \GLSplural[noindex,hyper=false]{#1}[]%
11571   }%
11572   {%
11573     \glsplural[noindex,hyper=false]{#1}[]%
11574   }%
11575 }%
11576 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```

11577 \newrobustcmd*{\glsxtrtitleplural}[1]{%
11578   \glsplural[noindex,hyper=false]{#1}[]%
11579 }

```

`lsxtrheadplural` Convert first letter to upper case.

```

11580 \newcommand*{\Glsxtrheadplural}[1]{%
11581   \protect\NoCaseChange
11582   {%
11583     \glsifattribute{#1}{headuc}{true}%
11584     {%
11585       \GLSplural[noindex,hyper=false]{#1}[]%
11586     }%
11587     {%
11588       \Glsplural[noindex,hyper=false]{#1}[]%
11589     }%
11590   }%
11591 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

11592 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11593   \Glsplural[noindex,hyper=false]{#1}[]%
11594 }

```

`glsxtrheadfirst` As above but for the first value.

```

11595 \newcommand*{\glsxtrheadfirst}[1]{%
11596   \protect\NoCaseChange
11597   {%
11598     \glsifattribute{#1}{headuc}{true}%
11599     {%
11600       \GLSfirst[noindex,hyper=false]{#1}[]%
11601     }%
11602     {%
11603       \glsfirst[noindex,hyper=false]{#1}[]%
11604     }%
11605   }%

```


11606 }

`lxsxtrtitlefirst` Command to display first value in section title and table of contents.

```
11607 \newrobustcmd*{\glxsxtrtitlefirst}[1]{%
11608   \glsfirst[noindex,hyper=false]{#1}[]%
11609 }
```

`Glsxtrheadfirst` First letter converted to upper case

```
11610 \newcommand*{\Glsxtrheadfirst}[1]{%
11611   \protect\NoCaseChange
11612   {%
11613     \glsifattribute{#1}{headuc}{true}%
11614     {%
11615       \GLSfirst[noindex,hyper=false]{#1}[]%
11616     }%
11617     {%
11618       \Glsfirst[noindex,hyper=false]{#1}[]%
11619     }%
11620   }%
11621 }
```

`lxsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11622 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11623   \Glsfirst[noindex,hyper=false]{#1}[]%
11624 }
```

`headfirstplural` As above but for the firstplural value.

```
11625 \newcommand*{\glxsxtrheadfirstplural}[1]{%
11626   \protect\NoCaseChange
11627   {%
11628     \glsifattribute{#1}{headuc}{true}%
11629     {%
11630       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11631     }%
11632     {%
11633       \glsfirstplural[noindex,hyper=false]{#1}[]%
11634     }%
11635   }%
11636 }
```

`itlefirstplural` Command to display firstplural value in section title and table of contents.

```
11637 \newrobustcmd*{\glxsxtritlefirstplural}[1]{%
11638   \glsfirstplural[noindex,hyper=false]{#1}[]%
11639 }
```

`headfirstplural` First letter converted to upper case

```
11640 \newcommand*{\Glsxtrheadfirstplural}[1]{%
```

```

11641 \protect\NoCaseChange
11642 {%
11643   \glsifattribute{#1}{headuc}{true}%
11644   {%
11645     \GLSfirstplural[noindex,hyper=false]{#1}[]%
11646   }%
11647   {%
11648     \GLSfirstplural[noindex,hyper=false]{#1}[]%
11649   }%
11650 }%
11651 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11652 \newrobustcmd*{\GLSxtrtitlefirstplural}[1]{%
11653   \GLSfirstplural[noindex,hyper=false]{#1}[]%
11654 }

```

`\glsxtrheadlong` Command used to display long form in the page header.

```

11655 \newcommand*{\glsxtrheadlong}[1]{%
11656   \protect\NoCaseChange
11657   {%
11658     \glsifattribute{#1}{headuc}{true}%
11659     {%
11660       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11661     }%
11662     {%
11663       \glsxtrlong[noindex,hyper=false]{#1}[]%
11664     }%
11665   }%
11666 }

```

`\glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

11667 \newrobustcmd*{\glsxtrtitlelong}[1]{%
11668   \glsxtrlong[noindex,hyper=false]{#1}[]%
11669 }

```

`\glsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

11670 \newcommand*{\glsxtrheadlongpl}[1]{%
11671   \protect\NoCaseChange
11672   {%
11673     \glsifattribute{#1}{headuc}{true}%
11674     {%
11675       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11676     }%
11677     {%

```

```

11678     \glsxtrlongpl [noindex,hyper=false] {#1} []%
11679   }%
11680 }%
11681 }

```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

11682 \newrobustcmd*{\gsxtrtitlelongpl}[1]{%
11683   \glsxtrlongpl [noindex,hyper=false] {#1} []%
11684 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

11685 \newcommand*{\Glsxtrheadlong}[1]{%
11686   \protect\NoCaseChange
11687   {%
11688     \glsifattribute{#1}{headuc}{true}%
11689     {%
11690       \GLSxtrlong [noindex,hyper=false] {#1} []%
11691     }%
11692     {%
11693       \Glsxtrlong [noindex,hyper=false] {#1} []%
11694     }%
11695   }%
11696 }

```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11697 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11698   \Glsxtrlong [noindex,hyper=false] {#1} []%
11699 }

```

`lxsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```

11700 \newcommand*{\Glsxtrheadlongpl}[1]{%
11701   \protect\NoCaseChange
11702   {%
11703     \glsifattribute{#1}{headuc}{true}%
11704     {%
11705       \GLSxtrlongpl [noindex,hyper=false] {#1} []%
11706     }%
11707     {%
11708       \Glsxtrlongpl [noindex,hyper=false] {#1} []%
11709     }%
11710   }%
11711 }

```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11712 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11713   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11714 }

```

`\glsxtrheadfull` Command used to display full form in the page header.

```

11715 \newcommand*{\glsxtrheadfull}[1]{%
11716   \protect\NoCaseChange
11717   {%
11718     \glsifattribute{#1}{headuc}{true}%
11719     {%
11720       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11721     }%
11722     {%
11723       \glsxtrfull[noindex,hyper=false]{#1}[]%
11724     }%
11725   }%
11726 }

```

`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```

11727 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11728   \glsxtrfull[noindex,hyper=false]{#1}[]%
11729 }

```

`glsxtrheadfullpl` 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.

```

11730 \newcommand*{\glsxtrheadfullpl}[1]{%
11731   \protect\NoCaseChange
11732   {%
11733     \glsifattribute{#1}{headuc}{true}%
11734     {%
11735       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11736     }%
11737     {%
11738       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11739     }%
11740   }%
11741 }

```

`glsxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```

11742 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
11743   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11744 }

```

`\GLSxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```

11745 \newcommand*{\GLSxtrheadfull}[1]{%
11746   \protect\NoCaseChange

```

```

11747 {%
11748   \glsifattribute{#1}{headuc}{true}%
11749   {%
11750     \GLSxtrfull[noindex,hyper=false]{#1}[]%
11751   }%
11752   {%
11753     \Glsxtrfull[noindex,hyper=false]{#1}[]%
11754   }%
11755 }%
11756 }

```

`\Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11757 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11758   \Glsxtrfull[noindex,hyper=false]{#1}[]%
11759 }

```

`\Glsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11760 \newcommand*{\Glsxtrheadfullpl}[1]{%
11761   \protect\NoCaseChange
11762   {%
11763     \glsifattribute{#1}{headuc}{true}%
11764     {%
11765       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11766     }%
11767     {%
11768       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11769     }%
11770   }%
11771 }

```

`\Glsxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11772 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11773   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11774 }

```

`\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.

```

11775 \ifdef\texorpdfstring
11776 {
11777   \newcommand*{\glsfmtshort}[1]{%
11778     \texorpdfstring
11779     {\glsxtrtitleshort{#1}}%
11780     {\glsentryshort{#1}}%
11781   }
11782 }

```

```

11783 {
11784 \newcommand*\glsfmtshort}[1]{%
11785 \glsxtrtitleshort{#1}}
11786 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11787 \ifdef\teorpdfstring
11788 {
11789 \newcommand*\glsfmtshortpl}[1]{%
11790 \teorpdfstring
11791 {\glsxtrtitleshortpl{#1}}%
11792 {\glsentryshortpl{#1}}%
11793 }
11794 }
11795 {
11796 \newcommand*\glsfmtshortpl}[1]{%
11797 \glsxtrtitleshortpl{#1}}
11798 }

```

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).

```

11799 \ifdef\teorpdfstring
11800 {
11801 \newcommand*\Glsfmtshort}[1]{%
11802 \teorpdfstring
11803 {\Glsxtrtitleshort{#1}}%
11804 {\glsentryshort{#1}}%
11805 }
11806 }
11807 {
11808 \newcommand*\Glsfmtshort}[1]{%
11809 \Glsxtrtitleshort{#1}}
11810 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11811 \ifdef\teorpdfstring
11812 {
11813 \newcommand*\Glsfmtshortpl}[1]{%
11814 \teorpdfstring
11815 {\Glsxtrtitleshortpl{#1}}%
11816 {\glsentryshortpl{#1}}%
11817 }
11818 }
11819 {
11820 \newcommand*\Glsfmtshortpl}[1]{%
11821 \Glsxtrtitleshortpl{#1}}

```

11822 }

`\glsfmtname` As above but for the name value.

```
11823 \ifdef\textorpdfstring
11824 {
11825   \newcommand*\glsfmtname[1]{%
11826     \textorpdfstring
11827     {\glsxtrtitlename{#1}}%
11828     {\glsentryname{#1}}%
11829   }
11830 }
11831 {
11832   \newcommand*\glsfmtname[1]{%
11833     \glsxtrtitlename{#1}}
11834 }
```

`\Glsfmtname` First letter converted to upper case.

```
11835 \ifdef\textorpdfstring
11836 {
11837   \newcommand*\Glsfmtname[1]{%
11838     \textorpdfstring
11839     {\Glsxtrtitlename{#1}}%
11840     {\glsentryname{#1}}%
11841   }
11842 }
11843 {
11844   \newcommand*\Glsfmtname[1]{%
11845     \Glsxtrtitlename{#1}}
11846 }
```

`\glsfmttext` As above but for the text value.

```
11847 \ifdef\textorpdfstring
11848 {
11849   \newcommand*\glsfmttext[1]{%
11850     \textorpdfstring
11851     {\glsxtrtitletext{#1}}%
11852     {\glsentrytext{#1}}%
11853   }
11854 }
11855 {
11856   \newcommand*\glsfmttext[1]{%
11857     \glsxtrtitletext{#1}}
11858 }
```

`\Glsfmttext` First letter converted to upper case.

```
11859 \ifdef\textorpdfstring
11860 {
11861   \newcommand*\Glsfmttext[1]{%
11862     \textorpdfstring
```

```

11863   {\Glsxtrtitletext{#1}}%
11864   {\glsentrytext{#1}}%
11865 }
11866 }
11867 {
11868   \newcommand*{\Glsfmttext}[1]{%
11869     \Glsxtrtitletext{#1}}
11870 }

```

`\glsfmtplural` As above but for the plural value.

```

11871 \ifdef\texorpdfstring
11872 {
11873   \newcommand*{\glsfmtplural}[1]{%
11874     \texorpdfstring
11875     {\glsxtrtitleplural{#1}}%
11876     {\glsentryplural{#1}}%
11877   }
11878 }
11879 {
11880   \newcommand*{\glsfmtplural}[1]{%
11881     \glsxtrtitleplural{#1}}
11882 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11883 \ifdef\texorpdfstring
11884 {
11885   \newcommand*{\Glsfmtplural}[1]{%
11886     \texorpdfstring
11887     {\Glsxtrtitleplural{#1}}%
11888     {\glsentryplural{#1}}%
11889   }
11890 }
11891 {
11892   \newcommand*{\Glsfmtplural}[1]{%
11893     \Glsxtrtitleplural{#1}}
11894 }

```

`\glsfmtfirst` As above but for the first value.

```

11895 \ifdef\texorpdfstring
11896 {
11897   \newcommand*{\glsfmtfirst}[1]{%
11898     \texorpdfstring
11899     {\glsxtrtitlefirst{#1}}%
11900     {\glsentryfirst{#1}}%
11901   }
11902 }
11903 {
11904   \newcommand*{\glsfmtfirst}[1]{%
11905     \glsxtrtitlefirst{#1}}

```


11906 }

`\Glsfmtfirst` First letter converted to upper case.

```
11907 \ifdef\texorpdfstring
11908 {
11909   \newcommand*\Glsfmtfirst}[1]{%
11910     \texorpdfstring
11911     {\Glsxtrtitlefirst{#1}}%
11912     {\glsentryfirst{#1}}%
11913   }
11914 }
11915 {
11916   \newcommand*\Glsfmtfirst}[1]{%
11917     \Glsxtrtitlefirst{#1}}
11918 }
```

`\glsfmtfirstpl` As above but for the firstplural value.

```
11919 \ifdef\texorpdfstring
11920 {
11921   \newcommand*\glsfmtfirstpl}[1]{%
11922     \texorpdfstring
11923     {\glsxtrtitlefirstplural{#1}}%
11924     {\glsentryfirstplural{#1}}%
11925   }
11926 }
11927 {
11928   \newcommand*\glsfmtfirstpl}[1]{%
11929     \glsxtrtitlefirstplural{#1}}
11930 }
```

`\Glsfmtfirstpl` First letter converted to upper case.

```
11931 \ifdef\texorpdfstring
11932 {
11933   \newcommand*\Glsfmtfirstpl}[1]{%
11934     \texorpdfstring
11935     {\Glsxtrtitlefirstplural{#1}}%
11936     {\glsentryfirstplural{#1}}%
11937   }
11938 }
11939 {
11940   \newcommand*\Glsfmtfirstpl}[1]{%
11941     \Glsxtrtitlefirstplural{#1}}
11942 }
```

`\glsfmtlong` As above but for the long value.

```
11943 \ifdef\texorpdfstring
11944 {
11945   \newcommand*\glsfmtlong}[1]{%
11946     \texorpdfstring
```

```

11947   {\glsxtrtitlelong{#1}}%
11948   {\glsentrylong{#1}}%
11949   }
11950 }
11951 {
11952   \newcommand*{\glsfmtlong}[1]{%
11953     \glsxtrtitlelong{#1}}
11954 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11955 \ifdef\textorpdfstring
11956 {
11957   \newcommand*{\Glsfmtlong}[1]{%
11958     \textorpdfstring
11959     {\Glsxtrtitlelong{#1}}%
11960     {\glsentrylong{#1}}%
11961   }
11962 }
11963 {
11964   \newcommand*{\Glsfmtlong}[1]{%
11965     \Glsxtrtitlelong{#1}}
11966 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11967 \ifdef\textorpdfstring
11968 {
11969   \newcommand*{\glsfmtlongpl}[1]{%
11970     \textorpdfstring
11971     {\glsxtrtitlelongpl{#1}}%
11972     {\glsentrylongpl{#1}}%
11973   }
11974 }
11975 {
11976   \newcommand*{\glsfmtlongpl}[1]{%
11977     \glsxtrtitlelongpl{#1}}
11978 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11979 \ifdef\textorpdfstring
11980 {
11981   \newcommand*{\Glsfmtlongpl}[1]{%
11982     \textorpdfstring
11983     {\Glsxtrtitlelongpl{#1}}%
11984     {\glsentrylongpl{#1}}%
11985   }
11986 }
11987 {
11988   \newcommand*{\Glsfmtlongpl}[1]{%
11989     \Glsxtrtitlelongpl{#1}}

```

11990 }

`\glsfmtfull` In-line full format.

```
11991 \ifdef\textorpdfstring
11992 {
11993   \newcommand*\glsfmtfull}[1]{%
11994     \textorpdfstring
11995     {\glsxtrtitlefull{#1}}%
11996     {\glsxtrinlinefullformat{#1}{}}%
11997   }
11998 }
11999 {
12000   \newcommand*\glsfmtfull}[1]{%
12001     \glsxtrtitlefull{#1}}
12002 }
```

`\Glsfmtfull` First letter converted to upper case.

```
12003 \ifdef\textorpdfstring
12004 {
12005   \newcommand*\Glsfmtfull}[1]{%
12006     \textorpdfstring
12007     {\Glsxtrtitlefull{#1}}%
12008     {\Glsxtrinlinefullformat{#1}{}}%
12009   }
12010 }
12011 {
12012   \newcommand*\Glsfmtfull}[1]{%
12013     \Glsxtrtitlefull{#1}}
12014 }
```

`\glsfmtfullpl` In-line full plural format.

```
12015 \ifdef\textorpdfstring
12016 {
12017   \newcommand*\glsfmtfullpl}[1]{%
12018     \textorpdfstring
12019     {\glsxtrtitlefullpl{#1}}%
12020     {\glsxtrinlinefullplformat{#1}{}}%
12021   }
12022 }
12023 {
12024   \newcommand*\glsfmtfullpl}[1]{%
12025     \glsxtrtitlefullpl{#1}}
12026 }
```

`\Glsfmtfullpl` First letter converted to upper case.

```
12027 \ifdef\textorpdfstring
12028 {
12029   \newcommand*\Glsfmtfullpl}[1]{%
12030     \textorpdfstring
```

```

12031   {\Glsxtrtitlefullpl{#1}}%
12032   {\Glsxtrinlinelinefullplformat{#1}{}}%
12033 }
12034 }
12035 {
12036   \newcommand*{\Glsfmtfullpl}[1]{%
12037     \Glsxtrtitlefullpl{#1}}
12038 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

12039 \newcommand*{\RequireGlossariesExtraLang}[1]{%
12040   \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
12041 }

```

`sariesExtraLang`

```

12042 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
12043   \ProvidesFile{glossariesxtr-#1.ldf}%
12044 }

```

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.

```

12045 \newcommand{\glsxtr@loaddialect}{%
12046   \IfTrackedLanguageFileExists{\this@dialect}%
12047   {glossariesxtr-}% prefix
12048   {.ldf}%
12049   {%
12050     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
12051   }%
12052   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glsxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

12053   \@glsxtrdialecthook
12054 }

12055 \@ifpackageloaded{tracklang}
12056 {%
12057   \AnyTrackedLanguages
12058   {%

```

```

12059   \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
12060   }%
12061   {}%
12062 }
12063 {}

```

Load glossaries-extra-stylemods if required.

```
12064 \@glsxtr@redefstyles
```

and set the style:

```
12065 \@glsxtr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for bib2gls and is automatically loaded by the record option.

```
12066 \NeedsTeXFormat{LaTeX2e}
```

```
12067 \ProvidesPackage{glossaries-extra-bib2gls}[2019/04/09 v1.41 (NLCT)]
```

Provide convenient shortcut commands for predefined glossary types.

printunsrtacronyms

```

12068 \ifglsacronym
12069   \providecommand*\printunsrtacronyms[1] [] {%
12070     \printunsrtglossary[type=\acronymtype,#1]}%
12071 \fi

```

printunsrtindex

```

12072 \ifglossaryexists{index}
12073 {
12074   \providecommand*\printunsrtindex[1] [] {%
12075     \printunsrtglossary[type=index,#1]}%
12076 }{}

```

printunsrtsymbols

```

12077 \ifglossaryexists{symbols}
12078 {
12079   \providecommand*\printunsrtsymbols[1] [] {%
12080     \printunsrtglossary[type=symbols,#1]}%
12081 }{}

```

printunsrtnumbers

```

12082 \ifglossaryexists{numbers}
12083 {
12084   \providecommand*\printunsrtnumbers[1] [] {%
12085     \printunsrtglossary[type=numbers,#1]}%
12086 }{}

```

rtabbreviations

```
12087 \ifglossaryexists{abbreviations}
12088 {
12089   \providecommand*\printunsrtabbreviations}[1] []{%
12090     \printunsrtglossary[type=abbreviations,#1]}%
12091 }{}
```

These are some convenient macros for use with custom rules.

`\glshex`

```
12092 \newcommand*\glshex{\string\u}
```

lscapturedgroup

```
12093 \newcommand*\glscapturedgroup{\string\}$}
```

nZeroChildCount For use with bib2gls's save-child-count resource option.

```
12094 \newcommand*\GlsXtrIfHasNonZeroChildCount}[3]{%
12095   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
12096 }
```

rprovidecommand For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```
12097 \newcommand*\glsextrprovidecommand{\providecommand}
```

gl srenewcommand Like `\renewcommand` but only generates a warning rather than an error if the command isn't defined.

```
12098 \newcommand*\gl srenewcommand{\@star@or@long\glsextr@renewcommand}
```

tr@renewcommand

```
12099 \newcommand*\glsextr@renewcommand}[1]{%
12100   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
12101   \expandafter\@ifundefined\@gtempa
12102     {%
12103       \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
12104     }%
12105   \relax
12106   \relax
12107   \let\@ifdefinable\@rc@ifdefinable
12108   \new@command#1%
12109 }
```

lossarylocation For use with indexcounter and bib2gls.

```
12110 \newcommand*\glsextr@wrglossarylocation}[2]{#1}
```

IndexCounterLink

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with `indexcounter` and `bib2gls`.

```
12111 \ifdef\hyperref
12112 {%
12113   \newcommand*\GlsXtrIndexCounterLink}[2]{%
12114     \glstrifhasfield{indexcounter}{#2}%
12115     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
12116     {#1}%
12117   }
12118 }
12119 {
12120   \newcommand*\GlsXtrIndexCounterLink}[2]{#1}
12121 }
```

`\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.

```
12122 \newcommand*\GlsXtrDualField}{dual}
```

`\GlsXtrDualBackLink`

`\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```
12123 \newcommand*\GlsXtrDualBackLink}[2]{%
12124   \glstrifhasfield{\GlsXtrDualField}{#2}%
12125   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
12126   {#2}%
12127 }
```

`\TeXEntryAliases` Convenient shortcut for use with `entry-type-aliases` to alias standard `BIBTEX` entry types to `@bibtexentry`.

```
12128 \newcommand*\GlsXtrBibTeXEntryAliases}{%
12129   article=bibtexentry,
12130   book=bibtexentry,
12131   booklet=bibtexentry,
12132   conference=bibtexentry,
12133   inbook=bibtexentry,
12134   incollection=bibtexentry,
12135   inproceedings=bibtexentry,
12136   manual=bibtexentry,
12137   mastersthesis=bibtexentry,
12138   misc=bibtexentry,
12139   phdthesis=bibtexentry,
12140   proceedings=bibtexentry,
```

```

12141 techreport=bibtexentry,
12142 unpublished=bibtexentry
12143 }

```

`\provideBibTeXFields` Convenient shortcut to define the standard $\text{BIB}_{\text{T}}\text{X}$ fields.

```

12144 \newcommand*\GlsXtrProvideBibTeXFields}{%
12145   \glsaddstoragekey{address}{}{\glsxtrbibaddress}%
12146   \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
12147   \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}%
12148   \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
12149   \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
12150   \glsaddstoragekey{howpublished}{}{\glsxtrbibhowpublished}%
12151   \glsaddstoragekey{institution}{}{\glsxtrbibinstitution}%
12152   \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
12153   \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
12154   \glsaddstoragekey{note}{}{\glsxtrbibnote}%
12155   \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
12156   \glsaddstoragekey{organization}{}{\glsxtrbiborganization}%
12157   \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
12158   \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}%
12159   \glsaddstoragekey{school}{}{\glsxtrbibschooll}%
12160   \glsaddstoragekey{series}{}{\glsxtrbibseries}%
12161   \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
12162   \glsaddstoragekey{bibtex-type}{}{\glsxtrbibtype}%
12163   \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
12164 }

```

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.

```

12165 \newcommand*\glsxtrmultisupplocation}[3]{%
12166   {%
12167     \def\glsxtrsupplocationurl{#2}%
12168     \glsxtrhypernumber{#1}%
12169   }%
12170 }

```

`\trdisplaysupploc`

`\glsxtrdisplaysupploc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```

12171 \newcommand*\glsxtrdisplaysupploc[5]{%
12172   \setentrycounter{#1}{#2}%

```



```
12173 \glstrmultisupplocation{#5}{#4}{#3}%
12174 }
```

`\displaylocnameref` `\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`.

```
12175 \ifundef\hyperlink
12176 {
12177   \newcommand*\glstrdisplaylocnameref}[8]{%
12178     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
12179   }
12180 }
12181 {
```

Default action uses `<hcounter>`. Equations and pages typically don't have a title, so check the counter name.

```
12182 \newcommand*\glstrdisplaylocnameref}[8]{%
12183   \ifstrequal{#2}{equation}%
12184   {\glstrnameref{link}{#3}{(#4)}{#2.#7}{#8}}%
12185   {%
12186     \ifstreempty{#5}%
12187     {%
```

No title, so just use the location as the link text.

```
12188     \glstrnameref{link}{#3}{#4}{#2.#7}{#8}%
12189   }%
12190   {%
12191     \ifstrequal{#2}{page}%
12192     {\glstrnameref{link}{#3}{#4}{#2.#7}{#8}}%
12193     {\glstrnameref{link}{#3}{#5}{#2.#7}{#8}}%
12194   }%
12195   }%
12196 }
12197 }
```

`\glstrnameref{link}` `\glstrfmtnameref{link}{<format>}{<title>}{<href>}{<external file>}`

```
12198 \newcommand*\glstrnameref{link}[4]{%
```

Locally change `\glshypernumber` to `\@firstofone` to remove the normal location hyperlink.

```
12199 \begingroup
12200   \let\glshypernumber\@firstofone
```

If the *<external file>* argument is empty, an internal link is used, otherwise an external one is needed.

```

12201 \ifstrempy{#4}%
12202 {\glxtrfmtinternalnameref{#3}{#1}{#2}}%
12203 {\glxtrfmtexternalnameref{#3}{#1}{#2}{#4}}%
12204 \endgroup
12205 }

```

lsxtrnameloclink

```

\glxtrnamerefloclink{<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.

```

12206 \newcommand{\glxtrnameloclink}[6]{%
12207 \begingroup
12208 \setentrycounter[#1]{#2}%
12209 \def\glxtr@locationhypertext{#5}%
12210 \let\glshypernumber\@firstofone
12211 \def\@glsnumberformat{#3}%
12212 \def\glxtrsupplocationurl{#6}%
12213 \toks@={}%
12214 \@glxtr@bibgls@removespaces#4 \@nil
12215 \endgroup
12216 }

```

ls@removespaces

```

12217 \def\@glxtr@bibgls@removespaces#1 #2\@nil{%
12218 \toks@=\expandafter{\the\toks@#1}%
12219 \ifx\#2\%
12220 \edef\x{\the\toks@}%
12221 \ifx\x\empty
12222 \else
12223 \protected@edef\x{\glsetrycounter\@gls@counterprefix\the\toks@}%
12224 \ifdefvoid\glxtrsupplocationurl
12225 {%
12226 \expandafter\glxtrfmtinternalnameref\expandafter{\x}%
12227 {\@glsnumberformat}{\glxtr@locationhypertext}%
12228 }%
12229 {%
12230 \expandafter\glxtrfmtexternalnameref\expandafter{\x}%
12231 {\@glsnumberformat}{\glxtr@locationhypertext}{\glxtrsupplocationurl}%
12232 }%
12233 \fi
12234 \else
12235 \@gls@ReturnAfterFi{%

```

```

12236 \@glxtr@bibgls@removespaces#2\@nil
12237 }%
12238 \fi
12239 }

```

internalnameref `\glxtrfmtinternalnameloc{<target>}{<format>}{<title>}`

```

12240 \newcommand*{\glxtrfmtinternalnameref}[3]{%
12241 \csuse{#2}{\glsdohyperlink{#1}{#3}}%
12242 }

```

externalnameref `\glxtrfmtexternalnameloc{<target>}{<format>}{<title>}{<file>}`

```

12243 \newcommand*{\glxtrfmtexternalnameref}[4]{%
12244 \csuse{#2}{\hyperref{#4}{}{#1}{#3}}%
12245 }

```

\glxtrSetWidest `\glxtrSetWidest{<type>}{<level>}{<text>}`

As from **bib2gls** v1.8, this is used by the `set-widest` resource option for the `alttree` and the styles provided by the `glossary-longextra` package.

```

12246 \newcommand*{\glxtrSetWidest}[3]{%

```

Check which style options have been provided. (The style packages may not have been loaded.)

```

12247 \ifdef\glupdatewidest
12248 {%
12249 \ifdef\gslongextraUpdateWidest
12250 {%

```

Relevant style packages all loaded. If the `<type>` has been given, append to glossary preamble.

```

12251 \ifstrempy{#1}
12252 {%
12253 \glupdatewidest[#2]{#3}%
12254 \ifnum#2=0\relax
12255 \gslongextraUpdateWidest{#3}%
12256 \else
12257 \gslongextraUpdateWidestChild{#2}{#3}%
12258 \fi
12259 }%
12260 }%

```

```

12261     \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
12262     \ifnum#2=0\relax
12263         \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12264     \else
12265         \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12266     \fi
12267 }%
12268 }%
12269 {%

```

Only altree.

```

12270     \ifstrempy{#1}
12271     {%
12272         \glsupdatewidest[#2]{#3}%
12273     }%
12274     {%
12275         \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
12276     }%
12277 }%
12278 }%
12279 {%

```

\glsupdatewidest hasn't been defined. This could just mean that the glossaries-extra-stylemods package hasn't been loaded.

```

12280     \ifdef\glssetwidest
12281     {%
12282         \ifdef\glslongextraUpdateWidest
12283         {%

```

Relevant glossary-tree and glossary-longextra have been loaded. If the <type> has been given, append to glossary preamble.

```

12284     \ifstrempy{#1}
12285     {%
12286         \glssetwidest[#2]{#3}%
12287         \ifnum#2=0\relax
12288             \glslongextraUpdateWidest{#3}%
12289         \else
12290             \glslongextraUpdateWidestChild{#2}{#3}%
12291         \fi
12292     }%
12293     {%
12294         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
12295         \ifnum#2=0\relax
12296             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12297         \else
12298             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12299         \fi
12300     }%
12301 }%
12302 {%

```

Only alttree.

```
12303     \ifstrempy{#1}
12304     {%
12305         \glsssetwidest[#2]{#3}%
12306     }%
12307     {%
12308         \apptoglossary preamble[#1]{\glsssetwidest[#2]{#3}}%
12309     }%
12310 }%
12311 }%
12312 {%
12313     \ifdef\gls longextraUpdateWidest
12314     {%
        glossary-longextra has been loaded.
12315     \ifstrempy{#1}
12316     {%
12317         \ifnum#2=0\relax
12318             \gls longextraUpdateWidest{#3}%
12319         \else
12320             \gls longextraUpdateWidestChild{#2}{#3}%
12321         \fi
12322     }%
12323     {%
12324         \ifnum#2=0\relax
12325             \apptoglossary preamble[#1]{\gls longextraUpdateWidest{#3}}%
12326         \else
12327             \apptoglossary preamble[#1]{\gls longextraUpdateWidestChild{#2}{#3}}%
12328         \fi
12329     }%
12330 }%
        Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.
12331     }%
12332 }%
12333 }%
12334 }
```

etWidestFallback

```
\glsxtrSetWidestFallback{<max depth>}{<list>}
```

Used when `bib2gls` can't determine the widest name. The `<list>` argument is a comma-separated list of glossary labels. The `<max depth>` refers to the maximum hierarchical depth.

This will either be 0 (only top-level entries) or 2 (up to two child-levels).

```
12335 \newcommand*{\glsxtrSetWidestFallback}[2]{%
12336     \ifnum#1=0\relax
12337     \ifdef\glsFindWidestTopLevelName
12338     {%
```

```

12339   \glsFindWidestTopLevelName[#2]%
12340 }%
12341 {%
12342   \GlossariesExtraWarning{You need stylemods={tree} to
12343     provide a fallback for set-widest}%
12344 }%
12345 \else
12346 \ifdef\glsFindWidestLevelTwo
12347 {%
12348   \glsFindWidestLevelTwo[#2]%
12349   \ifdef\glslongextraUpdateWidestChild
12350   {%
12351     \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
12352     \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
12353   }%
12354   {}}%
12355 }%
12356 {%
12357   \GlossariesExtraWarning{You need stylemods={tree} to
12358     provide a fallback for set-widest}%
12359 }%
12360 \fi
12361 }

```

`r@labelprefixes` List of label prefixes.

```
12362 \newcommand*{\@glsxtr@labelprefixes}{}

```

`arlabelprefixes` List of label prefixes.

```

12363 \newcommand*{\glsxtrclearlabelprefixes}{%
12364   \renewcommand*{\@glsxtr@labelprefixes}{}%
12365 }

```

`raddlabelprefix` Add prefix to the list. These should be added in the order of precedence with the last one as a fallback. This doesn't check against duplicates as it may be useful to replicate a prefix at the end as the fallback.

```

12366 \newcommand*{\glsxtraddlabelprefix}[1]{%
12367   \ifstrempy{#1}%
12368   {\glsxtraddlabelprefix{\empty}}%
12369   {%
12370     \ifdefempty\@glsxtr@labelprefixes
12371     {\def\@glsxtr@labelprefixes{#1}}%
12372     {\appto\@glsxtr@labelprefixes{,#1}}%
12373   }%
12374 }

```

`pendlabelprefix` Inserts at the start of the list.

```

12375 \newcommand*{\glsxtrprependlabelprefix}[1]{%
12376   \ifstrempy{#1}%

```

```

12377 {\glxtrprependlabelprefix{\empty}}%
12378 {%
12379   \ifdefempty\@glxtr@labelprefixes
12380   {\def\@glxtr@labelprefixes{#1}}%
12381   {\preto\@glxtr@labelprefixes{#1,}}%
12382 }%
12383 }

```

`\glxtrifinlabelprefixlist{<prefix>}{<true>}{<false>}`

Test if the given prefix is in the list.

```

12384 \newcommand*\glxtrifinlabelprefixlist}[3]{%
12385   \ifstrempy{#1}%
12386   {\glxtrifinlabelprefixlist{\empty}{#2}{#3}}%
12387   {%
12388     \DTLifinlist{#1}{\@glxtr@labelprefixes}{#2}{#3}%
12389   }%
12390 }

```

`prefixlabellist` This is provided for the benefit of `bib2gls`. It's possible that the user may add more prefixes after the start of the document, but that can lead to inconsistencies. The final element of the list (the fallback) is the only prefix of interest for `bib2gls`.

```

12391 \AtBeginDocument{%
12392   \protected@write\@auxout{}\string\providecommand{\string\@glxtr@prefixlabellist}[1]{}%
12393   \protected@write\@auxout{}\string\@glxtr@prefixlabellist{\@glxtr@labelprefixes}%
12394 }

```

`@prefixedlabel` Iterate through all the prefixes and find the first prefix and label combination that exists. If none found, this could mean that it's the first \LaTeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```

12395 \newcommand*\@glxtr@get@prefixedlabel}[1]{%
12396   \begingroup

```

Initialise to the unprefixed label in the event that the list is empty.

```

12397   \edef\@gls@thislabel{#1}%
12398   \@for\@glxtr@prefix:=\@glxtr@labelprefixes\do
12399   {%
12400     \edef\@gls@thislabel{\@glxtr@prefix#1}%
12401     \ifglsentryexists{\@gls@thislabel}{\@endfortrue}{}%
12402   }%
12403   \edef\x{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\x
12404 }

```

`\dgl` Like `\gls` but tries the prefixes. (Can't use `\p` as that's provided by `glossaries-prefix`.) Since this command is designed for `bib2gls`'s dual entry system, the "d" stands for "dual".

```

12405 \newrobustcmd*\dgl{\@gls@hyp@opt\dgl}

```

\@dGls

```
12406 \newcommand*{\@dGls}[2] [] {%
12407   \@glsxtr@get@prefixedlabel{#2}%
12408   \new@ifnextchar [{\@gls@{#1}\@gls@thislabel}]{\@gls@{#1}\@gls@thislabel} []}%
12409 }
```

\dGlspl

```
12410 \newrobustcmd*{\dGlspl}{\@gls@hyp@opt\dGlspl}
```

\@dGlspl

```
12411 \newcommand*{\@dGlspl}[2] [] {%
12412   \@glsxtr@get@prefixedlabel{#2}%
12413   \new@ifnextchar [{\@glspl@{#1}\@gls@thislabel}]{\@glspl@{#1}\@gls@thislabel} []}%
12414 }
```

\dGls

```
12415 \newrobustcmd*{\dGls}{\@gls@hyp@opt\dGls}
```

\@dGls

```
12416 \newcommand*{\@dGls}[2] [] {%
12417   \@glsxtr@get@prefixedlabel{#2}%
12418   \new@ifnextchar [{\@Gls@{#1}\@gls@thislabel}]{\@Gls@{#1}\@gls@thislabel} []}%
12419 }
```

\dGlspl

```
12420 \newrobustcmd*{\dGlspl}{\@gls@hyp@opt\dGlspl}
```

\@dGlspl

```
12421 \newcommand*{\@dGlspl}[2] [] {%
12422   \@glsxtr@get@prefixedlabel{#2}%
12423   \new@ifnextchar [{\@Glspl@{#1}\@gls@thislabel}]{\@Glspl@{#1}\@gls@thislabel} []}%
12424 }
```

\dGLS

```
12425 \newrobustcmd*{\dGLS}{\@gls@hyp@opt\dGLS}
```

\@dGLS

```
12426 \newcommand*{\@dGLS}[2] [] {%
12427   \@glsxtr@get@prefixedlabel{#2}%
12428   \new@ifnextchar [{\@GLS@{#1}\@gls@thislabel}]{\@GLS@{#1}\@gls@thislabel} []}%
12429 }
```

\dGLSpl

```
12430 \newrobustcmd*{\dGLSpl}{\@gls@hyp@opt\dGLSpl}
```


`\dGLSp1`

```
12431 \newcommand*{\dGLSp1}[2] [] {%
12432   \@glsxtr@get@prefixedlabel{#2}%
12433   \new@ifnextchar[{\@GLSp1@{#1}{\@gls@thislabel}}{\@GLSp1@{#1}{\@gls@thislabel}[]}%
12434 }
```

`\dglslink` Like `\glslink` but tries the prefixes.

```
12435 \newrobustcmd*{\dglslink}[3] [] {%
12436   \@glsxtr@get@prefixedlabel{#2}%
12437   \glslink[#1]{\@gls@thislabel}{#3}%
12438 }
```

`\dglstdisp` Like `\glsdisp` but tries the prefixes.

```
12439 \newrobustcmd*{\dglstdisp}[3] [] {%
12440   \@glsxtr@get@prefixedlabel{#2}%
12441   \glsdisp[#1]{\@gls@thislabel}{#3}%
12442 }
```

Provide missing Greek letters for use in maths mode. These commands are recognised by `bib2gls` and will be mapped to the Mathematical Greek Italic letters. This ensures that the Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The \TeX version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

`\Alpha`

```
12443 \providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
12444 \providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
12445 \providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
12446 \providecommand*{\Zeta}{\mathrm{Z}}
```

`\Eta`

```
12447 \providecommand*{\Eta}{\mathrm{H}}
```

`\Iota`

```
12448 \providecommand*{\Iota}{\mathrm{I}}
```

`\Kappa`

```
12449 \providecommand*{\Kappa}{\mathrm{K}}
```

`\Mu`

```
12450 \providecommand*{\Mu}{\mathrm{M}}
```

`\Nu`

12451 `\providecommand*\Nu{\mathrm{N}}`

`\Omicron`

12452 `\providecommand*\Omicron{\mathrm{O}}`

`\Rho`

12453 `\providecommand*\Rho{\mathrm{P}}`

`\Tau`

12454 `\providecommand*\Tau{\mathrm{T}}`

`\Chi`

12455 `\providecommand*\Chi{\mathrm{X}}`

`\Digamma`

12456 `\providecommand*\Digamma{\mathrm{F}}`

`\omicron`

12457 `\providecommand*\omicron{\mathit{o}}`

Provide corresponding upright characters if upgreek has been loaded. (The upper case characters are the same as above.)

12458 `\@ifpackageloaded{upgreek}%`

12459 `{`

`\Upalpha`

12460 `\providecommand*\Upalpha{\mathrm{A}}`

`\Upbeta`

12461 `\providecommand*\Upbeta{\mathrm{B}}`

`\Upepsilon`

12462 `\providecommand*\Upepsilon{\mathrm{E}}`

`\Upzeta`

12463 `\providecommand*\Upzeta{\mathrm{Z}}`

`\Upeta`

12464 `\providecommand*\Upeta{\mathrm{H}}`

`\Upiota`

12465 `\providecommand*\Upiota{\mathrm{I}}`

`\Upkappa`

12466 `\providecommand*\Upkappa{\mathrm{K}}`

```

\Upmu
12467 \providecommand*\Upmu{\mathrm{M}}

\Upnu
12468 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
12469 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
12470 \providecommand*\Uprho{\mathrm{P}}

\Uptau
12471 \providecommand*\Uptau{\mathrm{T}}

\Upchi
12472 \providecommand*\Upchi{\mathrm{X}}

\upomicron
12473 \providecommand*\upomicron{\mathrm{o}}

12474 }%
12475 {}% upgreek.sty not loaded

```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesextr-<tag>.ldf` (where *<tag>* identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of *<tag>*. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```

sort-rule={\glsxtrcontrolrules
; \glsxtrspacerules
; \glsxtrnonprintablerules
; \glsxtrcombiningdiacriticrules
, \glsxtrhyphenrules
< \glsxtrgeneralpuncrules
< \glsxtrdigitrules
< \glsxtrfractionrules
< \glsxtrGeneralLatinIVrules
< \glsxtrMathItalicGreekIrules
}

```

`\xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they’ve been made active.

```
12476 \newcommand*{\glxtrcontrolrules}{%
12477 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
12478 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
12479 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
12480 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
12481 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
12482 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
12483 0010\string'\string=\glshex 0011
12484 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
12485 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
12486 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
12487 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
12488 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
12489 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
12490 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
12491 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
12492 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
12493 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
12494 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
12495 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
12496 }
```

`\sxtrspacerules` These are space characters.

```
12497 \newcommand*{\glxtrspacerules}{%
12498 \string' \string'\string;
12499 \string'\glshex 00A0\string'\string;
12500 \string'\glshex 2000\string'\string;
12501 \string'\glshex 2001\string'\string;
12502 \string'\glshex 2002\string'\string;
12503 \string'\glshex 2003\string'\string;
12504 \string'\glshex 2004\string'\string;
12505 \string'\glshex 2005\string'\string;
12506 \string'\glshex 2006\string'\string;
12507 \string'\glshex 2007\string'\string;
12508 \string'\glshex 2008\string'\string;
12509 \string'\glshex 2009\string'\string;
12510 \string'\glshex 200A\string'\string;
12511 \string'\glshex 3000\string'
12512 }
```

`\nprintablerules` These are non-printable characters (BOM, tabs, line feed and carriage return).

```
12513 \newcommand*{\glxtrnonprintablerules}{%
12514 \string'\glshex FEFF\string'\string;
12515 \string'\glshex 000A\string'\string;
12516 \string'\glshex 0009\string'\string;
```

```

12517 \string'\glshex 000C\string'\string;
12518 \string'\glshex 000B\string'
12519 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

12520 \newcommand*{\glxtrcombiningsdiacriticrules}{%
12521 \glxtrcombiningsdiacriticIrules\string;
12522 \glxtrcombiningsdiacriticIIrules\string;
12523 \glxtrcombiningsdiacriticIIIrules\string;
12524 \glxtrcombiningsdiacriticIVrules
12525 }

```

diacriticIrules First set of combining diacritic marks.

```

12526 \newcommand*{\glxtrcombiningsdiacriticIrules}{%
12527 \glshex 0301\string;% combining acute
12528 \glshex 0300\string;% combining grave
12529 \glshex 0306\string;% combining breve
12530 \glshex 0302\string;% combining circumflex
12531 \glshex 030C\string;% combining caron
12532 \glshex 030A\string;% combining ring
12533 \glshex 030D\string;% combining vertical line above
12534 \glshex 0308\string;% combining diaeresis
12535 \glshex 030B\string;% combining double acute
12536 \glshex 0303\string;% combining tilde
12537 \glshex 0307\string;% combining dot above
12538 \glshex 0304% combining macron
12539 }

```

diacriticIIrules Second set of combining diacritic marks.

```

12540 \newcommand*{\glxtrcombiningsdiacriticIIrules}{%
12541 \glshex 0337\string;% combining short solidus overlay
12542 \glshex 0327\string;% combining cedilla
12543 \glshex 0328\string;% combining ogonek
12544 \glshex 0323\string;% combining dot below
12545 \glshex 0332\string;% combining low line
12546 \glshex 0305\string;% combining overline
12547 \glshex 0309\string;% combining hook above
12548 \glshex 030E\string;% combining double vertical line above
12549 \glshex 030F\string;% combining double grave accent
12550 \glshex 0310\string;% combining candrabindu
12551 \glshex 0311\string;% combining inverted breve
12552 \glshex 0312\string;% combining turned comma above
12553 \glshex 0313\string;% combining comma above
12554 \glshex 0314\string;% combining reversed comma above
12555 \glshex 0315\string;% combining comma above right
12556 \glshex 0316\string;% combining grave accent below
12557 \glshex 0317% combining acute accent below
12558 }

```

acriticIIIrules Third set of combining diacritic marks.

```
12559 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
12560 \glshex 0318\string;% combining left tack below
12561 \glshex 0319\string;% combining right tack below
12562 \glshex 031A\string;% combining left angle above
12563 \glshex 031B\string;% combining horn
12564 \glshex 031C\string;% combining left half ring below
12565 \glshex 031D\string;% combining up tack below
12566 \glshex 031E\string;% combining down tack below
12567 \glshex 031F\string;% combining plus sign below
12568 \glshex 0320\string;% combining minus sign below
12569 \glshex 0321\string;% combining palatalized hook below
12570 \glshex 0322\string;% combining retroflex hook below
12571 \glshex 0324\string;% combining diaeresis below
12572 \glshex 0325\string;% combining ring below
12573 \glshex 0326\string;% combining comma below
12574 \glshex 0329\string;% combining vertical line below
12575 \glshex 032A\string;% combining bridge below
12576 \glshex 032B\string;% combining inverted double arch below
12577 \glshex 032C\string;% combining caron below
12578 \glshex 032D\string;% combining circumflex accent below
12579 \glshex 032E\string;% combining breve below
12580 \glshex 032F\string;% combining inverted breve below
12581 \glshex 0330\string;% combining tilde below
12582 \glshex 0331\string;% combining macron below
12583 \glshex 0333\string;% combining double low line
12584 \glshex 0334\string;% combining tilde overlay
12585 \glshex 0335\string;% combining short stroke overlay
12586 \glshex 0336\string;% combining long stroke overlay
12587 \glshex 0338\string;% combining long solidus overlay
12588 \glshex 0339\string;% combining combining right half ring below
12589 \glshex 033A\string;% combining inverted bridge below
12590 \glshex 033B\string;% combining square below
12591 \glshex 033C\string;% combining seagull below
12592 \glshex 033D\string;% combining x above
12593 \glshex 033E\string;% combining vertical tilde
12594 \glshex 033F\string;% combining double overline
12595 \glshex 0342\string;% combining Greek perispomeni
12596 \glshex 0344\string;% combining Greek dialytika tonos
12597 \glshex 0345\string;% combining Greek ypogegrammeni
12598 \glshex 0360\string;% combining double tilde
12599 \glshex 0361\string;% combining double inverted breve
12600 \glshex 0483\string;% combining Cyrillic titlo
12601 \glshex 0484\string;% combining Cyrillic palatalization
12602 \glshex 0485\string;% combining Cyrillic dasia pneumata
12603 \glshex 0486% combining Cyrillic psili pneumata
12604 }
```

iacriticIVrules Fourth set of combining diacritic marks.

```

12605 \newcommand*{\glxtrcombingdiacriticIVrules}{%
12606 \glshex 20D0\string;% combining left harpoon above
12607 \glshex 20D1\string;% combining right harpoon above
12608 \glshex 20D2\string;% combining long vertical line overlay
12609 \glshex 20D3\string;% combining short vertical line overlay
12610 \glshex 20D4\string;% combining anticlockwise arrow above
12611 \glshex 20D5\string;% combining clockwise arrow above
12612 \glshex 20D6\string;% combining left arrow above
12613 \glshex 20D7\string;% combining right arrow above
12614 \glshex 20D8\string;% combining ring overlay
12615 \glshex 20D9\string;% combining clockwise ring overlay
12616 \glshex 20DA\string;% combining anticlockwise ring overlay
12617 \glshex 20DB\string;% combining three dots above
12618 \glshex 20DC\string;% combining four dots above
12619 \glshex 20DD\string;% combining enclosing circle
12620 \glshex 20DE\string;% combining enclosing square
12621 \glshex 20DF\string;% combining enclosing diamond
12622 \glshex 20E0\string;% combining enclosing circle backslash
12623 \glshex 20E1% combining left right arrow above
12624 }

```

sxtrhyphenrules Hyphens.

```

12625 \newcommand*{\glxtrhyphenrules}{%
12626 \string'\string-\string'\string;% ASCII hyphen
12627 \glshex 00AD\string;% soft hyphen
12628 \glshex 2010\string;% hyphen
12629 \glshex 2011\string;% non-breaking hyphen
12630 \glshex 2012\string;% figure dash
12631 \glshex 2013\string;% en dash
12632 \glshex 2014\string;% em dash
12633 \glshex 2015\string;% horizontal bar
12634 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
12635 }

```

eneralpuncrules General punctuation.

```

12636 \newcommand*{\glxtrgeneralpuncrules}{%
12637 \glxtrgeneralpuncIrules
12638 \string<\glxtrcurrencyrules
12639 \string<\glxtrgeneralpuncIIrules
12640 }

```

eneralpuncIrules First set of general punctuation.

```

12641 \newcommand*{\glxtrgeneralpuncIrules}{%
12642 \string'\glshex 005F\string'% underscore
12643 \string<\glshex 00AF% macron
12644 \string<\string'\glshex 002C\string'% comma
12645 \string<\string'\glshex 003B\string'% semi-colon
12646 \string<\string'\glshex 003A\string'% colon
12647 \string<\string'\glshex 0021\string'% exclamation mark

```

12648 \string<\glshex 00A1% inverted exclamation mark
12649 \string<\string'\glshex 003F\string'% question mark
12650 \string<\glshex 00BF% inverted question mark
12651 \string<\string'\glshex 002F\string'% solidus
12652 \string<\string'\glshex 002E\string'% full stop
12653 \string<\glshex 00B4% acute accent
12654 \string<\string'\glshex 0060\string'% grave accent
12655 \string<\string'\glshex 005E\string'% circumflex accent
12656 \string<\glshex 00A8% diaeresis
12657 \string<\string'\glshex 007E\string'% tilde
12658 \string<\glshex 00B7% middle dot
12659 \string<\glshex 00B8% cedilla
12660 \string<\string'\glshex 0027\string'% straight apostrophe
12661 \string<\string'\glshex 0022\string'% straight double quote
12662 \string<\glshex 00AB% left guillemet
12663 \string<\glshex 00BB% right guillemet
12664 \string<\string'\glshex 0028\string'% left parenthesis
12665 \string=&\glshex 207D\string=&\glshex 208D% super/subscript left parenthesis
12666 \string<\string'\glshex 0029\string'% right parenthesis
12667 \string=&\glshex 207E\string=&\glshex 208E% super/subscript right parenthesis
12668 \string<\string'\glshex 005B\string'% left square bracket
12669 \string<\string'\glshex 005D\string'% right square bracket
12670 \string<\string'\glshex 007B\string'% left curly bracket
12671 \string<\string'\glshex 007D\string'% right curly bracket
12672 \string<\glshex 00A7% section sign
12673 \string<\glshex 00B6% pilcrow sign
12674 \string<\glshex 00A9% copyright sign
12675 \string<\glshex 00AE% registered sign
12676 \string<\string'\glshex 0040\string'% at sign
12677 }

trcurrencyrules General punctuation.

12678 \newcommand*{\glxtrcurrencyrules}{%
12679 \glshex 00A4% currency sign
12680 \string<\glshex 0E3F% Thai currency symbol baht
12681 \string<\glshex 00A2% cent sign
12682 \string<\glshex 20A1% colon sign
12683 \string<\glshex 20A2% cruzeiro sign
12684 \string<\string'\glshex 0024\string'% dollar sign
12685 \string<\glshex 20AB% dong sign
12686 \string<\glshex 20AC% euro sign
12687 \string<\glshex 20A3% French franc sign
12688 \string<\glshex 20A4% lira sign
12689 \string<\glshex 20A5% mill sign
12690 \string<\glshex 20A6% naira sign
12691 \string<\glshex 20A7% peseta sign
12692 \string<\glshex 00A3% pound sign
12693 \string<\glshex 20A8% rupee sign
12694 \string<\glshex 20AA% new sheqel sign

12695 \string<\glshex 20A9% won sign
 12696 \string<\glshex 00A5% yen sign
 12697 }

eralpuncIIrules Second set of general punctuation.

12698 \newcommand*{\glxtrgeneralpuncIIrules}{%
 12699 \string'\glshex 002A\string'% asterisk
 12700 \string<\string'\glshex 005C\string'% backslash
 12701 \string<\string'\glshex 0026\string'% ampersand
 12702 \string<\string'\glshex 0023\string'% hash sign
 12703 \string<\string'\glshex 0025\string'% percent sign
 12704 \string<\string'\glshex 002B\string'% plus sign
 12705 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
 12706 \string<\glshex 00B1% plus-minus sign
 12707 \string<\glshex 00F7% division sign
 12708 \string<\glshex 00D7% multiplication sign
 12709 \string<\string'\glshex 003C\string'% less-than sign
 12710 \string<\string'\glshex 003D\string'% equals sign
 12711 \string<\string'\glshex 003E\string'% greater-than sign
 12712 \string<\glshex 00AC% not sign
 12713 \string<\string'\glshex 007C\string'% vertical bar (pipe)
 12714 \string<\glshex 00A6% broken bar
 12715 \string<\glshex 00B0% degree sign
 12716 \string<\glshex 00B5% micron sign
 12717 }

eralLatinIrules Basic Latin alphabet.

12718 \newcommand*{\glxtrGeneralLatinIrules}{%
 12719 \glxtrLatinA
 12720 \string<b,B%
 12721 \string<c,C%
 12722 \string<d,D%
 12723 \string<\glxtrLatinE
 12724 \string<f,F%
 12725 \string<g,G%
 12726 \string<\glxtrLatinH
 12727 \string<\glxtrLatinI
 12728 \string<j,J%
 12729 \string<\glxtrLatinK
 12730 \string<\glxtrLatinL
 12731 \string<\glxtrLatinM
 12732 \string<\glxtrLatinN
 12733 \string<\glxtrLatinO
 12734 \string<\glxtrLatinP
 12735 \string<q,Q%
 12736 \string<r,R%
 12737 \string<\glxtrLatinS
 12738 \string<\glxtrLatinT
 12739 \string<u,U%

```

12740 \string<v,V%
12741 \string<w,W%
12742 \string<\glxtrLatinX
12743 \string<y,Y%
12744 \string<z,Z
12745 }

```

alLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12746 \newcommand*{\glxtrGeneralLatinIIrules}{%
12747 \glxtrLatinA
12748 \string<b,B%
12749 \string<c,C%
12750 \string<d,D%
12751 \string<\glxtrLatinEth
12752 \string<\glxtrLatinE
12753 \string<f,F%
12754 \string<g,G%
12755 \string<\glxtrLatinH
12756 \string<\glxtrLatinI
12757 \string<j,J%
12758 \string<\glxtrLatinK
12759 \string<\glxtrLatinL
12760 \string<\glxtrLatinM
12761 \string<\glxtrLatinN
12762 \string<\glxtrLatinO
12763 \string<\glxtrLatinP
12764 \string<q,Q%
12765 \string<r,R%
12766 \string<\glxtrLatinS
12767 \string& SS \string, \glxtrLatinEszettSs
12768 \string<\glxtrLatinT
12769 \string<u,U%
12770 \string<v,V%
12771 \string<w,W%
12772 \string<\glxtrLatinX
12773 \string<y,Y%
12774 \string<z,Z%
12775 }

```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12776 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12777 \glxtrLatinA
12778 \string<b,B%
12779 \string<c,C%
12780 \string<d,D%
12781 \string<\glxtrLatinEth
12782 \string<\glxtrLatinE
12783 \string<f,F%
12784 \string<g,G%

```

```

12785 \string<\glxtrLatinH
12786 \string<\glxtrLatinI
12787 \string<j,J%
12788 \string<\glxtrLatinK
12789 \string<\glxtrLatinL
12790 \string<\glxtrLatinM
12791 \string<\glxtrLatinN
12792 \string<\glxtrLatinO
12793 \string<\glxtrLatinP
12794 \string<q,Q%
12795 \string<r,R%
12796 \string<\glxtrLatinS
12797 \string& SZ, \glxtrLatinEszettSz
12798 \string<\glxtrLatinT
12799 \string<u,U%
12800 \string<v,V%
12801 \string<w,W%
12802 \string<\glxtrLatinX
12803 \string<y,Y%
12804 \string<z,Z%
12805 }

```

`\glxtrGeneralLatinIVrules` General Latin alphabet (Æ treated as AE and Œ treated as OE, Þ treated as TH, ß treated as SS, eth between D and E).

```

12806 \newcommand*{\glxtrGeneralLatinIVrules}{%
12807 \glxtrLatinA
12808 \string& AE , \glxtrLatinAELigature
12809 \string<b,B%
12810 \string<c,C%
12811 \string<d,D%
12812 \string<\glxtrLatinEth
12813 \string<\glxtrLatinE
12814 \string<f,F%
12815 \string<g,G%
12816 \string<\glxtrLatinH
12817 \string<\glxtrLatinI
12818 \string<j,J%
12819 \string<\glxtrLatinK
12820 \string<\glxtrLatinL
12821 \string<\glxtrLatinM
12822 \string<\glxtrLatinN
12823 \string<\glxtrLatinO
12824 \string& OE , \glxtrLatinOELigature
12825 \string<\glxtrLatinP
12826 \string<q,Q%
12827 \string<r,R%
12828 \string<\glxtrLatinS
12829 \string& SS , \glxtrLatinEszettSs
12830 \string<\glxtrLatinT

```

```

12831 \string& th =\glshex 00DE
12832 \string& TH =\glshex 00FE
12833 \string<u,U%
12834 \string<v,V%
12835 \string<w,W%
12836 \string<\glxtrLatinX
12837 \string<y,Y%
12838 \string<z,Z%
12839 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12840 \newcommand*{\glxtrGeneralLatinVrules}{%
12841 \glxtrLatinA
12842 \string<b,B%
12843 \string<c,C%
12844 \string<d,D%
12845 \string<\glxtrLatinEth
12846 \string<\glxtrLatinE
12847 \string<f,F%
12848 \string<g,G%
12849 \string<\glxtrLatinH
12850 \string<\glxtrLatinI
12851 \string<j,J%
12852 \string<\glxtrLatinK
12853 \string<\glxtrLatinL
12854 \string<\glxtrLatinM
12855 \string<\glxtrLatinN
12856 \string<\glxtrLatinO
12857 \string<\glxtrLatinP
12858 \string<q,Q%
12859 \string<r,R%
12860 \string<\glxtrLatinS
12861 \string& SS , \glxtrLatinEszettSs
12862 \string<\glxtrLatinT
12863 \string& th =\glshex 00DE
12864 \string& TH =\glshex 00FE
12865 \string<u,U%
12866 \string<v,V%
12867 \string<w,W%
12868 \string<\glxtrLatinX
12869 \string<y,Y%
12870 \string<z,Z%
12871 }

```

eralLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12872 \newcommand*{\glxtrGeneralLatinVIrules}{%
12873 \glxtrLatinA
12874 \string<b,B%
12875 \string<c,C%

```

```

12876 \string<d,D%
12877 \string<\glxtrLatinEth
12878 \string<\glxtrLatinE
12879 \string<f,F%
12880 \string<g,G%
12881 \string<\glxtrLatinH
12882 \string<\glxtrLatinI
12883 \string<j,J%
12884 \string<\glxtrLatinK
12885 \string<\glxtrLatinL
12886 \string<\glxtrLatinM
12887 \string<\glxtrLatinN
12888 \string<\glxtrLatinO
12889 \string<\glxtrLatinP
12890 \string<q,Q%
12891 \string<r,R%
12892 \string<\glxtrLatinS
12893 \string& SZ , \glxtrLatinEszettSz
12894 \string<\glxtrLatinT
12895 \string& th =\glshex 00DE
12896 \string& TH =\glshex 00FE
12897 \string<u,U%
12898 \string<v,V%
12899 \string<w,W%
12900 \string<\glxtrLatinX
12901 \string<y,Y%
12902 \string<z,Z%
12903 }

```

alLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, Œ between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

12904 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12905 \glxtrLatinA
12906 \string<\glxtrLatinAELigature
12907 \string<b,B%
12908 \string<c,C%
12909 \string<d,D%
12910 \string<\glxtrLatinEth
12911 \string<\glxtrLatinE
12912 \string<f,F%
12913 \string<\glxtrLatinInsularG
12914 \string<\glxtrLatinH
12915 \string<\glxtrLatinI
12916 \string<j,J%
12917 \string<\glxtrLatinK
12918 \string<\glxtrLatinL
12919 \string<\glxtrLatinM
12920 \string<\glxtrLatinN
12921 \string<\glxtrLatinO

```

12922 \string<\glxtrLatinOELigature
 12923 \string<\glxtrLatinP
 12924 \string<q,Q%
 12925 \string<r,R%
 12926 \string<\glshex 017F=\glxtrLatinS % s and long s
 12927 \string<\glxtrLatinT
 12928 \string<\glxtrLatinThorn
 12929 \string<u,U%
 12930 \string<v,V%
 12931 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
 12932 \string<\glxtrLatinX
 12933 \string<y,Y%
 12934 \string<z,Z%
 12935 }

1LatinVIIIrules General Latin alphabet (\mathcal{A} treated as AE and \mathcal{C} treated as OE, \mathcal{P} treated as TH, \mathcal{B} treated as SS, eth treated as D, \mathcal{O} treated as O, \mathcal{L} treated as L).

12936 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
 12937 \glxtrLatinA
 12938 \string& AE , \glxtrLatinAELigature
 12939 \string<b,B%
 12940 \string<c,C%
 12941 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
 12942 \string<\glxtrLatinE
 12943 \string<f,F%
 12944 \string<g,G%
 12945 \string<\glxtrLatinH
 12946 \string<\glxtrLatinI
 12947 \string<j,J%
 12948 \string<\glxtrLatinK
 12949 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
 12950 \string<\glxtrLatinM
 12951 \string<\glxtrLatinN
 12952 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
 12953 \string& OE , \glxtrLatinOELigature
 12954 \string<\glxtrLatinP
 12955 \string<q,Q%
 12956 \string<r,R%
 12957 \string<\glxtrLatinS
 12958 \string& SS , \glxtrLatinEszettSs
 12959 \string<\glxtrLatinT
 12960 \string& th =\glshex 00DE
 12961 \string& TH =\glshex 00FE
 12962 \string<u,U%
 12963 \string<v,V%
 12964 \string<w,W%
 12965 \string<\glxtrLatinX
 12966 \string<y,Y%
 12967 \string<z,Z%

12968 }

\glxtrLatinA

12969 \newcommand*{\glxtrLatinA}{%
12970 a\string=\glshex 00AA\string=\glshex 2090,A
12971 }

\glxtrLatinE

12972 \newcommand*{\glxtrLatinE}{%
12973 e\string=\glshex 2091,E
12974 }

\glxtrLatinH

12975 \newcommand*{\glxtrLatinH}{%
12976 h\string=\glshex 2095,H
12977 }

\glxtrLatinI

12978 \newcommand*{\glxtrLatinI}{%
12979 i\string=\glshex 2071,I
12980 }

\glxtrLatinK

12981 \newcommand*{\glxtrLatinK}{%
12982 k\string=\glshex 2096,K
12983 }

\glxtrLatinL

12984 \newcommand*{\glxtrLatinL}{%
12985 l\string=\glshex 2097,L
12986 }

\glxtrLatinM

12987 \newcommand*{\glxtrLatinM}{%
12988 m\string=\glshex 2098,M
12989 }

\glxtrLatinN

12990 \newcommand*{\glxtrLatinN}{%
12991 n\string=\glshex 207F\string=\glshex 2099,N
12992 }

\glxtrLatinO

12993 \newcommand*{\glxtrLatinO}{%
12994 o\string=\glshex 00BA\string=\glshex 2092,O
12995 }

`\glxtrLatinP`

```
12996 \newcommand*{\glxtrLatinP}{%
12997   p\string=\glshex 209A,P
12998 }
```

`\glxtrLatinS`

```
12999 \newcommand*{\glxtrLatinS}{%
13000   s\string=\glshex 209B,S
13001 }
```

`\glxtrLatinT`

```
13002 \newcommand*{\glxtrLatinT}{%
13003   t\string=\glshex 209C,T
13004 }
```

`\glxtrLatinX`

```
13005 \newcommand*{\glxtrLatinX}{%
13006   x\string=\glshex 2093,X
13007 }
```

`lsxtrLatinSchwa` Latin schwa (lower case, subscript and upper case).

```
13008 \newcommand*{\glxtrLatinSchwa}{%
13009   \glshex 0259\string=\glshex 2094,\glshex 018F
13010 }
```

`trLatinEszettSs`

```
13011 \newcommand*{\glxtrLatinEszettSs}{%
13012   \glshex 00DF% eszett
13013   \string=\glshex 017Fs % long S s
13014 }
```

`trLatinEszettSz`

```
13015 \newcommand*{\glxtrLatinEszettSz}{%
13016   \glshex 00DF% eszett
13017   \string= \glshex 017Fz % long S z
13018 }
```

`\glxtrLatinEth`

```
13019 \newcommand*{\glxtrLatinEth}{%
13020   \glshex 00F0,\glshex 00D0% eth
13021 }
```

`lsxtrLatinThorn`

```
13022 \newcommand*{\glxtrLatinThorn}{%
13023   \glshex 00FE,\glshex 00DE% thorn
13024 }
```


LatinAELigature

```
13025 \newcommand*{\glxtrLatinAELigature}{%
13026 \glshex 00E6,\glshex 00C6% AE-ligature
13027 }
```

LatinOELigature

```
13028 \newcommand*{\glxtrLatinOELigature}{%
13029 \glshex 0153,\glshex 0152% OE-ligature
13030 }
```

\glxtrLatinAA

```
13031 \newcommand*{\glxtrLatinAA}{%
13032 \glshex 00E5=a\glshex 030A,% \aa
13033 \glshex 00C5=A\glshex 030A% \AA
13034 }
```

glxtrLatinWynn

```
13035 \newcommand*{\glxtrLatinWynn}{%
13036 \glshex 01BF,\glshex 01F7% wynn
13037 }
```

trLatinInsularG

```
13038 \newcommand*{\glxtrLatinInsularG}{%
13039 \glshex 1D79,\glshex A77D% insular G
13040 \string; g, G
13041 }
```

sxtrLatinOslash

```
13042 \newcommand*{\glxtrLatinOslash}{%
13043 \glshex 00F8,\glshex 00D8% \o, \O
13044 }
```

sxtrLatinLslash

```
13045 \newcommand*{\glxtrLatinLslash}{%
13046 \glshex 0142,\glshex 0141% \l, \L
13047 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
13048 \newcommand*{\glxtrMathUpGreekIrules}{%
13049 \glxtrUpAlpha
13050 \string<\glxtrUpBeta
13051 \string<\glxtrUpGamma
13052 \string<\glxtrUpDelta
13053 \string<\glxtrUpEpsilon
13054 \string<\glxtrUpDigamma
13055 \string<\glxtrUpZeta
13056 \string<\glxtrUpEta
13057 \string<\glxtrUpTheta
```

```

13058 \string<\glxtrUpIota
13059 \string<\glxtrUpKappa
13060 \string<\glxtrUpLambda
13061 \string<\glxtrUpMu
13062 \string<\glxtrUpNu
13063 \string<\glxtrUpXi
13064 \string<\glxtrUpOmicron
13065 \string<\glxtrUpPi
13066 \string<\glxtrUpRho
13067 \string<\glxtrUpSigma
13068 \string<\glxtrUpTau
13069 \string<\glxtrUpUpsilon
13070 \string<\glxtrUpPhi
13071 \string<\glxtrUpChi
13072 \string<\glxtrUpPsi
13073 \string<\glxtrUpOmega
13074 }

```

`hUpGreekIIrules` Doesn't include digamma.

```

13075 \newcommand*{\glxtrMathUpGreekIIrules}{%
13076 \glxtrUpAlpha
13077 \string<\glxtrUpBeta
13078 \string<\glxtrUpGamma
13079 \string<\glxtrUpDelta
13080 \string<\glxtrUpEpsilon
13081 \string<\glxtrUpZeta
13082 \string<\glxtrUpEta
13083 \string<\glxtrUpTheta
13084 \string<\glxtrUpIota
13085 \string<\glxtrUpKappa
13086 \string<\glxtrUpLambda
13087 \string<\glxtrUpMu
13088 \string<\glxtrUpNu
13089 \string<\glxtrUpXi
13090 \string<\glxtrUpOmicron
13091 \string<\glxtrUpPi
13092 \string<\glxtrUpRho
13093 \string<\glxtrUpSigma
13094 \string<\glxtrUpTau
13095 \string<\glxtrUpUpsilon
13096 \string<\glxtrUpPhi
13097 \string<\glxtrUpChi
13098 \string<\glxtrUpPsi
13099 \string<\glxtrUpOmega
13100 }

```

`alicGreekIrules` Includes (upright) digamma between epsilon and zeta (there isn't an italic digamma), so don't mix with `\glxtrMathUpGreekIrules` or there may be unexpected results.

```

13101 \newcommand*{\glxtrMathItalicGreekIrules}{%

```

```

13102 \glxtrMathItalicAlpha
13103 \string<\glxtrMathItalicBeta
13104 \string<\glxtrMathItalicGamma
13105 \string<\glxtrMathItalicDelta
13106 \string<\glxtrMathItalicEpsilon
13107 \string<\glxtrUpDigamma
13108 \string<\glxtrMathItalicZeta
13109 \string<\glxtrMathItalicEta
13110 \string<\glxtrMathItalicTheta
13111 \string<\glxtrMathItalicIota
13112 \string<\glxtrMathItalicKappa
13113 \string<\glxtrMathItalicLambda
13114 \string<\glxtrMathItalicMu
13115 \string<\glxtrMathItalicNu
13116 \string<\glxtrMathItalicXi
13117 \string<\glxtrMathItalicOmicron
13118 \string<\glxtrMathItalicPi
13119 \string<\glxtrMathItalicRho
13120 \string<\glxtrMathItalicSigma
13121 \string<\glxtrMathItalicTau
13122 \string<\glxtrMathItalicUpsilon
13123 \string<\glxtrMathItalicPhi
13124 \string<\glxtrMathItalicChi
13125 \string<\glxtrMathItalicPsi
13126 \string<\glxtrMathItalicOmega
13127 }

```

licGreekIIrules Doesn't include digamma.

```

13128 \newcommand*{\glxtrMathItalicGreekIIrules}{%
13129 \glxtrMathItalicAlpha
13130 \string<\glxtrMathItalicBeta
13131 \string<\glxtrMathItalicGamma
13132 \string<\glxtrMathItalicDelta
13133 \string<\glxtrMathItalicEpsilon
13134 \string<\glxtrMathItalicZeta
13135 \string<\glxtrMathItalicEta
13136 \string<\glxtrMathItalicTheta
13137 \string<\glxtrMathItalicIota
13138 \string<\glxtrMathItalicKappa
13139 \string<\glxtrMathItalicLambda
13140 \string<\glxtrMathItalicMu
13141 \string<\glxtrMathItalicNu
13142 \string<\glxtrMathItalicXi
13143 \string<\glxtrMathItalicOmicron
13144 \string<\glxtrMathItalicPi
13145 \string<\glxtrMathItalicRho
13146 \string<\glxtrMathItalicSigma
13147 \string<\glxtrMathItalicTau
13148 \string<\glxtrMathItalicUpsilon

```

```

13149 \string<\glxtrMathItalicPhi
13150 \string<\glxtrMathItalicChi
13151 \string<\glxtrMathItalicPsi
13152 \string<\glxtrMathItalicOmega
13153 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

13154 newcommand*{\glxtrMathItalicUpperGreekIrules}{%
13155 \glshex 1D6E2% upper case alpha (maths italic)
13156 \string<\glshex 1D6E3% upper case beta (maths italic)
13157 \string<\glshex 1D6E4% upper case gamma (maths italic)
13158 \string<\glshex 1D6E5% upper case delta (maths italic)
13159 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13160 \string<\glshex 03DC% upper case digamma
13161 \string<\glshex 1D6E7% upper case zeta (maths italic)
13162 \string<\glshex 1D6E8% upper case eta (maths italic)
13163 \string<\glshex 1D6E9% upper case theta (maths italic)
13164 \string=\glshex 1D6F3% upper case theta variant (maths italic)
13165 \string<\glshex 1D6EA% upper case iota (maths italic)
13166 \string<\glshex 1D6EB% upper case kappa (maths italic)
13167 \string<\glshex 1D6EC% upper case lambda (maths italic)
13168 \string<\glshex 1D6ED% upper case mu (maths italic)
13169 \string<\glshex 1D6EE% upper case nu (maths italic)
13170 \string<\glshex 1D6EF% upper case xi (maths italic)
13171 \string<\glshex 1D6F0% upper case omicron (maths italic)
13172 \string<\glshex 1D6F1% upper case pi (maths italic)
13173 \string<\glshex 1D6F2% upper case rho (maths italic)
13174 \string<\glshex 1D6F4% upper case sigma (maths italic)
13175 \string<\glshex 1D6F5% upper case tau (maths italic)
13176 \string<\glshex 1D6F6% upper case upsilon (maths italic)
13177 \string<\glshex 1D6F7% upper case phi (maths italic)
13178 \string<\glshex 1D6F8% upper case chi (maths italic)
13179 \string<\glshex 1D6F9% upper case psi (maths italic)
13180 \string<\glshex 1D6FA% upper case omega (maths italic)
13181 }

```

UpperGreekIIrules Upper case only (doesn't include upright digamma).

```

13182 newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
13183 \glshex 1D6E2% upper case alpha (maths italic)
13184 \string<\glshex 1D6E3% upper case beta (maths italic)
13185 \string<\glshex 1D6E4% upper case gamma (maths italic)
13186 \string<\glshex 1D6E5% upper case delta (maths italic)
13187 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13188 \string<\glshex 1D6E7% upper case zeta (maths italic)
13189 \string<\glshex 1D6E8% upper case eta (maths italic)
13190 \string<\glshex 1D6E9% upper case theta (maths italic)
13191 \string=\glshex 1D6F3% upper case theta variant (maths italic)
13192 \string<\glshex 1D6EA% upper case iota (maths italic)
13193 \string<\glshex 1D6EB% upper case kappa (maths italic)

```

13194 \string<\glshex 1D6EC% upper case lambda (maths italic)
13195 \string<\glshex 1D6ED% upper case mu (maths italic)
13196 \string<\glshex 1D6EE% upper case nu (maths italic)
13197 \string<\glshex 1D6EF% upper case xi (maths italic)
13198 \string<\glshex 1D6F0% upper case omicron (maths italic)
13199 \string<\glshex 1D6F1% upper case pi (maths italic)
13200 \string<\glshex 1D6F2% upper case rho (maths italic)
13201 \string<\glshex 1D6F4% upper case sigma (maths italic)
13202 \string<\glshex 1D6F5% upper case tau (maths italic)
13203 \string<\glshex 1D6F6% upper case upsilon (maths italic)
13204 \string<\glshex 1D6F7% upper case phi (maths italic)
13205 \string<\glshex 1D6F8% upper case chi (maths italic)
13206 \string<\glshex 1D6F9% upper case psi (maths italic)
13207 \string<\glshex 1D6FA% upper case omega (maths italic)
13208 }

LowerGreekIrules Lower case only (includes upright digamma).

13209 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%
13210 \glshex 1D6FC% lower case alpha (maths italic)
13211 \string<\glshex 1D6FD% lower case beta (maths italic)
13212 \string<\glshex 1D6FE% lower case gamma (maths italic)
13213 \string<\glshex 1D6FF% lower case delta (maths italic)
13214 \string<\glshex 1D700% lower case epsilon (maths italic)
13215 \string=\glshex 1D716% lower case epsilon variant (maths italic)
13216 \string<\glshex 03DD% lower case digamma
13217 \string<\glshex 1D701% lower case zeta (maths italic)
13218 \string<\glshex 1D702% lower case eta (maths italic)
13219 \string<\glshex 1D703% lower case theta (maths italic)
13220 \string=\glshex 1D717% lower case theta variant (maths italic)
13221 \string<\glshex 1D704% lower case iota (maths italic)
13222 \string<\glshex 1D705% lower case kappa (maths italic)
13223 \string=\glshex 1D718% lower case kappa variant (maths italic)
13224 \string<\glshex 1D706% lower case lambda (maths italic)
13225 \string<\glshex 1D707% lower case mu (maths italic)
13226 \string<\glshex 1D708% lower case nu (maths italic)
13227 \string<\glshex 1D709% lower case xi (maths italic)
13228 \string<\glshex 1D70A% lower case omicron (maths italic)
13229 \string<\glshex 1D70B% lower case pi (maths italic)
13230 \string=\glshex 1D71B% lower case pi variant (maths italic)
13231 \string<\glshex 1D70C% lower case rho (maths italic)
13232 \string=\glshex 1D71A% lower case rho variant (maths italic)
13233 \string<\glshex 1D70D% lower case final sigma (maths italic)
13234 \string=\glshex 1D70E% lower case sigma (maths italic)
13235 \string<\glshex 1D70F% lower case tau (maths italic)
13236 \string<\glshex 1D710% lower case upsilon (maths italic)
13237 \string<\glshex 1D711% lower case phi (maths italic)
13238 \string=\glshex 1D719% lower case phi variant (maths italic)
13239 \string<\glshex 1D712% lower case chi (maths italic)
13240 \string<\glshex 1D713% lower case psi (maths italic)

13241 \string<\glshex 1D714% lower case omega (maths italic)
 13242 }

LowerGreekiIrules Lower case only (doesn't includes upright digamma).

13243 \newcommand*{\glxtrMathItalicLowerGreekiIrules}{%
 13244 \glshex 1D6FC% lower case alpha (maths italic)
 13245 \string<\glshex 1D6FD% lower case beta (maths italic)
 13246 \string<\glshex 1D6FE% lower case gamma (maths italic)
 13247 \string<\glshex 1D6FF% lower case delta (maths italic)
 13248 \string<\glshex 1D700% lower case epsilon (maths italic)
 13249 \string=\glshex 1D716% lower case epsilon variant (maths italic)
 13250 \string<\glshex 1D701% lower case zeta (maths italic)
 13251 \string<\glshex 1D702% lower case eta (maths italic)
 13252 \string<\glshex 1D703% lower case theta (maths italic)
 13253 \string=\glshex 1D717% lower case theta variant (maths italic)
 13254 \string<\glshex 1D704% lower case iota (maths italic)
 13255 \string<\glshex 1D705% lower case kappa (maths italic)
 13256 \string=\glshex 1D718% lower case kappa variant (maths italic)
 13257 \string<\glshex 1D706% lower case lambda (maths italic)
 13258 \string<\glshex 1D707% lower case mu (maths italic)
 13259 \string<\glshex 1D708% lower case nu (maths italic)
 13260 \string<\glshex 1D709% lower case xi (maths italic)
 13261 \string<\glshex 1D70A% lower case omicron (maths italic)
 13262 \string<\glshex 1D70B% lower case pi (maths italic)
 13263 \string=\glshex 1D71B% lower case pi variant (maths italic)
 13264 \string<\glshex 1D70C% lower case rho (maths italic)
 13265 \string=\glshex 1D71A% lower case rho variant (maths italic)
 13266 \string<\glshex 1D70D% lower case final sigma (maths italic)
 13267 \string=\glshex 1D70E% lower case sigma (maths italic)
 13268 \string<\glshex 1D70F% lower case tau (maths italic)
 13269 \string<\glshex 1D710% lower case upsilon (maths italic)
 13270 \string<\glshex 1D711% lower case phi (maths italic)
 13271 \string=\glshex 1D719% lower case phi variant (maths italic)
 13272 \string<\glshex 1D712% lower case chi (maths italic)
 13273 \string<\glshex 1D713% lower case psi (maths italic)
 13274 \string<\glshex 1D714% lower case omega (maths italic)
 13275 }

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

13276 \newcommand*{\glxtrMathGreekIrules}{%
 13277 \glxtrMathItalicAlpha
 13278 \string;\glxtrUpAlpha
 13279 \string<\glxtrMathItalicBeta
 13280 \string;\glxtrUpBeta
 13281 \string<\glxtrMathItalicGamma
 13282 \string;\glxtrUpGamma
 13283 \string<\glxtrMathItalicDelta
 13284 \string;\glxtrUpDelta
 13285 \string<\glxtrMathItalicEpsilon

```

13286 \string;\glxtrUpEpsilon
13287 \string<\glxtrUpDigamma
13288 \string<\glxtrMathItalicZeta
13289 \string;\glxtrUpZeta
13290 \string<\glxtrMathItalicEta
13291 \string;\glxtrUpEta
13292 \string<\glxtrMathItalicTheta
13293 \string;\glxtrUpTheta
13294 \string<\glxtrMathItalicIota
13295 \string;\glxtrUpIota
13296 \string<\glxtrMathItalicKappa
13297 \string;\glxtrUpKappa
13298 \string<\glxtrMathItalicLambda
13299 \string;\glxtrUpLambda
13300 \string<\glxtrMathItalicMu
13301 \string;\glxtrUpMu
13302 \string<\glxtrMathItalicNu
13303 \string;\glxtrUpNu
13304 \string<\glxtrMathItalicXi
13305 \string;\glxtrUpXi
13306 \string<\glxtrMathItalicOmicron
13307 \string;\glxtrUpOmicron
13308 \string<\glxtrMathItalicPi
13309 \string;\glxtrUpPi
13310 \string<\glxtrMathItalicRho
13311 \string;\glxtrUpRho
13312 \string<\glxtrMathItalicSigma
13313 \string;\glxtrUpSigma
13314 \string<\glxtrMathItalicTau
13315 \string;\glxtrUpTau
13316 \string<\glxtrMathItalicUpsilon
13317 \string;\glxtrUpUpsilon
13318 \string<\glxtrMathItalicPhi
13319 \string;\glxtrUpPhi
13320 \string<\glxtrMathItalicChi
13321 \string;\glxtrUpChi
13322 \string<\glxtrMathItalicPsi
13323 \string;\glxtrUpPsi
13324 \string<\glxtrMathItalicOmega
13325 \string;\glxtrUpOmega
13326 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

13327 \newcommand*{\glxtrMathGreekIIrules}{%
13328 \glxtrMathItalicAlpha
13329 \string;\glxtrUpAlpha
13330 \string<\glxtrMathItalicBeta
13331 \string;\glxtrUpBeta
13332 \string<\glxtrMathItalicGamma

```

```

13333 \string;\glxtrUpGamma
13334 \string<\glxtrMathItalicDelta
13335 \string;\glxtrUpDelta
13336 \string<\glxtrMathItalicEpsilon
13337 \string;\glxtrUpEpsilon
13338 \string<\glxtrMathItalicZeta
13339 \string;\glxtrUpZeta
13340 \string<\glxtrMathItalicEta
13341 \string;\glxtrUpEta
13342 \string<\glxtrMathItalicTheta
13343 \string;\glxtrUpTheta
13344 \string<\glxtrMathItalicIota
13345 \string;\glxtrUpIota
13346 \string<\glxtrMathItalicKappa
13347 \string;\glxtrUpKappa
13348 \string<\glxtrMathItalicLambda
13349 \string;\glxtrUpLambda
13350 \string<\glxtrMathItalicMu
13351 \string;\glxtrUpMu
13352 \string<\glxtrMathItalicNu
13353 \string;\glxtrUpNu
13354 \string<\glxtrMathItalicXi
13355 \string;\glxtrUpXi
13356 \string<\glxtrMathItalicOmicron
13357 \string;\glxtrUpOmicron
13358 \string<\glxtrMathItalicPi
13359 \string;\glxtrUpPi
13360 \string<\glxtrMathItalicRho
13361 \string;\glxtrUpRho
13362 \string<\glxtrMathItalicSigma
13363 \string;\glxtrUpSigma
13364 \string<\glxtrMathItalicTau
13365 \string;\glxtrUpTau
13366 \string<\glxtrMathItalicUpsilon
13367 \string;\glxtrUpUpsilon
13368 \string<\glxtrMathItalicPhi
13369 \string;\glxtrUpPhi
13370 \string<\glxtrMathItalicChi
13371 \string;\glxtrUpChi
13372 \string<\glxtrMathItalicPsi
13373 \string;\glxtrUpPsi
13374 \string<\glxtrMathItalicOmega
13375 \string;\glxtrUpOmega
13376 }

```

`\glxtrUpAlpha`

```

13377 \newcommand*{\glxtrUpAlpha}{%
13378 \glshex 03B1,% lower case alpha
13379 \glshex 0391% upper case alpha

```


13380 }

\backslash glxtrUpBeta

13381 \backslash newcommand* $\{\backslash$ glxtrUpBeta $\}{\%$
13382 \backslash glshex 03B2,% lower case beta
13383 \backslash glshex 0392% upper case beta
13384 }

\backslash glxtrUpGamma

13385 \backslash newcommand* $\{\backslash$ glxtrUpGamma $\}{\%$
13386 \backslash glshex 03B3,% lower case gamma
13387 \backslash glshex 0393% upper case gamma
13388 }

\backslash glxtrUpDelta

13389 \backslash newcommand* $\{\backslash$ glxtrUpDelta $\}{\%$
13390 \backslash glshex 03B4,% lower case delta
13391 \backslash glshex 0394% upper case delta
13392 }

glxtrUpEpsilon

13393 \backslash newcommand* $\{\backslash$ glxtrUpEpsilon $\}{\%$
13394 \backslash glshex 03B5% lower case epsilon
13395 \backslash string= \backslash glshex 03F5,% lower case epsilon variant
13396 \backslash glshex 0395% upper case epsilon
13397 }

glxtrUpDigamma

13398 \backslash newcommand* $\{\backslash$ glxtrUpDigamma $\}{\%$
13399 \backslash glshex 03DD,% lower case digamma
13400 \backslash glshex 03DC% upper case digamma
13401 }

\backslash glxtrUpZeta

13402 \backslash newcommand* $\{\backslash$ glxtrUpZeta $\}{\%$
13403 \backslash glshex 03B6,% lower case zeta
13404 \backslash glshex 0396% upper case zeta
13405 }

\backslash glxtrUpEta

13406 \backslash newcommand* $\{\backslash$ glxtrUpEta $\}{\%$
13407 \backslash glshex 03B7,% lower case eta
13408 \backslash glshex 0397% upper case eta
13409 }

\backslash glxtrUpTheta

13410 \backslash newcommand* $\{\backslash$ glxtrUpTheta $\}{\%$
13411 \backslash glshex 03B8% lower case theta

```
13412 \string=\glshex 03D1,% lower case theta variant
13413 \glshex 0398% upper case theta
13414 }
```

\backslash glxtrUpIota

```
13415 \newcommand*{\glxtrUpIota}{%
13416 \glshex 03B9,% lower case iota
13417 \glshex 0399% upper case iota
13418 }
```

\backslash glxtrUpKappa

```
13419 \newcommand*{\glxtrUpKappa}{%
13420 \glshex 03BA% lower case kappa
13421 \string=\glshex 03F0,% lower case kappa variant
13422 \glshex 039A% upper case kappa
13423 }
```

\backslash glxtrUpLambda

```
13424 \newcommand*{\glxtrUpLambda}{%
13425 \glshex 03BB,% lower lambda
13426 \glshex 039B% upper case lambda
13427 }
```

\backslash glxtrUpMu

```
13428 \newcommand*{\glxtrUpMu}{%
13429 \glshex 03BC,% lower case mu
13430 \glshex 039C% upper case mu
13431 }
```

\backslash glxtrUpNu

```
13432 \newcommand*{\glxtrUpNu}{%
13433 \glshex 03BD,% lower case nu
13434 \glshex 039D% upper case nu
13435 }
```

\backslash glxtrUpXi

```
13436 \newcommand*{\glxtrUpXi}{%
13437 \glshex 03BE,% lower case xi
13438 \glshex 039E% upper case xi
13439 }
```

\backslash glxtrUpOmicron

```
13440 \newcommand*{\glxtrUpOmicron}{%
13441 \glshex 03BF,% lower case omicron
13442 \glshex 039F% upper case omicron
13443 }
```

`\glxtrUpPi`

```
13444 \newcommand*{\glxtrUpPi}{%
13445 \glshex 03C0% lower case pi
13446 \string=\glshex 03D6,% lower case pi variant
13447 \glshex 03A0% upper case pi
13448 }
```

`\glxtrUpRho`

```
13449 \newcommand*{\glxtrUpRho}{%
13450 \glshex 03C1% lower case rho
13451 \string=\glshex 03F1,% lower case rho variant
13452 \glshex 03A1% upper case rho
13453 }
```

`\glxtrUpSigma`

```
13454 \newcommand*{\glxtrUpSigma}{%
13455 \glshex 03C2% lower case sigma
13456 \string=\glshex 03C3,% lower case sigma
13457 \glshex 03A3% upper case sigma
13458 }
```

`\glxtrUpTau`

```
13459 \newcommand*{\glxtrUpTau}{%
13460 \glshex 03C4,% lower case tau
13461 \glshex 03A4% upper case tau
13462 }
```

`\glxtrUpUpsilon`

```
13463 \newcommand*{\glxtrUpUpsilon}{%
13464 \glshex 03C5,% lower case upsilon
13465 \glshex 03A5% upper case upsilon
13466 }
```

`\glxtrUpPhi`

```
13467 \newcommand*{\glxtrUpPhi}{%
13468 \glshex 03C6% lower case phi
13469 \string=\glshex 03D5,% lower case phi variant
13470 \glshex 03A6% upper case phi
13471 }
```

`\glxtrUpChi`

```
13472 \newcommand*{\glxtrUpChi}{%
13473 \glshex 03C7,% lower case chi
13474 \glshex 03A7% upper case chi
13475 }
```

`\glxtrUpPsi`

```
13476 \newcommand*{\glxtrUpPsi}{%
```

```
13477 \glshex 03C8,% lower case psi
13478 \glshex 03A8% upper case psi
13479 }
```

`\glsxtrUpOmega`

```
13480 \newcommand*{\glsxtrUpOmega}{%
13481 \glshex 03C9,% lower case omega
13482 \glshex 03A9% upper case omega
13483 }
```

`MathItalicAlpha`

```
13484 \newcommand*{\glsxtrMathItalicAlpha}{%
13485 \glshex 1D6FC,% lower case alpha (maths italic)
13486 \glshex 1D6E2% upper case alpha (maths italic)
13487 }
```

`rMathItalicBeta`

```
13488 \newcommand*{\glsxtrMathItalicBeta}{%
13489 \glshex 1D6FD,% lower case beta (maths italic)
13490 \glshex 1D6E3% upper case beta (maths italic)
13491 }
```

`MathItalicGamma`

```
13492 \newcommand*{\glsxtrMathItalicGamma}{%
13493 \glshex 1D6FE,% lower case gamma (maths italic)
13494 \glshex 1D6E4% upper case gamma (maths italic)
13495 }
```

`MathItalicDelta`

```
13496 \newcommand*{\glsxtrMathItalicDelta}{%
13497 \glshex 1D6FF,% lower case delta (maths italic)
13498 \glshex 1D6E5% upper case delta (maths italic)
13499 }
```

`thItalicEpsilon`

```
13500 \newcommand*{\glsxtrMathItalicEpsilon}{%
13501 \glshex 1D700% lower case epsilon (maths italic)
13502 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
13503 \glshex 1D6E6% upper case epsilon (maths italic)
13504 }
```

`rMathItalicZeta`

```
13505 \newcommand*{\glsxtrMathItalicZeta}{%
13506 \glshex 1D701,% lower case zeta (maths italic)
13507 \glshex 1D6E7% upper case zeta (maths italic)
13508 }
```

trMathItalicEta

```
13509 \newcommand*{\glxtrMathItalicEta}{%
13510 \glshex 1D702,% lower case eta (maths italic)
13511 \glshex 1D6E8% upper case eta (maths italic)
13512 }
```

MathItalicTheta

```
13513 \newcommand*{\glxtrMathItalicTheta}{%
13514 \glshex 1D703% lower case theta (maths italic)
13515 \string=\glshex 1D717,% lower case theta variant (maths italic)
13516 \glshex 1D6E9% upper case theta (maths italic)
13517 \string=\glshex 1D6F3% upper case theta variant (maths italic)
13518 }
```

rMathItalicIota

```
13519 \newcommand*{\glxtrMathItalicIota}{%
13520 \glshex 1D704,% lower case iota (maths italic)
13521 \glshex 1D6EA% upper case iota (maths italic)
13522 }
```

MathItalicKappa

```
13523 \newcommand*{\glxtrMathItalicKappa}{%
13524 \glshex 1D705% lower case kappa (maths italic)
13525 \string=\glshex 1D718,% lower case kappa variant (maths italic)
13526 \glshex 1D6EB% upper case kappa (maths italic)
13527 }
```

athItalicLambda

```
13528 \newcommand*{\glxtrMathItalicLambda}{%
13529 \glshex 1D706,% lower case lambda (maths italic)
13530 \glshex 1D6EC% upper case lambda (maths italic)
13531 }
```

xtrMathItalicMu

```
13532 \newcommand*{\glxtrMathItalicMu}{%
13533 \glshex 1D707,% lower case mu (maths italic)
13534 \glshex 1D6ED% upper case mu (maths italic)
13535 }
```

xtrMathItalicNu

```
13536 \newcommand*{\glxtrMathItalicNu}{%
13537 \glshex 1D708,% lower case nu (maths italic)
13538 \glshex 1D6EE% upper case nu (maths italic)
13539 }
```

xtrMathItalicXi

```
13540 \newcommand*{\glxtrMathItalicXi}{%
13541 \glshex 1D709,% lower case xi (maths italic)
```

```
13542 \glshex 1D6EF% upper case xi (maths italic)
13543 }
```

thItalicOmicron

```
13544 \newcommand*{\glsxtrMathItalicOmicron}{%
13545 \glshex 1D70A,% lower case omicron (maths italic)
13546 \glshex 1D6F0% upper case omicron (maths italic)
13547 }
```

xtrMathItalicPi

```
13548 \newcommand*{\glsxtrMathItalicPi}{%
13549 \glshex 1D70B% lower case pi (maths italic)
13550 \string=\glshex 1D71B,% lower case pi variant (maths italic)
13551 \glshex 1D6F1% upper case pi (maths italic)
13552 }
```

trMathItalicRho

```
13553 \newcommand*{\glsxtrMathItalicRho}{%
13554 \glshex 1D70C% lower case rho (maths italic)
13555 \string=\glshex 1D71A,% lower case rho variant (maths italic)
13556 \glshex 1D6F2% upper case rho (maths italic)
13557 }
```

MathItalicSigma

```
13558 \newcommand*{\glsxtrMathItalicSigma}{%
13559 \glshex 1D70D% lower case final sigma (maths italic)
13560 \string=\glshex 1D70E,% lower case sigma (maths italic)
13561 \glshex 1D6F4% upper case sigma (maths italic)
13562 }
```

trMathItalicTau

```
13563 \newcommand*{\glsxtrMathItalicTau}{%
13564 \glshex 1D70F,% lower case tau (maths italic)
13565 \glshex 1D6F5% upper case tau (maths italic)
13566 }
```

thItalicUpsilon

```
13567 \newcommand*{\glsxtrMathItalicUpsilon}{%
13568 \glshex 1D710,% lower case upsilon (maths italic)
13569 \glshex 1D6F6% upper case upsilon (maths italic)
13570 }
```

trMathItalicPhi

```
13571 \newcommand*{\glsxtrMathItalicPhi}{%
13572 \glshex 1D711% lower case phi (maths italic)
13573 \string=\glshex 1D719,% lower case phi variant (maths italic)
13574 \glshex 1D6F7% upper case phi (maths italic)
13575 }
```

trMathItalicChi

```
13576 \newcommand*{\glxtrMathItalicChi}{%
13577 \glshex 1D712,% lower case chi (maths italic)
13578 \glshex 1D6F8% upper case chi (maths italic)
13579 }
```

trMathItalicPsi

```
13580 \newcommand*{\glxtrMathItalicPsi}{%
13581 \glshex 1D713,% lower case psi (maths italic)
13582 \glshex 1D6F9% upper case psi (maths italic)
13583 }
```

MathItalicOmega

```
13584 \newcommand*{\glxtrMathItalicOmega}{%
13585 \glshex 1D714,% lower case omega (maths italic)
13586 \glshex 1D6FA% upper case omega (maths italic)
13587 }
```

thItalicPartial

```
13588 \newcommand*{\glxtrMathItalicPartial}{%
13589 \glshex 1D715% partial differential (maths italic)
13590 }
```

MathItalicNabla

```
13591 \newcommand*{\glxtrMathItalicNabla}{%
13592 \glshex 1D6FB% nabla (maths italic)
13593 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
13594 \newcommand*{\glxtrdigitrules}{%
13595 0\string=\glshex 2080\string=\glshex 2070
13596 \string<1\string=\glshex 2081\string=\glshex 00B9
13597 \string<2\string=\glshex 2082\string=\glshex 00B2
13598 \string<3\string=\glshex 2083\string=\glshex 00B3
13599 \string<4\string=\glshex 2084\string=\glshex 2074
13600 \string<5\string=\glshex 2085\string=\glshex 2075
13601 \string<6\string=\glshex 2086\string=\glshex 2076
13602 \string<7\string=\glshex 2087\string=\glshex 2077
13603 \string<8\string=\glshex 2088\string=\glshex 2078
13604 \string<9\string=\glshex 2089\string=\glshex 2079
13605 }
```

BasicDigitrules Digits from the Basic Latin set.

```
13606 \newcommand*{\glxtrBasicDigitrules}{%
13607 0\string<1\string<2\string<3\string<4%
13608 \string<5\string<6\string<7\string<8\string<9%
13609 }
```

criptDigitrules Subscript digits.

```
13610 \newcommand*{\glxtrSubScriptDigitrules}{%
13611 \glshex 2080% subscript 0
13612 \string<\glshex 2081% subscript 1
13613 \string<\glshex 2082% subscript 2
13614 \string<\glshex 2083% subscript 3
13615 \string<\glshex 2084% subscript 4
13616 \string<\glshex 2085% subscript 5
13617 \string<\glshex 2086% subscript 6
13618 \string<\glshex 2087% subscript 7
13619 \string<\glshex 2088% subscript 8
13620 \string<\glshex 2089% subscript 9
13621 }
```

criptDigitrules Superscript digits.

```
13622 \newcommand*{\glxtrSuperScriptDigitrules}{%
13623 \glshex 2070% superscript 0
13624 \string<\glshex 00B9% superscript 1
13625 \string<\glshex 00B2% superscript 2
13626 \string<\glshex 00B3% superscript 3
13627 \string<\glshex 2074% superscript 4
13628 \string<\glshex 2075% superscript 5
13629 \string<\glshex 2076% superscript 6
13630 \string<\glshex 2077% superscript 7
13631 \string<\glshex 2078% superscript 8
13632 \string<\glshex 2079% superscript 9
13633 }
```

trfractionrules Vulgar fractions.

```
13634 \newcommand*{\glxtrfractionrules}{%
13635 \glshex 215F% fraction numerator one (1/)
13636 \string<\glshex 2189% zero thirds (0/3 = 0)
13637 \string<\glshex 2152% one tenth (1/10 = 0.1)
13638 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
13639 \string<\glshex 215B% one eighth (1/8 = 0.125)
13640 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
13641 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
13642 \string<\glshex 2155% one fifth (1/5 = 0.2)
13643 \string<\glshex 00BC% one quarter (1/4 = 0.25)
13644 \string<\glshex 2153% one third (1/3 ~ 0.333)
13645 \string<\glshex 215C% three eighths (3/8 = 0.375)
13646 \string<\glshex 2156% two fifths (2/5 = 0.4)
13647 \string<\glshex 00BD% one half (1/2 = 0.5)
13648 \string<\glshex 2157% three fifths (3/5 = 0.6)
13649 \string<\glshex 215D% five eighths (5/8 = 0.625)
13650 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
13651 \string<\glshex 00BE% three quarters (3/4 = 0.75)
13652 \string<\glshex 2158% four fifths (4/5 = 0.8)
13653 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
```



```

13654 \string<\glshex 215E% seven eighths (7/8 = 0.875)
13655 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

13656 \renewcommand{\@glsextrdialecthook}{%
13657   \ifundef\CurrentTrackedScript
13658   {%
13659     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13660     {%
13661       \edef\CurrentTrackedScript{%
13662         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13663     }%
13664   }%
13665 }%
13666 {}%
13667 \ifdef\CurrentTrackedScript
13668 {%
13669   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
13670   \def\TrackLangRequireDialectPrefix{glossariesextr-}%
13671   \let\CurrentTrackedTag\CurrentTrackedScript
13672   \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.1df}
13673   {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13674   {}%
13675   \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13676 }%
13677 {}%
13678 }

```

If `\glsextr@loaddialect` has been defined, then `glossaries-extra-bib2gls` has been loaded after `glossaries-extra`. (For example, through `\glossariesextrasetup`.) Not recommended, but if this has been done try to find the associated language resources.

```

13679 \ifdef\glsextr@loaddialect
13680 {%
13681   \@ifpackageloaded{tracklang}
13682   {%
13683     \AnyTrackedLanguages
13684     {%
13685       \ForEachTrackedDialect{\this@dialect}{\glsextr@loaddialect}%
13686     }%
13687   }%
13688 }
13689 {}
13690 }
13691 {}

```

2 Style Adjustments (glossaries-extra-stylemods.sty)

This package adjusts the predefined styles so that they include the post description hook. Also, some other minor adjustments may be made to make existing styles more flexible.

2.1 Package Initialisation

First identify package:

```
13692 \NeedsTeXFormat{LaTeX2e}
13693 \ProvidesPackage{glossaries-extra-stylemods}[2019/04/09 v1.41 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-<option>.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glsxtr@loadstyles`.

```
sxtr@loadstyles
```

```
13694 \newcommand*{\@glsxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13695 \DeclareOption{all}{%
13696   \appto\@glsxtr@loadstyles{%
13697     \RequirePackage{glossary-inline}%
13698     \RequirePackage{glossary-list}%
13699     \RequirePackage{glossary-tree}%
13700     \RequirePackage{glossary-mcols}%
13701     \RequirePackage{glossary-long}%
13702     \RequirePackage{glossary-longragged}%
13703     \RequirePackage{glossary-longbooktabs}%
13704     \RequirePackage{glossary-super}%
13705     \RequirePackage{glossary-superragged}%
13706     \RequirePackage{glossary-bookindex}%
13707     \RequirePackage{glossary-longextra}%
13708     \RequirePackage{glossary-topic}%
13709   }
13710 }

13711 \DeclareOption*{%
13712   \IfFileExists{glossary-\CurrentOption.sty}
13713   {\eappto\@glsxtr@loadstyles{%
13714     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13715   }%
```

```

13716  {%
13717    \PackageError{glossaries-extra-styles}%
13718    {Unknown option ‘\CurrentOption’}{}%
13719  }%
13720 }

```

Process the package options:

```
13721 \ProcessOptions
```

Load the required packages:

```
13722 \@glsxtr@loadstyles
```

Adjust the styles so that they all have the post description hook. Also, instead of having a hard-coded `\space` before the location, use:

`sxtrprelocation` This uses `\providecommand` as the same command is also provided by `glossary-bookindex`.

```
13723 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of `glossaries`:

`ewglossarystyle`

```

13724 \providecommand{\renewglossarystyle}[2]{%
13725   \ifcsundef{@glsstyle@#1}%
13726   {%
13727     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13728   }%
13729   {%
13730     \csdef{@glsstyle@#1}{#2}%
13731   }%
13732 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the `listdotted` style need modifying to add this.

```

13733 \ifdef{\@glsstyle@listdotted}
13734 {%
13735   \renewglossarystyle{listdotted}{%
13736     \setglossarystyle{list}%
13737     \renewcommand*{\glossentry}[2]{%
13738       \item[]\makebox[\glslistdottedwidth][l]{%
13739         \glstarget{##1}{\glossentryname{##1}}%
13740         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13741         \glossentrydesc{##1}\glspostdescription}%
13742     \renewcommand*{\subglossentry}[3]{%
13743       \item[]\makebox[\glslistdottedwidth][l]{%
13744         \glssubentryitem{##2}%
13745         \glstarget{##2}{\glossentryname{##2}}%
13746         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%

```

```

13748   \glossentrydesc{##2}\glspostdescription}%
13749 }
13750 }
13751 {%

```

Assume the style isn't required if it hasn't already been defined.

```
13752 }
```

The sublistedotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

The other list styles would be easier to adapt if the space before the number list wasn't hard coded.

```

13753 \ifdef{\@glsstyle@list}
13754 {%

```

listprelocation Space before number list for top-level entries.

```
13755 \newcommand{\glslistprelocation}{\glsxtrprelocation}
```

childprelocation Space before number list for child entries.

```
13756 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13757 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13758 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

glslistgroupskip

```
13759 \newcommand{\glslistgroupskip}{\nobreak\indexspace\nobreak}
```

Redefine list to use these commands.

```

13760 \renewglossarystyle{list}{%
13761   \renewenvironment{theglossary}%
13762     {\begin{description}}{\end{description}}%
13763   \renewcommand*\glossaryheader{}%
13764   \renewcommand*\glsgroupheading}[1]{}%
13765   \renewcommand*\glossentry}[2]{%
13766     \item[\glsentryitem{##1}]%
13767       \glstarget{##1}{\glossentryname{##1}}]
13768     \glslistdesc{##1}\glslistprelocation ##2}%
13769   \renewcommand*\subglossentry}[3]{%
13770     \glssubentryitem{##2}%
13771     \glstarget{##2}{\strut}\space
13772     \glslistdesc{##2}%
13773     \glslistchildprelocation ##3\glslistchildpostlocation}%
13774   \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glslistgroupskip\fi}%
13775 }
13776 }
13777 {}

```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```

13778 \ifdef{\@glsstyle@altlist}
13779 {%
13780   \renewglossarystyle{altlist}{%
13781     \setglossarystyle{list}%
13782     \renewcommand*\glossentry}[2]{%
13783       \item[\glsentryitem{##1}]%
13784         \glstarget{##1}{\glossentryname{##1}}}%
13785     \mbox{}\par\nobreak\@afterheading
13786     \glslistdesc{##1}\glslistprelocation ##2}%
13787   \renewcommand*\subglossentry}[3]{%
13788     \par
13789     \glssubentryitem{##2}%
13790     \glstarget{##2}{\strut}\glslistdesc{##2}%
13791     \glslistchildprelocation ##3}%
13792   }
13793 }
13794 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13795 \ifdef{\@glsstyle@listgroup}
13796 {%
13797   \renewglossarystyle{listgroup}{%
13798     \setglossarystyle{list}%
13799     \renewcommand*\glsgroupheading}[1]{%
13800       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}%
13801     \mbox{}\par\nobreak\@afterheading
13802   }%
13803 }
13804 }
13805 {}

```

Similarly for listhypergroup.

```

13806 \ifdef{\@glsstyle@listhypergroup}
13807 {%
13808   \renewglossarystyle{listhypergroup}{%
13809     \setglossarystyle{list}%
13810     \renewcommand*\glossaryheader}{%
13811       \glslistnavigationitem{\glsnavigation}}%
13812     \renewcommand*\glsgroupheading}[1]{%
13813       \item[\glslistgroupheaderfmt
13814         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}%
13815       \mbox{}\par\nobreak\@afterheading
13816     }%
13817   }
13818 }
13819 {}

```

Similarly for altlistgroup.

```

13820 \ifdef{\@glsstyle@altlistgroup}

```

```

13821 {%
13822   \renewglossarystyle{altlistgroup}{%
13823     \setglossarystyle{altlist}%
13824     \renewcommand*{\glsgroupheading}[1]{%
13825       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]}%
13826     \mbox{}\par\nobreak\@afterheading
13827   }%
13828 }
13829 }
13830 {}

```

Similarly for altlisthypergroup.

```

13831 \ifdef{\@glsstyle@altlisthypergroup}
13832 {%
13833   \renewglossarystyle{altlisthypergroup}{%
13834     \setglossarystyle{altlist}%
13835     \renewcommand*{\glossaryheader}{%
13836       \glslistnavigationitem{\glsnavigation}}}%
13837   \renewcommand*{\glsgroupheading}[1]{%
13838     \item[\glslistgroupheaderfmt
13839       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}%
13840     \mbox{}\par\nobreak\@afterheading
13841   }%
13842 }
13843 }
13844 {}

```

2.3 Longtable Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glxtrprelocation`.

```

13845 \ifcsdef{@glsstyle@long}
13846 {%
13847   \renewglossarystyle{long}{%
13848     \renewenvironment{theglossary}%
13849       {\begin{longtable}[lp{\glsdescwidth}}}%
13850       {\end{longtable}}}%
13851   \renewcommand*{\glossaryheader}{}%
13852   \renewcommand*{\glsgroupheading}[1]{}%
13853   \renewcommand{\glossentry}[2]{%
13854     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13855     \glossentrydesc{##1}\glspostdescription
13856     \glxtrprelocation ##2\tabularnewline
13857   }%
13858   \renewcommand{\subglossentry}[3]{%
13859     &
13860     \glssubentryitem{##2}%
13861     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription

```

```

13862     \glstrprelocation ##3\tabularnewline
13863 }%
13864 \ifglsnogroupskip
13865     \renewcommand*{\glsgroupskip}{}%
13866 \else
13867     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13868 \fi
13869 }
13870 }
13871 {}

```

Three column style:

```

13872 \ifcsdef{@glsstyle@long3col}
13873 {%
13874     \renewglossarystyle{long3col}{%
13875         \renewenvironment{theglossary}%
13876             {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13877             {\end{longtable}}}%
13878     \renewcommand*{\glossaryheader}{}%
13879     \renewcommand*{\glsgroupheading}[1]{}%
13880     \renewcommand{\glossentry}[2]{%
13881         \glstarget{##1}\glstarget{##1}{\glossentryname{##1}} &
13882         \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13883     }%
13884     \renewcommand{\subglossentry}[3]{%
13885         &
13886         \glssubentryitem{##2}%
13887         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13888         ##3\tabularnewline
13889     }%

```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```

13890     \ifglsnogroupskip
13891         \renewcommand*{\glsgroupskip}{}%
13892     \else
13893         \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13894     \fi
13895 }
13896 }
13897 {}

```

Four column style:

```

13898 \ifcsdef{@glsstyle@long4col}
13899 {%
13900     \renewglossarystyle{long4col}{%
13901         \renewenvironment{theglossary}%
13902             {\begin{longtable}{llll}}}%
13903             {\end{longtable}}}%
13904     \renewcommand*{\glossaryheader}{}%
13905     \renewcommand*{\glsgroupheading}[1]{}%

```

```

13906 \renewcommand{\glossentry}[2]{%
13907   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13908   \glossentrydesc{##1}\glspostdescription &
13909   \glossentrysymbol{##1} &
13910   ##2\tabularnewline
13911 }%
13912 \renewcommand{\subglossentry}[3]{%
13913   &
13914   \glssubentryitem{##2}%
13915   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13916   \glossentrysymbol{##2} & ##3\tabularnewline
13917 }%

13918 \ifglsgroupskip
13919   \renewcommand*\{glsgroupskip}{}%
13920 \else
13921   \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13922 \fi
13923 }
13924 }
13925 {}

```

The styles in glossary-longbooktabs are all based on the styles in glossary-long, so no adjustments are needed for that package.

2.4 Long Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glxtrprelocation`.

```

13926 \ifcsdef{@glstyle@longragged}
13927 {%
13928   \renewglossarystyle{longragged}{%
13929     \renewenvironment{theglossary}%
13930       {\begin{longtable}[l>{\raggedright}p{\glstdescwidth}}}%
13931       {\end{longtable}}%
13932     \renewcommand*\{glossaryheader}{}%
13933     \renewcommand*\{glsgroupheading}[1]{}%
13934     \renewcommand{\glossentry}[2]{%
13935       \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13936       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13937       \tabularnewline
13938     }%
13939     \renewcommand{\subglossentry}[3]{%
13940       &
13941       \glssubentryitem{##2}%
13942       \glstarget{##2}{\strut}\glossentrydesc{##2}%
13943       \glspostdescription\glxtrprelocation ##3%
13944       \tabularnewline

```



```

13945 }%
13946 \ifglsnogroupskip
13947   \renewcommand*{\glsgroupskip}{}%
13948 \else
13949   \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13950 \fi
13951 }
13952 }
13953 {}

```

Three and four column styles don't use `\glsxtrprelocation` since the number list is in its own column.

```

13954 \ifcsdef{@glsstyle@longragged3col}
13955 {%
13956   \renewglossarystyle{longragged3col}{%
13957     \renewenvironment{theglossary}%
13958       {\begin{longtable}[l>{\raggedright}p{\glsdescwidth}%
13959         >{\raggedright}p{\glspagelistwidth}}}%
13960       {\end{longtable}}}%
13961   \renewcommand*{\glossaryheader}{}%
13962   \renewcommand*{\glsgroupheading}[1]{}%
13963   \renewcommand{\glossentry}[2]{%
13964     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13965     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13966   }%
13967   \renewcommand{\subglossentry}[3]{%
13968     &
13969     \glssubentryitem{##2}%
13970     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13971     ##3\tabularnewline
13972   }%
13973   \ifglsnogroupskip
13974     \renewcommand*{\glsgroupskip}{}%
13975   \else
13976     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13977   \fi
13978 }
13979 }
13980 {}

```

Four column style:

```

13981 \ifcsdef{@glsstyle@altlongragged4col}
13982 {%
13983   \renewglossarystyle{altlongragged4col}{%
13984     \renewenvironment{theglossary}%
13985       {\begin{longtable}[l>{\raggedright}p{\glsdescwidth}l%
13986         >{\raggedright}p{\glspagelistwidth}}}%
13987       {\end{longtable}}}%
13988   \renewcommand*{\glossaryheader}{}%

```

```

13989 \renewcommand*\glsgroupheading}[1]{}%
13990 \renewcommand{\glossentry}[2]{%
13991   \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13992   \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13993   ##2\tabularnewline
13994 }%
13995 \renewcommand{\subglossentry}[3]{%
13996   &
13997   \glssubentryitem{##2}%
13998   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13999   \glossentrysymbol{##2} & ##3\tabularnewline
14000 }%

14001 \ifglsgroupskip
14002   \renewcommand*\glsgroupskip}{}%
14003 \else
14004   \renewcommand*\glsgroupskip}{& & \tabularnewline}%
14005 \fi
14006 }
14007 }
14008 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glstrprelocation`.

```

14009 \ifcsdef{@glstyle@super}
14010 {%
14011   \renewglossarystyle{super}{%
14012     \renewenvironment{theglossary}%
14013       {\tablehead{}}\tabletail{}}%
14014     \begin{supertabular}{lp{\glstdescwidth}}%
14015     {\end{supertabular}}%
14016   \renewcommand*\glossaryheader}{}%
14017   \renewcommand*\glsgroupheading}[1]{}%
14018   \renewcommand{\glossentry}[2]{%
14019     \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14020     \glossentrydesc{##1}\glspostdescription
14021     \glstrprelocation ##2\tabularnewline
14022   }%
14023   \renewcommand{\subglossentry}[3]{%
14024     &
14025     \glssubentryitem{##2}%
14026     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
14027     \glstrprelocation ##3\tabularnewline
14028   }%
14029   \ifglsgroupskip
14030     \renewcommand*\glsgroupskip}{}%

```

```

14031 \else
14032 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14033 \fi
14034 }
14035 }
14036 {}

```

Three column style:

```

14037 \ifcsdef{@glsstyle@super3col}
14038 {%
14039 \renewglossarystyle{super3col}{%
14040 \renewenvironment{theglossary}%
14041 {\tablehead{}}\tabletail{}}%
14042 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
14043 {\end{supertabular}}%
14044 \renewcommand*{\glossaryheader}{}%
14045 \renewcommand*{\glsgroupheading}[1]{}%
14046 \renewcommand{\glossentry}[2]{%
14047 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14048 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14049 }%
14050 \renewcommand{\subglossentry}[3]{%
14051 &
14052 \glssubentryitem{##2}%
14053 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14054 ##3\tabularnewline
14055 }%
14056 \ifglsnogroupskip
14057 \renewcommand*{\glsgroupskip}{}%
14058 \else
14059 \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14060 \fi
14061 }
14062 }
14063 {}

```

Four column styles:

```

14064 \ifcsdef{@glsstyle@super4col}
14065 {%
14066 \renewglossarystyle{super4col}{%
14067 \renewenvironment{theglossary}%
14068 {\tablehead{}}\tabletail{}}%
14069 \begin{supertabular}{llll}}{%
14070 \end{supertabular}}%
14071 \renewcommand*{\glossaryheader}{}%
14072 \renewcommand*{\glsgroupheading}[1]{}%
14073 \renewcommand{\glossentry}[2]{%
14074 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14075 \glossentrydesc{##1}\glspostdescription &

```

```

14076     \glossentrysymbol{##1} & ##2\tabularnewline
14077 }%
14078 \renewcommand{\subglossentry}[3]{%
14079     &
14080     \glssubentryitem{##2}%
14081     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14082     \glossentrysymbol{##2} & ##3\tabularnewline
14083 }%

14084 \ifglsgroupskip
14085     \renewcommand*\{glsgroupskip}\{}%
14086 \else
14087     \renewcommand*\{glsgroupskip}\{& & \tabularnewline}%
14088 \fi
14089 }
14090 }
14091 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glstrprelocation`.

```

14092 \ifcsdef{@glsstyle@superragged}
14093 {%
14094     \renewglossarystyle{superragged}{%
14095         \renewenvironment{theglossary}%
14096             {\tablehead{}\tabletail}%
14097             \begin{supertabular}[1>\raggedright]p{\glsdescwidth}}%
14098             {\end{supertabular}}%
14099     \renewcommand*\{glossaryheader}\{}%
14100     \renewcommand*\{glsgroupheading}[1]\{}%
14101     \renewcommand{\glossentry}[2]{%
14102         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14103         \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
14104         \tabularnewline
14105     }%
14106     \renewcommand{\subglossentry}[3]{%
14107         &
14108         \glssubentryitem{##2}%
14109         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
14110         \glstrprelocation ##3%
14111         \tabularnewline
14112     }%
14113     \ifglsgroupskip
14114         \renewcommand*\{glsgroupskip}\{}%
14115     \else
14116         \renewcommand*\{glsgroupskip}\{& \tabularnewline}%

```

```

14117   \fi
14118   }
14119 }
14120 {}

```

Three column style:

```

14121 \ifcsdef{@glsstyle@superragged3col}
14122 {%
14123   \renewglossarystyle{superragged3col}{%
14124     \renewenvironment{theglossary}%
14125       {\tablehead{}}\tabletail{}}%
14126     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
14127       >{\raggedright}p{\glspagelistwidth}}}%
14128     {\end{supertabular}}%
14129     \renewcommand*{\glossaryheader}{}%
14130     \renewcommand*{\glsgroupheading}[1]{}%
14131     \renewcommand{\glossentry}[2]{%
14132       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14133       \glossentrydesc{##1}\glspostdescription &
14134       ##2\tabularnewline
14135     }%
14136     \renewcommand{\subglossentry}[3]{%
14137       &
14138       \glssubentryitem{##2}%
14139       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14140       ##3\tabularnewline
14141     }%
14142   \ifglsnogroupskip
14143     \renewcommand*{\glsgroupskip}{}%
14144   \else
14145     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14146   \fi
14147 }
14148 }
14149 {}

```

Four columns:

```

14150 \ifcsdef{@glsstyle@altsuperragged4col}
14151 {%
14152   \renewglossarystyle{altsuperragged4col}{%
14153     \renewenvironment{theglossary}%
14154       {\tablehead{}}\tabletail{}}%
14155     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
14156       >{\raggedright}p{\glspagelistwidth}}}%
14157     {\end{supertabular}}%
14158     \renewcommand*{\glossaryheader}{}%
14159     \renewcommand{\glossentry}[2]{%
14160       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14161       \glossentrydesc{##1}\glspostdescription &

```

```

14162     \glossentrysymbol{##1} & ##2\tabularnewline
14163 }%
14164 \renewcommand{\subglossentry}[3]{%
14165     &
14166     \glssubentryitem{##2}%
14167     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14168     \glossentrysymbol{##2} & ##3\tabularnewline
14169 }%

14170 \ifglsgroupskip
14171     \renewcommand*\{glsgroupskip}{}%
14172 \else
14173     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
14174 \fi
14175 }
14176 }
14177 {}

```

2.7 Inline Style

The inline style is dealt with slightly differently. The `\glspostdescription` hook is actually in `\glspostinline`, which is called at the end of the glossary. The original definition of `\glspostinline` also includes a space, which is unnecessary. Here, instead of redefining the inline style, just redefine `\glspostinline` and `\glsinlinedescformat`.

```

14178 \ifdef{\@glsstyle@inline}
14179 {%
14180     \renewcommand*\{glspostinline}{.\spacefactor\sfcode{.}}
14181     Just use \glsxtrpostdescription instead of \glspostdescription.
14182     \renewcommand*\{glsinlinedescformat}[3]{%
14183         \space#1\glsxtrpostdescription}
14184     \renewcommand*\{glsinlinesubdescformat}[3]{%
14185         #1\glsxtrpostdescription}

```

The default settings don't show the location lists, so there's no adjustment for `\glsxtrprelocation`.

```

14185 }
14186 {}

```

2.8 Tree Styles

Redefine both `\glstreenamefmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamefmt` to make it easier to change both at the same time or only change one without affecting the other.

```

14187 \ifdef\glstreenamefmt
14188 {
\glstreedefaultnamefmt
14189     \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}

```

`\glstreenamefmt`

```
14190 \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}
```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14191 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreenavigationfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14192 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreePreHeader` Takes the label as the first argument and title as the second argument so this can be modified to add a bookmark.

```
14193 \newcommand{\glstreePreHeader}[2]{}

14194 }
14195 }
```

The index style is redefined so that the space before the number list isn't hard coded.

```
14196 \ifdef{\@glsstyle@index}
```

```
14197 {
```

`\glstreeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```
14198 \newcommand*{\glstreeprelocation}{\glsxtrprelocation}
```

`\glstreechildprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```
14199 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

`\glstreegroupskip`

```
14200 \newcommand{\glstreegroupskip}{\nopagebreak\indexspace\nobreak}
```

Modify the index style.

```
14201 \renewglossarystyle{index}{%
```

```
14202 \renewenvironment{theglossary}%
```

```
14203 {\setlength{\parindent}{0pt}%
```

```
14204 \setlength{\parskip}{0pt plus 0.3pt}%
```

```
14205 \let\item\glstreeitem
```

```
14206 \let\subitem\glstreesubitem
```

```
14207 \let\subsubitem\glstreesubsubitem
```

```
14208 }%
```

```
14209 {\par}%
```

```
14210 \renewcommand*{\glossaryheader}{}%
```

```
14211 \renewcommand*{\glsgroupheading}[1]{}%
```

```
14212 \renewcommand*{\glossentry}[2]{}%
```

```
14213 \item\glsentryitem{##1}%
```

```
14214 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```
14215 \glstreesymbol{##1}%
```

```
14216 \glstreeDescLoc{##1}{##2}%
```

```
14217 }%
```

```
14218 \renewcommand{\subglossentry}[3]{}%
```

```

14219     \ifcase##1\relax
14220         \item
14221     \or
14222         \subitem
14223         \glssubentryitem{##2}%
14224     \else
14225         \subsubitem
14226     \fi
14227     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
14228     \glstreechildsymbol{##2}%
14229     \glstreeChildDescLoc{##2}{##3}%
14230 }%
14231 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glstreegroupskip\fi}%
14232 }
14233 }
14234 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

14235 \ifdef{\@glsstyle@indexgroup}
14236 {%
14237     \renewglossarystyle{indexgroup}{%
14238         \setglossarystyle{index}%
14239         \renewcommand*\glsgroupheading}[1]{%
14240             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14241             \glstreePreHeader{##1}{\glxtr@grptitle}%
14242             \item\glstreegroupheaderfmt{\glxtr@grptitle}%
14243             \glstreegroupskip\@afterheading
14244         }%
14245     }
14246 }
14247 {}

```

Similarly for indexhypergroup.

```

14248 \ifdef{\@glsstyle@indexhypergroup}
14249 {%
14250     \renewglossarystyle{indexhypergroup}{%
14251         \setglossarystyle{index}%
14252         \renewcommand*\glossaryheader{%
14253             \item\glstreenavigationfmt{\glsnavigation}%
14254             \glstreegroupskip\@afterheading}%
14255         \renewcommand*\glsgroupheading}[1]{%
14256             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14257             \glstreePreHeader{##1}{\glxtr@grptitle}%
14258             \item\glstreegroupheaderfmt
14259                 {\glsnavhypertarget{##1}{\glxtr@grptitle}}%
14260             \glstreegroupskip\@afterheading}%
14261     }%
14262 }
14263 {}

```

Adjust tree style to remove hard coded space before number list.


```
14264 \ifdef{\@glsstyle@tree}
14265 {%
```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```
14266 \newcommand{\glstreedesc}[1]{%
14267   \glstreedesc\glossentrydesc{#1}\glspostdescription
14268 }
```

`\glstreeDescLoc`

```
\glstreeDescLoc{<label>}{<location>}
```

This checks for the description and symbol. If both are missing, don't want a space at the start of the location list. (For example, the entry may just have a cross-reference.)

```
14269 \newcommand{\glstreeDescLoc}[2]{%
14270   \ifglshasdesc{#1}%
14271   {\glstreedesc{#1}\glstreeprelocation}%
14272   {\ifglshassymbol{#1}{\glstreeprelocation}{}}%
14273   #2%
14274 }
```

Similarly for the symbol.

`\glstreesymbol`

```
14275 \newcommand{\glstreesymbol}[1]{%
14276   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}}%
14277 }%
```

And for the child entries:

`\glstreechilddesc`

```
14278 \newcommand{\glstreechilddesc}[1]{%
14279   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
14280 }%
```

`\glstreeChildDescLoc`

```
14281 \newcommand{\glstreeChildDescLoc}[2]{%
14282   \ifglshasdesc{#1}%
14283   {\glstreechilddesc{#1}\glstreechildprelocation}%
14284   {\ifglshassymbol{#1}{\glstreechildprelocation}{}}%
14285   #2%
14286 }%
```

`\glstreechildsymbol` This just behaves in the same way as the top-level.

```
14287 \newcommand{\glstreechildsymbol}[1]{%
14288   \glstreesymbol{#1}%
14289 }%
```

```

14290 \renewglossarystyle{tree}{%
14291   \renewenvironment{theglossary}%
14292     {\setlength{\parindent}{0pt}%
14293     \setlength{\parskip}{0pt plus 0.3pt}}%
14294   }%
14295   \renewcommand*\glossaryheader{}{}%
14296   \renewcommand*\glsgroupheading}[1]{}%
14297   \renewcommand{\glossentry}[2]{%
14298     \hangindent0pt\relax
14299     \parindent0pt\relax
14300     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14301     \glstreesymbol{##1}%
14302     \glstreeDescLoc{##1}{##2}\par
14303   }%
14304   \renewcommand{\subglossentry}[3]{%
14305     \hangindent##1\glstreeindent\relax
14306     \parindent##1\glstreeindent\relax
14307     \ifnum##1=1\relax
14308       \glssubentryitem{##2}%
14309     \fi
14310     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
14311     \glstreechildsymbol{##2}%
14312     \glstreeChildDescLoc{##2}{##3}\par
14313   }%
14314   \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glstreegroupskip\fi}%
14315 }%
14316 }
14317 {}

```

The treegroup style is redefined to discourage a page break after the heading.

```

14318 \ifdef{\@glsstyle@treegroup}
14319 {%
14320   \renewglossarystyle{treegroup}{%
14321     \setglossarystyle{tree}%
14322     \renewcommand{\glsgroupheading}[1]{%
14323       \glstrgetgrouptitle{##1}{\glstr@grptitle}%
14324       \glstreePreHeader{##1}{\glstr@grptitle}%
14325       \par\noindent\glstreegroupheaderfmt{\glstr@grptitle}%
14326       \glstreegroupskip\@afterheading}%
14327   }
14328 }
14329 {}

```

Similarly for treehypergroup

```

14330 \ifdef{\@glsstyle@treehypergroup}
14331 {%
14332   \renewglossarystyle{treehypergroup}{%
14333     \setglossarystyle{tree}%
14334     \renewcommand*\glossaryheader{}{}%
14335     \par\noindent\glstreenavigationfmt{\glsnavigation}%

```

```

14336     \glstreegroupskip\@afterheading}%
14337 \renewcommand*\glsgroupheading}[1]{%
14338     \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14339     \glstreePreHeader{##1}{\glxtr@grptitle}%
14340     \par\noindent
14341     \glstreegroupheaderfmt
14342     {\glsnahypertarget{##1}{\glxtr@grptile}}%
14343     \glstreegroupskip\@afterheading}%
14344 }
14345 }
14346 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

14347 \ifdef{\@glsstyle@treenoname}
14348 {}%

```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```

14349 \newcommand{\glstreenonamedesc}[1]{%
14350     \glstreepredesc\glossentrydesc{##1}\glspostdescription
14351 }%

```

Similarly for the symbol.

treenonamesymbol

```

14352 \newcommand{\glstreenonamesymbol}[1]{%
14353     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}}%
14354 }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

14355 \newcommand{\glstreenonamechilddesc}[1]{%
14356     \glossentrydesc{##1}\glspostdescription
14357 }%

14358 \renewglossarystyle{treenoname}{%
14359     \renewenvironment{theglossary}%
14360     {\setlength{\parindent}{0pt}%
14361     \setlength{\parskip}{0pt plus 0.3pt}}%
14362     {}%
14363     \renewcommand*\glossaryheader{}%
14364     \renewcommand*\glsgroupheading}[1]{}%
14365     \renewcommand{\glossentry}[2]{%
14366         \hangindent0pt\relax
14367         \parindent0pt\relax
14368         \glstreeitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14369         \glstreenonamesymbol{##1}%
14370         \glstreenonamedesc{##1}%
14371         \glstreeprelocation##2\par
14372     }%

```

```

14373 \renewcommand{\subglossentry}[3]{%
14374 \hangindent##1\glstreeindent\relax
14375 \parindent##1\glstreeindent\relax
14376 \ifnum##1=1\relax
14377 \glssubentryitem{##2}%
14378 \fi
14379 \glstarget{##2}{\strut}%
14380 \glstreenonamechilddesc{##2}%
14381 \glstreechildprelocation##3\par
14382 }%
14383 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
14384 }
14385 }
14386 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

14387 \ifdef{\@glsstyle@treenonamegroup}
14388 {%
14389 \renewglossarystyle{treenonamegroup}{%
14390 \setglossarystyle{treenoname}%
14391 \renewcommand{\glsgroupheading}[1]{%
14392 \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14393 \glstreePreHeader{##1}{\glxtr@grptitle}%
14394 \par\noindent\glstreegroupheaderfmt{\glxtr@grptitle}%
14395 \glstreegroupskip\@afterheading
14396 }%
14397 }
14398 }
14399 {}

```

Similarly for `treenonamehypergroup`

```

14400 \ifdef{\@glsstyle@treenonamehypergroup}
14401 {%
14402 \renewglossarystyle{treenonamehypergroup}{%
14403 \setglossarystyle{treenoname}%
14404 \renewcommand*{\glossaryheader}{%
14405 \par\noindent\glstreenavigationfmt{\glsnavigation}%
14406 \glstreegroupskip\@afterheading}%
14407 \renewcommand*{\glsgroupheading}[1]{%
14408 \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14409 \glstreePreHeader{##1}{\glxtr@grptitle}%
14410 \par\noindent
14411 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glxtr@grptitle}}%
14412 \glstreegroupskip\@afterheading}%
14413 }
14414 }
14415 {}

```

The `almtree` style is redefined to make it easier to made minor adjustments.

```

14416 \ifdef{\@glsstyle@almtree}
14417 {%

```

Only redefine this style if it's already been defined.

symbolDescLocation `\glsxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```
14418 \newcommand{\glsxtralttreeSymbolDescLocation}[2]{%
14419   {%
14420     \let\par\glsxtrAltTreePar
14421     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
14422     \ifglshasdesc{#1}%
14423       {\glossentrydesc{#1}\glspostdescription\glstreeprelocation}{}%
14424     #2\par
14425   }%
14426 }
```

trAltTreeIndent Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```
14427 \newlength\glsxtrAltTreeIndent
```

ltxtrAltTreePar Multi-paragraph descriptions need to keep the hanging indent.

```
14428 \newcommand{\glsxtrAltTreePar}{%
14429   \@@par
14430   \glsxtrAltTreeSetHangIndent
14431   \setlength{\parindent}{\dimexpr\hangindent+\glsxtrAltTreeIndent}%
14432 }
```

symbolDescLocation `\glsxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```
14433 \newcommand{\glsxtralttreeSubSymbolDescLocation}[3]{%
14434   \glsxtralttreeSymbolDescLocation{#2}{#3}%
14435 }
```

trtreetopindent The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```
14436 \newlength\glsxtrtreetopindent
```

glsxtralttreeInit User-level initialisation for the alttree style.

```
14437 \newcommand*{\glsxtralttreeInit}{%
14438   \settowidth{\glsxtrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
14439   \glsxtrAltTreeIndent=\parindent
14440 }
```

```

\glssetwidest The original \glssetwidest only uses \def. This uses \gdef.
14441 \newcommand*{\glssetwidest}[2][0]{%
14442   \csgdef{@glswidestname\romannumeral#1}{#2}%
14443 }

\eglssetwidest The original \glssetwidest only uses \def. This uses \protected@csedef.
14444 \newcommand*{\eglssetwidest}[2][0]{%
14445   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14446 }

\xglssetwidest Like the above but uses \protected@csxdef.
14447 \newcommand*{\xglssetwidest}[2][0]{%
14448   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14449 }

glsupdatewidest Only sets if new value is wider than old value.
14450 \newcommand*{\glsupdatewidest}[2][0]{%
14451   \ifcsundef{@glswidestname\romannumeral#1}%
14452   {\csdef{@glswidestname\romannumeral#1}{#2}}%
14453   {%
14454     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14455     \settowidth{\dimen@ii}{#2}%
14456     \ifdim\dimen@ii>\dimen@
14457       \csdef{@glswidestname\romannumeral#1}{#2}%
14458     \fi
14459   }%
14460 }

glsupdatewidest As above but global definition.
14461 \newcommand*{\gglsupdatewidest}[2][0]{%
14462   \ifcsundef{@glswidestname\romannumeral#1}%
14463   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
14464   {%
14465     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14466     \settowidth{\dimen@ii}{#2}%
14467     \ifdim\dimen@ii>\dimen@
14468       \csgdef{@glswidestname\romannumeral#1}{#2}%
14469     \fi
14470   }%
14471 }

glsupdatewidest As \glsupdatewidest but expands value.
14472 \newcommand*{\eglsupdatewidest}[2][0]{%
14473   \ifcsundef{@glswidestname\romannumeral#1}%
14474   {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
14475   {%
14476     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14477     \settowidth{\dimen@ii}{#2}%

```

```

14478     \ifdim\dimen@ii>\dimen@
14479     \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14480     \fi
14481     }%
14482     }

```

`glupdatewidest` As above but global.

```

14483 \newcommand*\xglupdatewidest}[2][0]{%
14484 \ifcsundef{@glswidestname\romannumeral#1}%
14485 {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
14486 {%
14487 \settothewidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14488 \settothewidth{\dimen@ii}{#2}%
14489 \ifdim\dimen@ii>\dimen@
14490 \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14491 \fi
14492 }%
14493 }

```

`glswidestname` Provide a user-level macro to obtain the widest top-level name.

```

14494 \newcommand*\glswidestname{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

14495 \newcommand*\glswidestsubname}[1]{%
14496 \ifcsundef{@glswidestname\romannumeral#1}%
14497 {\@glswidestname}%
14498 {\csuse{@glswidestname\romannumeral#1}}%
14499 }

```

`glswidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glswidestTopLevelName`.

```

14500 \let\glswidestTopLevelName\glswidestTopLevelName

```

`glswidestUsedTopLevelName` Like `\glswidestTopLevelName` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```

14501 \newrobustcmd*\glswidestUsedTopLevelName}[1][\@glo@types]{%
14502 \dimen@=0pt\relax
14503 \gls@tmplen=0pt\relax
14504 \forallglossaries[#1]{\@gls@type}%
14505 {%
14506 \forallglsentries[\@gls@type]{\@glo@label}%
14507 {%
14508 \ifglsused{\@glo@label}%
14509 {%
14510 \ifglshasparent{\@glo@label}%
14511 }%
14512 }%

```

```

14513         \settowidth{\dimen@}%
14514         {\glstreenamefmt{\glstentryname{\@glo@label}}}%
14515         \ifdim\dimen@>\gls@tmplen
14516             \gls@tmplen=\dimen@
14517             \eglssetwidest{\glstentryname{\@glo@label}}%
14518         \fi
14519     }%
14520 }%
14521 {}%
14522 }%
14523 }%
14524 }

```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

14525 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
14526     \dimen@=0pt\relax
14527     \gls@tmplen=0pt\relax
14528     \forallglossaries[#1]{\@gls@type}%
14529     {%
14530         \forglentries[\@gls@type]{\@glo@label}%
14531         {%
14532             \ifglsused{\@glo@label}%
14533             {%
14534                 \settowidth{\dimen@}%
14535                 {\glstreenamefmt{\glstentryname{\@glo@label}}}%
14536                 \ifdim\dimen@>\gls@tmplen
14537                     \gls@tmplen=\dimen@
14538                     \eglssetwidest{\glstentryname{\@glo@label}}%
14539                 \fi
14540             }%
14541         }%
14542     }%
14543 }%
14544 }

```

`ndWidestAnyName` Like the above but doesn't check is the entry has been used.

```

14545 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
14546     \dimen@=0pt\relax
14547     \gls@tmplen=0pt\relax
14548     \forallglossaries[#1]{\@gls@type}%
14549     {%
14550         \forglentries[\@gls@type]{\@glo@label}%
14551         {%
14552             \settowidth{\dimen@}%
14553             {\glstreenamefmt{\glstentryname{\@glo@label}}}%
14554             \ifdim\dimen@>\gls@tmplen
14555                 \gls@tmplen=\dimen@
14556                 \eglssetwidest{\glstentryname{\@glo@label}}%

```



```

14557     \fi
14558     }%
14559     }%
14560   }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

14561 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
14562   \dimen@=0pt\relax
14563   \dimen@i=0pt\relax
14564   \dimen@ii=0pt\relax
14565   \forallglossaries[#1]{\@gls@type}%
14566   {%
14567     \forallglsentries[\@gls@type]{\@glo@label}%
14568     {%
14569       \ifglsused{\@glo@label}%
14570       {%
14571         \ifglsahasparent{\@glo@label}%
14572         {%
14573           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
14574           \ifglsahasparent{\@glo@parent}%
14575           {%
14576             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
14577             \ifglsahasparent{\@glo@parent}%
14578             }%
14579             {%
14580               \settowidth{\gls@tmplen}%
14581                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
14582               \ifdim\gls@tmplen>\dimen@ii
14583                 \dimen@ii=\gls@tmplen
14584                 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
14585                 \fi
14586               }%
14587             }%
14588             {%
14589               \settowidth{\gls@tmplen}%
14590                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
14591               \ifdim\gls@tmplen>\dimen@i
14592                 \dimen@i=\gls@tmplen
14593                 \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14594               \fi
14595             }%
14596           }%
14597         }%
14598         \settowidth{\gls@tmplen}%
14599           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
14600       \ifdim\gls@tmplen>\dimen@
14601         \dimen@=\gls@tmplen
14602         \eglssetwidest{\glsentryname{\@glo@label}}%

```

```

14603         \fi
14604     }%
14605 }%
14606 {}%
14607 }%
14608 }%
14609 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

14610 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
14611     \dimen@=0pt\relax
14612     \dimen@i=0pt\relax
14613     \dimen@ii=0pt\relax
14614     \forallglossaries[#1]{\@gls@type}%
14615     {%
14616         \forglentries[\@gls@type]{\@glo@label}%
14617         {%
14618             \ifglshasparent{\@glo@label}%
14619             {%
14620                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
14621                 \ifglshasparent{\@glo@parent}%
14622                 {%
14623                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
14624                     \ifglshasparent{\@glo@parent}%
14625                     }%
14626                 }%
14627                 \settowidth{\gls@tmplen}%
14628                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14629                 \ifdim\gls@tmplen>\dimen@ii
14630                     \dimen@ii=\gls@tmplen
14631                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
14632                 \fi
14633             }%
14634         }%
14635     }%
14636     \settowidth{\gls@tmplen}%
14637         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14638     \ifdim\gls@tmplen>\dimen@i
14639         \dimen@i=\gls@tmplen
14640         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14641     \fi
14642 }%
14643 }%
14644 {%
14645     \settowidth{\gls@tmplen}%
14646         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14647     \ifdim\gls@tmplen>\dimen@
14648         \dimen@=\gls@tmplen
14649         \eglssetwidest{\glsentryname{\@glo@label}}%

```

```

14650     \fi
14651     }%
14652     }%
14653     }%
14654 }

```

`\edAnyNameSymbol` Like the `\glsFindWidestUsedAnyName` but also measures the symbol. The length of the widest symbol is stored in the second argument should be a length register.

```

14655 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
14656   \dimen@=0pt\relax
14657   \gls@tmplen=0pt\relax
14658   #2=0pt\relax
14659   \forallglossaries[#1]{\@gls@type}%
14660   {%
14661     \forglentries[\@gls@type]{\@glo@label}%
14662     {%
14663       \ifglsused{\@glo@label}%
14664       {%
14665         \settowidth{\dimen@}%
14666         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14667         \ifdim\dimen@>\gls@tmplen
14668           \gls@tmplen=\dimen@
14669           \eglssetwidest{\glsentryname{\@glo@label}}%
14670         \fi
14671         \settowidth{\dimen@}%
14672         {\glsentrysymbol{\@glo@label}}%
14673         \ifdim\dimen@>#2\relax
14674           #2=\dimen@
14675         \fi
14676       }%
14677     }%
14678   }%
14679 }%
14680 }

```

`\stAnyNameSymbol` Like the above but doesn't check if the entry has been used.

```

14681 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14682   \dimen@=0pt\relax
14683   \gls@tmplen=0pt\relax
14684   #2=0pt\relax
14685   \forallglossaries[#1]{\@gls@type}%
14686   {%
14687     \forglentries[\@gls@type]{\@glo@label}%
14688     {%
14689       \settowidth{\dimen@}%
14690       {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14691       \ifdim\dimen@>\gls@tmplen
14692         \gls@tmplen=\dimen@
14693       \eglssetwidest{\glsentryname{\@glo@label}}%

```

```

14694     \fi
14695     \settowidth{\dimen@}%
14696       {\glsentrysymbol{\@glo@label}}}%
14697     \ifdim\dimen@>#2\relax
14698       #2=\dimen@
14699     \fi
14700   }%
14701 }%
14702 }

```

`eSymbolLocation` Like the `\glsFindWidestUsedAnyNameSymbol` but also measures the location list. This requires `\glsentrynumberlist`. The length of the widest symbol is stored in the second argument should be a length register. The length of the widest location list is stored in the third argument, which should also be a length register.

```

14703 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14704   \dimen@=0pt\relax
14705   \gls@tmplen=0pt\relax
14706   #2=0pt\relax
14707   #3=0pt\relax
14708   \forallglossaries[#1]{\@gls@type}%
14709   {%
14710     \forglsentries[\@gls@type]{\@glo@label}%
14711     {%
14712       \ifglsused{\@glo@label}%
14713       {%
14714         \settowidth{\dimen@}%
14715           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14716         \ifdim\dimen@>\gls@tmplen
14717           \gls@tmplen=\dimen@
14718           \eglssetwidest{\glsentryname{\@glo@label}}%
14719         \fi
14720         \settowidth{\dimen@}%
14721           {\glsentrysymbol{\@glo@label}}}%
14722         \ifdim\dimen@>#2\relax
14723           #2=\dimen@
14724         \fi
14725         \settowidth{\dimen@}%
14726           {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14727         \ifdim\dimen@>#3\relax
14728           #3=\dimen@
14729         \fi
14730       }%
14731     }%
14732   }%
14733 }%
14734 }

```

`eSymbolLocation` Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14735 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%

```

```

14736 \dimen@=0pt\relax
14737 \gls@tmplen=0pt\relax
14738 #2=0pt\relax
14739 #3=0pt\relax
14740 \forallglossaries[#1]{\@gls@type}%
14741 {%
14742   \forallglsentries[\@gls@type]{\@glo@label}%
14743   {%
14744     \settowidth{\dimen@}%
14745     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
14746     \ifdim\dimen@>\gls@tmplen
14747       \gls@tmplen=\dimen@
14748       \eglssetwidest{\glsentryname{\@glo@label}}%
14749     \fi
14750     \settowidth{\dimen@}%
14751     {\glsentrysymbol{\@glo@label}}%
14752     \ifdim\dimen@>#2\relax
14753       #2=\dimen@
14754     \fi
14755     \settowidth{\dimen@}%
14756     {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14757     \ifdim\dimen@>#3\relax
14758       #3=\dimen@
14759     \fi
14760   }%
14761 }%
14762 }

```

AnyNameLocation Like the `\glsFindWidestUsedAnyNameSymbolLocation` but doesn't measure the symbol. The length of the widest location list is stored in the second argument, which should be a length register.

```

14763 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14764   \dimen@=0pt\relax
14765   \gls@tmplen=0pt\relax
14766   #2=0pt\relax
14767   \forallglossaries[#1]{\@gls@type}%
14768   {%
14769     \forallglsentries[\@gls@type]{\@glo@label}%
14770     {%
14771       \ifglsused{\@glo@label}%
14772       {%
14773         \settowidth{\dimen@}%
14774         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
14775         \ifdim\dimen@>\gls@tmplen
14776           \gls@tmplen=\dimen@
14777           \eglssetwidest{\glsentryname{\@glo@label}}%
14778         \fi
14779         \settowidth{\dimen@}%
14780         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%

```

```

14781         \ifdim\dimen@>#2\relax
14782             #2=\dimen@
14783         \fi
14784     }%
14785     {}%
14786 }%
14787 }%
14788 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14789 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14790     \dimen@=0pt\relax
14791     \gls@tmplen=0pt\relax
14792     #2=0pt\relax
14793     \forallglossaries[#1]{\@gls@type}%
14794     {%
14795         \forallglsentries[\@gls@type]{\@glo@label}%
14796         {%
14797             \settowidth{\dimen@}%
14798             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14799             \ifdim\dimen@>\gls@tmplen
14800                 \gls@tmplen=\dimen@
14801                 \eglssetwidest{\glsentryname{\@glo@label}}%
14802             \fi
14803             \settowidth{\dimen@}%
14804             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14805             \ifdim\dimen@>#2\relax
14806                 #2=\dimen@
14807             \fi
14808         }%
14809     }%
14810 }

```

computeTreeIndent Compute the value of `\glstreeindent`. Argument is the entry label. (Ignored in default definition, but this command may be redefined to take the particular entry into account.) Note that the sub-levels modify `\glstreeindent`.

```

14811 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
14812     \glstreeindent=\glsxtrtreetopindent\relax
14813 }

```

computeTreeSubIndent `\glsxtrComputeTreeSubIndent{<level>}{<label>}{<register>}`

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```

14814 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%

```

```

14815 \ifcsundef{@glswidestname\romannumeral#1}%
14816 {%
14817 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14818 }%
14819 {%
14820 \settowidth{#3}{\glstreenamefmt{%
14821 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14822 }%
14823 }

```

eeSetHangIndent Set \hangindent for top-level entries:

```
14824 \newcommand*{\glxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14825 \newcommand*{\glxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine altree:

```

14826 \renewglossarystyle{almtree}{%
14827 \renewenvironment{theglossary}%
14828 {%
14829 \glxtralmtreeInit
14830 \def\@gls@prevlevel{-1}%
14831 \mbox{}\par}%
14832 {\par}%
14833 \renewcommand*{\glossaryheader}{}%
14834 \renewcommand*{\glsgroupheading}[1]{}%
14835 \renewcommand{\glossentry}[2]{%
14836 \ifnum\@gls@prevlevel=0\relax
14837 \else
14838 \glxtrComputeTreeIndent{##1}%
14839 \fi
14840 \parindent\glstreeindent
14841 \glxtrAltTreeSetHangIndent
14842 \makebox[0pt][r]%
14843 {%
14844 \glstreenamebox{\glstreeindent}%
14845 {%
14846 \glssentryitem{##1}%
14847 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14848 }%
14849 }%
14850 \glxtralmtreeSymbolDescLocation{##1}{##2}%
14851 \def\@gls@prevlevel{0}%
14852 }
14853 \renewcommand{\subglossentry}[3]{%
14854 \ifnum##1=1\relax
14855 \glssubentryitem{##2}%
14856 \fi
14857 \ifnum\@gls@prevlevel=##1\relax

```

```

14858 \else
14859 \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14860 \ifnum\@gls@prevlevel<##1\relax
14861 \setlength\glstreeindent\gls@tmplen
14862 \addtolength\glstreeindent\parindent
14863 \parindent\glstreeindent
14864 \else
14865 \ifnum\@gls@prevlevel=0\relax
14866 \glxtrComputeTreeIndent{##2}%
14867 \else
14868 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14869 \fi
14870 \addtolength\parindent{-\glstreeindent}%
14871 \setlength\glstreeindent\parindent
14872 \fi
14873 \fi
14874 \glxtrAltTreeSetSubHangIndent{##1}%
14875 \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
14876 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14877 \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14878 \def\@gls@prevlevel{##1}%
14879 }%
14880 \renewcommand*\{glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
14881 }
14882 }%
14883 {%
14884 }

```

Redefine `almtreegroup` so that it discourages a break after group headings. Can't use `\@afterheading` here as it messes with the first item of the group.

```

14885 \ifdef{\@glsstyle@almtreegroup}
14886 {%
14887 \renewglossarystyle{almtreegroup}{%
14888 \setglossarystyle{almtree}%
14889 \renewcommand{\glsgroupheading}[1]{\par
14890 \def\@gls@prevlevel{-1}%
14891 \hangindent0pt\relax
14892 \parindent0pt\relax
14893 \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
14894 \glstreePreHeader{##1}{\glxtr@grptitle}%
14895 \glstreegroupheaderfmt{\glxtr@grptitle}%
14896 \glstreegroupskip
14897 }%
14898 }%
14899 }%
14900 {%
14901 }

```

Similarly for `almtreehypergroup`.

```

14902 \ifdef{\@glsstyle@almtreehypergroup}

```



```

14903 {%
14904 \renewglossarystyle{almtreehypergroup}{%
14905   \setglossarystyle{almtree}%
14906   \renewcommand*{\glossaryheader}{%
14907     \par
14908     \def\@gls@prevlevel{-1}%
14909     \hangindent0pt\relax
14910     \parindent0pt\relax
14911     \glstreenavigationfmt{\glsnavigation}%
14912     \glstreegroupskip
14913   }%
14914 \renewcommand*{\glsgroupheading}[1]{%
14915   \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
14916   \glstreePreHeader{##1}{\glsxtr@grptitle}%
14917   \par
14918   \def\@gls@prevlevel{-1}%
14919   \hangindent0pt\relax
14920   \parindent0pt\relax
14921   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
14922   \glstreegroupskip
14923 }%
14924 }
14925 }%
14926 {%
14927 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14928 \ifdef{\@glsstyle@mcolindexgroup}
14929 {%
14930 \renewglossarystyle{mcolindexgroup}{%
14931   \setglossarystyle{mcolindex}%
14932   \renewcommand*{\glsgroupheading}[1]{%
14933     \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
14934     \glstreePreHeader{##1}{\glsxtr@grptitle}%
14935     \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
14936     \glstreegroupskip\@afterheading
14937   }%
14938 }
14939 }%
14940 {%
14941 }

```

Similarly for `mcolindexhypergroup`.

```

14942 \ifdef{\@glsstyle@mcolindexhypergroup}
14943 {%
14944 \renewglossarystyle{mcolindexhypergroup}{%
14945   \setglossarystyle{mcolindex}%

```

```

14946 \renewcommand*\glossaryheader}{%
14947 \item\glstreenavigationfmt{\glsnavigation}%
14948 \glstreegroupskip
14949 }%
14950 \renewcommand*\glsgroupheading}[1]{%
14951 \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
14952 \glstreePreHeader{##1}{\glsxtr@grptitle}%
14953 \item\glstreegroupheaderfmt
14954 {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
14955 \glstreegroupskip\@afterheading
14956 }%
14957 }
14958 }%
14959 {%
14960 }

```

Similarly for mcolindexspannav.

```

14961 \ifdef{\@glsstyle@mcolindexspannav}
14962 {%
14963 \renewglossarystyle{mcolindexspannav}{%
14964 \setglossarystyle{index}%
14965 \renewenvironment{theglossary}%
14966 {%
14967 \begin{multicols}{\glscols}[\noindent\glstreenavigationfmt{\glsnavigation}]}%
14968 \setlength{\parindent}{0pt}%
14969 \setlength{\parskip}{0pt plus 0.3pt}%
14970 \let\item\glstreeitem}%
14971 {\end{multicols}}}%
14972 \renewcommand*\glsgroupheading}[1]{%
14973 \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
14974 \glstreePreHeader{##1}{\glsxtr@grptitle}%
14975 \item\glstreegroupheaderfmt
14976 {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
14977 \glstreegroupskip\@afterheading
14978 }%
14979 }
14980 }%
14981 {%
14982 }

```

Similarly for mcoltreegroup.

```

14983 \ifdef{\@glsstyle@mcoltreegroup}
14984 {%
14985 \renewglossarystyle{mcoltreegroup}{%
14986 \setglossarystyle{mcoltree}%
14987 \renewcommand*\glsgroupheading}[1]{%
14988 \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
14989 \glstreePreHeader{##1}{\glsxtr@grptitle}%
14990 \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}}%
14991 \glstreegroupskip\@afterheading

```

```

14992 }%
14993 }
14994 }%
14995 {%
14996 }

```

Similarly for mcoltreehypergroup.

```

14997 \ifdef{\@glsstyle@mcoltreehypergroup}
14998 {%
14999   \renewglossarystyle{mcoltreehypergroup}{%
15000     \setglossarystyle{mcoltree}%
15001     \renewcommand*\glossaryheader{%
15002       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15003       \glstreegroupskip
15004     }%
15005     \renewcommand*\glsgroupheading}[1]{%
15006       \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15007       \glstreePreHeader{##1}{\glxtr@grptitle}%
15008       \par\noindent
15009       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glxtr@grptitle}}%
15010       \glstreegroupskip\@afterheading
15011     }%
15012   }
15013 }%
15014 {%
15015 }

```

Similarly for mcoltreespannav.

```

15016 \ifdef{\@glsstyle@mcoltreespannav}
15017 {%
15018   \renewglossarystyle{mcoltreespannav}{%
15019     \setglossarystyle{tree}%
15020     \renewenvironment{theglossary}%
15021     {%
15022       \begin{multicols}{\glsmcols}%
15023         [\noindent\glstreenavigationfmt{\glsnavigation}]%
15024         \setlength{\parindent}{0pt}%
15025         \setlength{\parskip}{0pt plus 0.3pt}%
15026       }%
15027     {\end{multicols}}%
15028     \renewcommand*\glsgroupheading}[1]{%
15029       \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15030       \glstreePreHeader{##1}{\glxtr@grptitle}%
15031       \par\noindent
15032       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glxtr@grptitle}}%
15033       \glstreegroupskip\@afterheading
15034     }%
15035   }
15036 }%
15037 {%

```

15038 }

Similarly for mcoltreenonamegroup.

```
15039 \ifdef{\@glsstyle@mcoltreenonamegroup}
15040 {%
15041   \renewglossarystyle{mcoltreenonamegroup}{%
15042     \setglossarystyle{mcoltreenoname}%
15043     \renewcommand{\glsgroupheading}[1]{%
15044       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15045       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15046       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15047       \glstreegroupskip\@afterheading
15048     }%
15049   }
15050 }%
15051 {%
15052 }
```

Similarly for mcoltreenonamehypergroup.

```
15053 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
15054 {%
15055   \renewglossarystyle{mcoltreenonamehypergroup}{%
15056     \setglossarystyle{mcoltreenoname}%
15057     \renewcommand*\glossaryheader{%
15058       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15059       \glstreegroupskip
15060     }%
15061     \renewcommand*\glsgroupheading[1]{%
15062       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15063       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15064       \par\noindent
15065       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15066       \glstreegroupskip\@afterheading}%
15067   }
15068 }%
15069 {%
15070 }
```

Similarly for mcoltreenonamespannav.

```
15071 \ifdef{\@glsstyle@mcoltreenonamespannav}
15072 {%
15073   \renewglossarystyle{mcoltreenonamespannav}{%
15074     \setglossarystyle{treenoname}%
15075     \renewenvironment{theglossary}%
15076     {%
15077       \begin{multicols}{\glscols}%
15078       [\noindent\glstreenavigationfmt{\glsnavigation}]%
15079       \setlength{\parindent}{0pt}%
15080       \setlength{\parskip}{0pt plus 0.3pt}%
15081     }%
15082     {\end{multicols}}%

```

```

15083   \renewcommand*\glsgroupheading}[1]{%
15084     \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15085     \glstreePreHeader{##1}{\glsxtr@grptitle}%
15086     \par\noindent
15087     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptile}}%
15088     \glstreegroupskip\@afterheading}%
15089   }
15090 }%
15091 {%
15092 }

```

mcolalmtree needs adjusting so that it uses \glsxtralttreeInit This doesn't use \mbox{\par which would unbalance the top of the columns.

```

15093 \ifdef{\@glsstyle@mcolalmtree}
15094 {%
15095   \renewglossarystyle{mcolalmtree}{%
15096     \setglossarystyle{alttree}%
15097     \renewenvironment{theglossary}%
15098     {%
15099       \glsxtralttreeInit
15100       \def\@gls@prevlevel{-1}%
15101       \begin{multicols}{\glsmlcols}%
15102     }%
15103     {\par\end{multicols}}}%
15104   }
15105 }%
15106 {%
15107 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

15108 \ifdef{\@glsstyle@mcolalmtreegroup}
15109 {%
15110   \renewglossarystyle{mcolalmtreegroup}{%
15111     \setglossarystyle{mcolalmtree}%
15112     \renewcommand*\glsgroupheading}[1]{%
15113       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15114       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15115       \par
15116       \def\@gls@prevlevel{-1}%
15117       \hangindentOpt\relax
15118       \parindentOpt\relax
15119       \glstreegroupheaderfmt{\glsxtr@grptitle}%
15120       \glstreegroupskip
15121     }%
15122   }
15123 }%
15124 {%
15125 }

```

Similarly for mcolalmtreehypergroup.

```

15126 \ifdef{\@glsstyle@mcolalmtreehypergroup}
15127 {%
15128   \renewglossarystyle{mcolalmtreehypergroup}{%
15129     \setglossarystyle{mcolalmtree}%
15130     \renewcommand*\glossaryheader}{%
15131       \par
15132       \def\@gls@prevlevel{-1}%
15133       \hangindentOpt\relax
15134       \parindentOpt\relax
15135       \glstreenavigationfmt{\glsnavigation}%
15136       \glstreegroupskip
15137     }%
15138     \renewcommand*\glsgroupheading}[1]{%
15139       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15140       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15141       \par
15142       \def\@gls@prevlevel{-1}%
15143       \hangindentOpt\relax
15144       \parindentOpt\relax
15145       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15146       \glstreegroupskip
15147     }%
15148   }
15149 }%
15150 {%
15151 }

```

Similarly for mcolalmtreespannav.

```

15152 \ifdef{\@glsstyle@mcolalmtreespannav}
15153 {%
15154   \renewglossarystyle{mcolalmtreespannav}{%
15155     \setglossarystyle{almtree}%
15156     \renewenvironment{theglossary}%
15157     {%
15158       \glsxtralmtreeInit
15159       \def\@gls@prevlevel{-1}%
15160       \begin{multicols}{\glsmcols}%
15161         [\noindent\glstreenavigationfmt{\glsnavigation}]%
15162     }%
15163     {\par\end{multicols}}%
15164     \renewcommand*\glsgroupheading}[1]{%
15165       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15166       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15167       \par
15168       \def\@gls@prevlevel{-1}%
15169       \hangindentOpt\relax
15170       \parindentOpt\relax
15171       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15172       \glstreegroupskip
15173     }%

```

```
15174 }  
15175 }%  
15176 {%  
15177 }
```

Reset the default style

```
15178 \ifx\@glossary@default@style\relax  
15179 \else  
15180 \setglossarystyle{\@glxtr@current@style}  
15181 \fi
```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
15182 \NeedsTeXFormat{LaTeX2e}
15183 \ProvidesPackage{glossary-bookindex}[2019/04/09 v1.41 (NLCT)]
```

Load required packages.

```
15184 \RequirePackage{multicol}
15185 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
15186 \newcommand{\glxstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
15187 \newcommand*{\glxstrbookindexname}[1]{\glossentryname{#1}}
```

`ookindexsubname` Format used for sub entries.

```
15188 \newcommand*{\glxstrbookindexsubname}[1]{\glxstrbookindexname{#1}}
```

`xsxtrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
15189 \providecommand*{\glxstrprelocation}{\space}
```

`ndexprelocation` Separator used before location list for top-level entries. Version 1.22 has removed the `\ifglxsnopostdot` check since this style doesn't display the description.

```
15190 \newcommand*{\glxstrbookindexprelocation}[1]{%
15191 \glxstrifhasfield{location}{#1}%
15192 {,\glxstrprelocation}%
15193 {\glxstrprelocation}%
15194 }
```

`xsubprelocation` Separator used before location list for sub-entries.

```
15195 \newcommand*{\glxstrbookindexsubprelocation}[1]{%
15196 \glxstrbookindexprelocation{#1}%
15197 }
```

`ookindexlocation` `\glxstrbookindexlocation{<label>}{<location>}`

Displays the location.

```
15198 \newcommand*{\glxstrbookindexlocation}[2]{#2}
```


`\indexsublocation` `\glsxtrbookindexlocation{<label>}{<location>}`

Displays the location for sub-entries.

```
15199 \newcommand*{\glsxtrbookindexsublocation}{\glsxtrbookindexlocation}
```

`\xparentchildsep` Separator used between top-level parent and child entry.

```
15200 \newcommand{\glsxtrbookindexparentchildsep}{\nopagebreak}
```

`\xrentsubchildsep` Separator used between sub-level parent and child entry.

```
15201 \newcommand{\glsxtrbookindexparentschildsep}{\glsxtrbookindexparentchildsep}
```

`\bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
15202 \newcommand{\glsxtrbookindexbetween}[2]{}
```

`\indexsubbetween` Between two level 1 entries identified by the labels in the arguments.

```
15203 \newcommand{\glsxtrbookindexsubbetween}[2]{}
```

`\xexsubsubbetween` Between two level 2 entries identified by the labels in the arguments.

```
15204 \newcommand{\glsxtrbookindexsubsubbetween}[2]{}
```

`\indexatendgroup` At the end of a letter group. The argument is the index of the last top-level entry.

```
15205 \newcommand{\glsxtrbookindexatendgroup}[1]{}
```

`\xexsubatendgroup` At the end of a letter group. The argument is the index of the last level 1 entry.

```
15206 \newcommand{\glsxtrbookindexsubatendgroup}[1]{}
```

`\xsubsubatendgroup` At the end of a letter group. The argument is the index of the last level 2 entry.

```
15207 \newcommand{\glsxtrbookindexsubsubatendgroup}[1]{}
```

`\kindexgroupskip` Group separator.

```
15208 \newcommand{\glsxtrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

`\indexformatheader` Group separator.

```
15209 \newcommand*{\glsxtrbookindexformatheader}[1]{%
15210 \par{\centering\glstreegroupheaderfmt{#1}\par}%
15211 }
```

`\bookindexbookmark` Book mark group heading if supported.

```
15212 \ifdef\pdfbookmark
15213 {%
15214 \newcommand*{\glsxtrbookindexbookmark}[2]{%
15215 \ifdefstring{@@glossarysec}{chapter}%
15216 {\pdfbookmark[1]{#1}{#2}}%
15217 {\pdfbookmark[2]{#1}{#2}}%
```

```

15218 }
15219 }
15220 {%
15221 \newcommand*\glxtrbookindexbookmark}[2]{}
15222 }

```

kindexcolspread

```

15223 \newcommand*\glxtrbookindexcolspread{}

```

dexmulticolseenv

```

15224 \newcommand*\glxtrbookindexmulticolseenv{multicols}

```

Define the style.

```

15225 \newglossarystyle{bookindex}{%
15226 \setglossarystyle{index}%
15227 \renewenvironment{theglossary}%
15228 {%
15229 \ifnum\glxtrbookindexcols>1\relax
15230 \ifdefempty\glxtrbookindexcolspread
15231 {%
15232 \edef\glxtr@beginbookindex{%
15233 \noexpand\begin{\glxtrbookindexmulticolseenv}
15234 {\glxtrbookindexcols}%
15235 }%
15236 }%
15237 {%
15238 \edef\glxtr@beginbookindex{%
15239 \noexpand\begin{\glxtrbookindexmulticolseenv}%
15240 {\glxtrbookindexcols}[\glxtrbookindexcolspread]%
15241 }%
15242 }%
15243 \else
15244 \def\glxtr@beginbookindex{}%
15245 \fi
15246 \glxtr@beginbookindex
15247 \setlength{\parindent}{0pt}%
15248 \setlength{\parskip}{0pt plus 0.3pt}%
15249 \let\@glxtr@bookindex@sep\glxtrbookindexparentchildsep
15250 \let\@glxtr@bookindex@subsep\glxtrbookindexparentschildsep
15251 \let\@glxtr@bookindex@between\@gobble
15252 \let\@glxtr@bookindex@subbetween\@gobble
15253 \let\@glxtr@bookindex@subsubbetween\@gobble
15254 \let\@glxtr@bookindex@atendgroup\relax
15255 \let\@glxtr@bookindex@subatendgroup\relax
15256 \let\@glxtr@bookindex@subsubatendgroup\relax
15257 \let\@glxtr@bookindexgroupskip\relax
15258 }%
15259 {%

```

Do end group hooks.

```
15260 \@glstr@bookindex@subsubatendgroup
15261 \@glstr@bookindex@subatendgroup
15262 \@glstr@bookindex@atendgroup
```

End multicols environment.

```
15263 \ifnum\glstrbookindexcols>1\relax
15264 \edef\glstr@endbookindex{%
15265 \noexpand\end{\glstrbookindexmulticolseenv}%
15266 }%
15267 \else
15268 \def\glstr@endbookindex{%
15269 \fi
15270 \glstr@endbookindex
15271 }%
```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```
15272 \renewcommand*\glossaryheader{\raggedright}%
```

Top level entry format.

```
15273 \renewcommand*\glossentry}[2]{%
```

Do separator.

```
15274 \@glstr@bookindex@between{##1}%
```

Update separators.

```
15275 \let\@glstr@bookindex@sep\glstrbookindexparentchildsep
15276 \let\@glstr@bookindex@subsep\glstrbookindexparentschildsep
15277 \let\@glstr@bookindex@subbetween@gobble
15278 \let\@glstr@bookindex@subsubbetween@gobble
15279 \edef\@glstr@bookindex@between{%
15280 \noexpand\glstrbookindexbetween{##1}%
15281 }%
15282 \edef\@glstr@bookindex@atendgroup{%
15283 \noexpand\glstrbookindexatendgroup{##1}%
15284 }%
15285 \let\@glstr@bookindex@subatendgroup\relax
15286 \let\@glstr@bookindex@subsubatendgroup\relax
```

Format entry.

```
15287 \glstreeitem
15288 \glstryitem{##1}%
15289 \glstarget{##1}{\glstrbookindexname{##1}}%
15290 \glstrbookindexprelocation{##1}%
15291 \glstrbookindexlocation{##1}{##2}%
15292 }%
15293 \renewcommand*\subglossentry}[3]{%
15294 \ifcase##1\relax
```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```
15295 \glstreeitem
15296 \or
```

Level 1.

```
15297 \@glxtr@bookindex@sep
15298 \@glxtr@bookindex@subbetween{##2}%
15299 \let\@glxtr@bookindex@sep\relax
```

Update separators.

```
15300 \let\@glxtr@bookindex@subsubbetween\@gobble
15301 \let\@glxtr@bookindex@subsep\glxtrbookindexparentschildsep
15302 \edef\@glxtr@bookindex@subbetween{%
15303     \noexpand\glxtrbookindexsubbetween{##2}%
15304 }%
15305 \edef\@glxtr@bookindex@atsubendgroup{%
15306     \noexpand\glxtrbookindexatsubendgroup{##1}%
15307 }%
```

Start sub-item.

```
15308 \glstreesubitem
15309 \glssubentryitem{##2}%
15310 \else
```

All other levels.

```
15311 \@glxtr@bookindex@subsep
15312 \@glxtr@bookindex@subsubbetween{##2}%
```

Update separators.

```
15313 \let\@glxtr@bookindex@subsep\relax
15314 \edef\@glxtr@bookindex@subsubbetween{%
15315     \noexpand\glxtrbookindexsubsubbetween{##2}%
15316 }%
15317 \edef\@glxtr@bookindex@atsubsubendgroup{%
15318     \noexpand\glxtrbookindexatsubsubendgroup{##1}%
15319 }%
```

Start sub-sub-item.

```
15320 \glstreesubsubitem
15321 \fi
```

Format entry.

```
15322 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
15323 \glxtrbookindexsubprelocation{##2}%
15324 \glxtrbookindexsublocation{##2}{##3}%
15325 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
15326 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
15327 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
15328 \@glxtr@bookindex@subsubatendgroup
15329 \@glxtr@bookindex@subatendgroup
```

```
15330 \glsxtr@bookindex@atendgroup
15331 \glsxtr@bookindexgroupskip
```

Update separators.

```
15332 \let\glsxtr@bookindexgroupskip\glsxtrbookindexgroupskip
15333 \let\glsxtr@bookindex@between\gobble
15334 \let\glsxtr@bookindex@atendgroup\relax
15335 \let\glsxtr@bookindex@subatendgroup\relax
15336 \let\glsxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
15337 \glsxtrgetgrouptitle{##1}{\glsxtrcurrentgrptitle}%
```

Do the PDF bookmark if supported.

```
15338 \glsxtrbookindexbookmark{\glsxtrcurrentgrptitle}{index.##1}%
```

Format the group title.

```
15339 \glsxtrbookindexformatheader{\glsxtrcurrentgrptitle}%
15340 \nopagebreak\indexspace\nopagebreak\@afterheading
15341 }%
15342 }
```

Some supplementary commands that may be useful. These store the entry label for the current page. Since the page number is needed in the control sequence, this uses `\glsxtrbookindexthepage` instead of `\thepage` in case the page numbering has been set to something that contains formatting commands.

`bookindexthepage` The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```
15343 \newcommand{\glsxtrbookindexthepage}{%
15344 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
15345 }
```

`bookindexmarkentry` Writes entry information to the `.aux` file. The argument is the entry label.

```
15346 \newcommand*{\glsxtrbookindexmarkentry}[1]{%
15347 \protected@write\@auxout
15348 {\let\glsxtrbookindexthepage\relax}%
15349 {\string\glsxtr@setbookindexmark{\glsxtrbookindexthepage}{#1}}%
15350 }
```

`setbookindexmark`

```
15351 \newcommand*{\glsxtr@setbookindexmark}[2]{%
15352 \ifcsundef{glsxtr@idxfirstmark@#1}%
15353 {\csgdef{glsxtr@idxfirstmark@#1}{#2}}%
15354 {}%
15355 \csgdef{glsxtr@idxlastmark@#1}{#2}%
15356 }
```

`bookindexfirstmarkfmt`

```
15357 \newcommand*{\glsxtrbookindexfirstmarkfmt}[1]{%
15358 \glstryname{#1}%
15359 }
```

kindexfirstmark

```
15360 \newcommand*{\glxtrbookindexfirstmark}{%  
15361   \letcs{\glxtr@label}{glxtr@idxfirstmark@\glxtrbookindexthepage}%  
15362   \ifdef\glxtr@label  
15363     {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%  
15364     {}%  
15365 }
```

indexlastmarkfmt

```
15366 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%  
15367   \gl Sentryname{#1}%  
15368 }
```

okindexlastmark

```
15369 \newcommand*{\glxtrbookindexlastmark}{%  
15370   \letcs{\glxtr@label}{glxtr@idxlastmark@\glxtrbookindexthepage}%  
15371   \ifdef\glxtr@label  
15372     {\glxtrbookindexlastmarkfmt{\glxtr@label}}%  
15373     {}%  
15374 }
```

4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
15375 \NeedsTeXFormat{LaTeX2e}
```

```
15376 \ProvidesPackage{glossary-longextra}[2019/04/09 v1.41 (NLCT)]
```

Load required packages.

```
15377 \RequirePackage{glossary-longbooktabs}
```

longextraNameFmt `\glslongextraNameFmt{<label>}`

Governs the way the name is displayed.

```
15378 \newcommand{\glslongextraNameFmt}[1]{%
```

```
15379 \glstentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%
```

```
15380 }
```

longextraDescFmt `\glslongextraDescFmt{<label>}`

Governs the way the description is displayed.

```
15381 \newcommand{\glslongextraDescFmt}[1]{%
```

```
15382 \glossentrydesc{#1}\glspostdescription
```

```
15383 }
```

longextraSymbolFmt `\glslongextraSymbolFmt{<label>}`

Governs the way the symbol is displayed.

```
15384 \newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

longextraLocationFmt `\glslongextraLocationFmt{<label>}{<location list>}`

Governs the way the location is displayed.

```
15385 \newcommand{\glslongextraLocationFmt}[2]{#2}
```

extraSubNameFmt `\glslongextraSubNameFmt{<level>}{<label>}`

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
15386 \newcommand{\glslongextraSubNameFmt}[2]{%
15387 \glssubentryitem{#2}\glstarget{#2}{\strut}%
15388 }
```

extraSubDescFmt `\glslongextraSubDescFmt{<level>}{<label>}`

Governs the way the child description is displayed.

```
15389 \newcommand{\glslongextraSubDescFmt}[2]{%
15390 \glslongextraDescFmt{#2}%
15391 }
```

extraSubSymbolFmt `\glslongextraSubSymbolFmt{<level>}{<label>}`

Governs the way the child symbol is displayed.

```
15392 \newcommand{\glslongextraSubSymbolFmt}[2]{%
15393 \glslongextraSymbolFmt{#2}%
15394 }
```

extraSubLocationFmt `\glslongextraSubLocationFmt{<level>}{<label>}{<location list>}`

Governs the way the child location list is displayed.

```
15395 \newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

extraNameAlign Alignment for the name column.

```
15396 \newcommand{\glslongextraNameAlign}{l}
```

extraDescAlign Alignment for the description column.

```
15397 \newcommand{\glslongextraDescAlign}{>\raggedright}p{\glsdescwidth}
```

extraSymbolAlign Alignment for the symbol column.

```
15398 \newcommand{\glslongextraSymbolAlign}{c}
```


`raLocationAlign` Alignment for the location column.

```
15399 \newcommand{\glslongextraLocationAlign}>{\raggedright}p{\glspagelistwidth}}
```

`traGroupHeading` Used to format the letter group headings. The first argument is the number of columns in the table. The second is the group *label* (not the title).

```
15400 \newcommand{\glslongextraGroupHeading}[2]{}
```

`traHeaderFormat` Format for the column headers.

```
15401 \newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}
```

`raNameDescHeader`

```
15402 \newcommand{\glslongextraNameDescHeader}{%
15403 \glslongextraNameDescTabularHeader\endhead
15404 \glslongextraNameDescTabularFooter\endfoot
15405 }
```

`scTabularHeader`

```
15406 \newcommand{\glslongextraNameDescTabularHeader}{%
15407 \toprule
15408 \glslongextraHeaderFmt\entryname &
15409 \glslongextraHeaderFmt\descriptionname\tabularnewline
15410 \midrule
15411 }
```

`scTabularFooter`

```
15412 \newcommand{\glslongextraNameDescTabularFooter}{%
15413 \bottomrule
15414 }
```

Unlike the `alttree` style, there aren't different widths for the hierarchical levels.

`extraSetWidest` Provide in case the tree styles haven't been loaded.

```
15415 \newcommand*{\glslongextraSetWidest}[1]{%
15416 \def\@glslongextrawidestname{#1}%
15417 }
```

`extrawidestname` Pick up the widest name from the `alttree` style if it has been set. (Will expand to nothing otherwise.)

```
15418 \newcommand*{\@glslongextrawidestname}{\csuse{\glswidestname}}
```

`traUpdateWidest`

```
15419 \newcommand*{\glslongextraUpdateWidest}[1]{%
15420 \ifundef\@glslongextrawidestname
15421 {\def\@glslongextrawidestname{#1}}%
15422 {%
15423 \settowidth{\dimen@}{\@glslongextrawidestname}%
15424 \settowidth{\dimen@ii}{#1}%
15425 \ifdim\dimen@ii>\dimen@
```

```

15426     \def\@glslongextrawidestname{#1}%
15427     \fi
15428 }%
15429 }

```

updateWidestChild `\glslongextraUpdateWidestChild{<level>}{<text>}`

Used by `\glsxtrSetWidest` in `glossaries-extra-bib2gls`. Does nothing by default, since the default action in these styles is to omit the child name. If the child name should be displayed, then this needs to be redefined to use `\glslongextraUpdateWidest`.

```
15430 \newcommand*\glslongextraUpdateWidestChild[2]{}
```

traSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```

15431 \newcommand{\glslongextraSetDescWidth}{%
15432   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%

```

Has the widest name been set.

```

15433   \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
15434   \ifdim\dimen@>\gls@tmplen
15435     \gls@tmplen=\dimen@
15436   \fi

```

Description width is `\linewidth` less `4\tabcolsep` less the width of the name column.

```

15437   \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
15438 }

```

SymSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```
15439 \newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15440   \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```
15441   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```

Subtract `2\tabcolsep` and the symbol header width.

```

15442   \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
15443 }

```

LocSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, location and description columns.

```
15444 \newcommand{\glslongextraLocSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15445   \glslongextraSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15446 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15447 }
```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that have name, symbol, location and description columns.

```
15448 \newcommand{\glslongextraSymLocSetDescWidth}{%
```

Work out the size for just the name, symbol and description style.

```
15449 \glslongextraSymSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15450 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15451 }
```

ExtraUseTabular If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```
15452 \newif\ifGlsLongExtraUseTabular
```

```
15453 \GlsLongExtraUseTabularfalse
```

raTabularVAlign Only used with the tabular setting.

```
15454 \newcommand*{\glslongextraTabularVAlign}{c}
```

long-name-desc Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```
15455 \newglossarystyle{long-name-desc}%
```

```
15456 {%
```

```
15457 \ifGlsLongExtraUseTabular
```

```
15458 \renewenvironment{theglossary}%
```

```
15459 {%
```

```
15460 \glslongextraSetDescWidth
```

```
15461 \edef\@glslongextra@begintab{%
```

```
15462 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
```

```
15463 \expandonce\glslongextraNameAlign
```

```
15464 \expandonce\glslongextraDescAlign}}%
```

```
15465 \@glslongextra@begintab
```

```
15466 }%
```

```
15467 {%
```

```
15468 \glslongextraNameDescTabularFooter
```

```
15469 \end{tabular}%
```

```
15470 }%
```

```
15471 \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
```

```
15472 \else
```

```
15473 \renewenvironment{theglossary}%
```

```
15474 {%
```

```
15475 \glspatchLToutput
```

```
15476 \glslongextraSetDescWidth
```

```
15477 \edef\@glslongextra@begintab{%
```

```
15478 \noexpand\begin{longtable}{%
```

```

15479         \expandonce\glslongextraNameAlign
15480         \expandonce\glslongextraDescAlign}}%
15481     \@glslongextra@begintab
15482     }%
15483     {\end{longtable}}%
15484     \renewcommand*\glossaryheader{\glslongextraNameDescHeader}%
15485     \fi
15486     \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{2}{##1}}%
15487     \renewcommand\glossentry[2]{%
15488         \glslongextraNameFmt{##1} &
15489         \glslongextraDescFmt{##1}\tabularnewline
15490     }%
15491     \renewcommand\subglossentry[3]{%
15492         \glslongextraSubNameFmt{##1}{##2}
15493         &
15494         \glslongextraSubDescFmt{##1}{##2}%
15495         \tabularnewline
15496     }%
15497     \ifglsnogroupskip
15498     \renewcommand*\glsgroupskip{}%
15499     \else
15500     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15501     \fi
15502 }

```

cLocationHeader

```

15503 \newcommand{\glslongextraNameDescLocationHeader}{%
15504 \glslongextraNameDescLocationTabularHeader\endhead
15505 \glslongextraNameDescLocationTabularFooter\endfoot
15506 }

```

onTabularHeader

```

15507 \newcommand{\glslongextraNameDescLocationTabularHeader}{%
15508 \toprule
15509 \glslongextraHeaderFmt\entryname &
15510 \glslongextraHeaderFmt\descriptionname &
15511 \glslongextraHeaderFmt\pagelistname\tabularnewline
15512 \midrule
15513 }

```

onTabularFooter

```

15514 \newcommand{\glslongextraNameDescLocationTabularFooter}{%
15515 \bottomrule
15516 }

```

g-name-desc-loc Three columns: name, description and location list.

```

15517 \newglossarystyle{long-name-desc-loc}%
15518 {%
15519     \ifGlsLongExtraUseTabular

```

```

15520 \renewenvironment{theglossary}%
15521   {%
15522     \glslongextraLocSetDescWidth
15523     \edef\@glslongextra@begintab{%
15524       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15525         \expandonce\glslongextraNameAlign
15526         \expandonce\glslongextraDescAlign
15527         \expandonce\glslongextraLocationAlign
15528       }}%
15529     \@glslongextra@begintab
15530   }%
15531   {%
15532     \glslongextraNameDescLocationTabularFooter
15533     \end{tabular}%
15534   }%
15535 \renewcommand*\glossaryheader{\glslongextraNameDescLocationTabularHeader}%
15536 \else
15537 \renewenvironment{theglossary}%
15538   {%
15539     \glspatchLToutput
15540     \glslongextraLocSetDescWidth
15541     \edef\@glslongextra@begintab{%
15542       \noexpand\begin{longtable}{%
15543         \expandonce\glslongextraNameAlign
15544         \expandonce\glslongextraDescAlign
15545         \expandonce\glslongextraLocationAlign
15546       }}%
15547     \@glslongextra@begintab
15548   }%
15549   {\end{longtable}}%
15550 \renewcommand*\glossaryheader{\glslongextraNameDescLocationHeader}%
15551 \fi
15552 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15553 \renewcommand\glossentry[2]{%
15554   \glslongextraNameFmt{##1} &
15555   \glslongextraDescFmt{##1} &
15556   \glslongextraLocationFmt{##1}{##2}\tabularnewline
15557 }%
15558 \renewcommand\subglossentry[3]{%
15559   \glslongextraSubNameFmt{##1}{##2}&
15560   \glslongextraSubDescFmt{##1}{##2}&
15561   \glslongextraSubLocationFmt{##1}{##2}{##3}%
15562   \tabularnewline
15563 }%
15564 \ifglsgroupskip
15565   \renewcommand*\glsgroupskip{}%
15566 \else
15567   \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15568 \fi

```

15569 }

aDescNameHeader

```
15570 \newcommand{\glslongextraDescNameHeader}{%
15571 \glslongextraDescNameTabularHeader\endhead
15572 \glslongextraDescNameTabularFooter\endfoot
15573 }
```

meTabularHeader

```
15574 \newcommand{\glslongextraDescNameTabularHeader}{%
15575 \toprule
15576 \glslongextraHeaderFmt\descriptionname&
15577 \glslongextraHeaderFmt\entryname \tabularnewline
15578 \midrule
15579 }
```

meTabularFooter

```
15580 \newcommand{\glslongextraDescNameTabularFooter}{%
15581 \bottomrule
15582 }
```

long-desc-name Like name-desc but swaps the columns.

```
15583 \newglossarystyle{long-desc-name}%
15584 {%
15585 \ifGlsLongExtraUseTabular
15586 \renewenvironment{theglossary}%
15587 {%
15588 \glslongextraSetDescWidth
15589 \edef\@glslongextra@begintab{%
15590 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15591 \expandonce\glslongextraDescAlign
15592 \expandonce\glslongextraNameAlign}}%
15593 \@glslongextra@begintab
15594 }%
15595 {%
15596 \glslongextraDescNameTabularFooter
15597 \end{tabular}%
15598 }%
15599 \renewcommand*\glossaryheader{\glslongextraDescNameTabularHeader}%
15600 \else
15601 \renewenvironment{theglossary}%
15602 {%
15603 \glspatchLToutput
15604 \glslongextraSetDescWidth
15605 \edef\@glslongextra@begintab{%
15606 \noexpand\begin{longtable}{%
15607 \expandonce\glslongextraDescAlign
15608 \expandonce\glslongextraNameAlign}}%
15609 \@glslongextra@begintab
```

```

15610 }%
15611 {\end{longtable}}%
15612 \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
15613 \fi
15614 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
15615 \renewcommand{\glossentry}[2]{%
15616 \glslongextraDescFmt{##1} &
15617 \glslongextraNameFmt{##1}\tabularnewline
15618 }%
15619 \renewcommand{\subglossentry}[3]{%
15620 \glslongextraSubDescFmt{##1}{##2} &
15621 \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15622 }%
15623 \ifglsnogroupskip
15624 \renewcommand*{\glsgroupskip}{}%
15625 \else
15626 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15627 \fi
15628 }

```

nDescNameHeader

```

15629 \newcommand{\glslongextraLocationDescNameHeader}{%
15630 \glslongextraLocationDescNameTabularHeader\endhead
15631 \glslongextraLocationDescNameTabularFooter\endfoot
15632 }

```

meTabularHeader

```

15633 \newcommand{\glslongextraLocationDescNameTabularHeader}{%
15634 \toprule
15635 \glslongextraHeaderFmt\pagelistname&
15636 \glslongextraHeaderFmt\descriptionname&
15637 \glslongextraHeaderFmt\entryname \tabularnewline
15638 \midrule
15639 }

```

meTabularFooter

```

15640 \newcommand{\glslongextraLocationDescNameTabularFooter}{%
15641 \bottomrule
15642 }

```

g-loc-desc-name Three columns: location, description and name.

```

15643 \newglossarystyle{long-loc-desc-name}%
15644 {%
15645 \ifGlsLongExtraUseTabular
15646 {%
15647 \glslongextraLocSetDescWidth
15648 \edef\@glslongextra@begintab{%
15649 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15650 \expandonce\glslongextraLocationAlign

```

```

15651         \expandonce\glslongextraDescAlign
15652         \expandonce\glslongextraNameAlign}}%
15653     \@glslongextra@begintab
15654 }%
15655 {%
15656     \glslongextraLocationDescNameTabularFooter
15657     \end{tabular}%
15658 }%
15659 \renewcommand*\glossaryheader{\glslongextraLocationDescNameTabularHeader}%
15660 \else
15661 \renewenvironment{theglossary}%
15662 {%
15663     \glspatchLToutput
15664     \glslongextraLocSetDescWidth
15665     \edef\@glslongextra@begintab{%
15666         \noexpand\begin{longtable}{%
15667             \expandonce\glslongextraLocationAlign
15668             \expandonce\glslongextraDescAlign
15669             \expandonce\glslongextraNameAlign}}%
15670     \@glslongextra@begintab
15671 }%
15672 {\end{longtable}}%
15673 \renewcommand*\glossaryheader{\glslongextraLocationDescNameHeader}%
15674 \fi
15675 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15676 \renewcommand{\glossentry}[2]{%
15677     \glslongextraLocationFmt{##1}{##2} &
15678     \glslongextraDescFmt{##1} &
15679     \glslongextraNameFmt{##1}\tabularnewline
15680 }%
15681 \renewcommand{\subglossentry}[3]{%
15682     \glslongextraSubLocationFmt{##1}{##2}{##3} &
15683     \glslongextraSubDescFmt{##1}{##2} &
15684     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15685 }%
15686 \ifglsnogroupskip
15687     \renewcommand*\glsgroupskip{}%
15688 \else
15689     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15690 \fi
15691 }

```

meDescSymHeader

```

15692 \newcommand{\glslongextraNameDescSymHeader}{%
15693     \glslongextraNameDescSymTabularHeader\endhead
15694     \glslongextraNameDescSymTabularFooter\endfoot
15695 }

```

ymTabularHeader


```

15696 \newcommand{\glslongextraNameDescSymTabularHeader}{%
15697 \toprule
15698 \glslongextraHeaderFmt\entryname &
15699 \glslongextraHeaderFmt\descriptionname &
15700 \glslongextraHeaderFmt\symbolname\tabularnewline
15701 \midrule
15702 }

```

ymTabularFooter

```

15703 \newcommand{\glslongextraNameDescSymTabularFooter}{%
15704 \bottomrule
15705 }

```

g-name-desc-sym Three column style with symbol in the third column.

```

15706 \newglossarystyle{long-name-desc-sym}%
15707 {%
15708 \ifGlsLongExtraUseTabular
15709 \renewenvironment{theglossary}%
15710 {%
15711 \glslongextraSymSetDescWidth
15712 \edef\@glslongextra@begintab{%
15713 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15714 \expandonce\glslongextraNameAlign
15715 \expandonce\glslongextraDescAlign
15716 \expandonce\glslongextraSymbolAlign
15717 }}%
15718 \@glslongextra@begintab
15719 }%
15720 {%
15721 \glslongextraNameDescSymTabularFooter
15722 \end{tabular}%
15723 }%
15724 \renewcommand*\glossaryheader{\glslongextraNameDescSymTabularHeader}%
15725 \else
15726 \renewenvironment{theglossary}%
15727 {%
15728 \glspatchLToutput
15729 \glslongextraSymSetDescWidth
15730 \edef\@glslongextra@begintab{%
15731 \noexpand\begin{longtable}{%
15732 \expandonce\glslongextraNameAlign
15733 \expandonce\glslongextraDescAlign
15734 \expandonce\glslongextraSymbolAlign
15735 }}%
15736 \@glslongextra@begintab
15737 }%
15738 {\end{longtable}}%
15739 \renewcommand*\glossaryheader{\glslongextraNameDescSymHeader}%
15740 \fi

```

```

15741 \renewcommand*\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15742 \renewcommand{\glossentry}[2]{%
15743   \glslongextraNameFmt{##1} &
15744   \glslongextraDescFmt{##1} &
15745   \glslongextraSymbolFmt{##1}\tabularnewline
15746 }%
15747 \renewcommand{\subglossentry}[3]{%
15748   \glslongextraSubNameFmt{##1}{##2} &
15749   \glslongextraSubDescFmt{##1}{##2} &
15750   \glslongextraSubSymbolFmt{##1}{##2}%
15751   \tabularnewline
15752 }%
15753 \ifglsnogroupskip
15754   \renewcommand*\glsgroupskip}{}%
15755 \else
15756   \renewcommand*\glsgroupskip}{\glspenaltygroupskip}%
15757 \fi
15758 }

```

mLocationHeader

```

15759 \newcommand{\glslongextraNameDescSymLocationHeader}{%
15760 \glslongextraNameDescSymLocationTabularHeader\endhead
15761 \glslongextraNameDescSymLocationTabularFooter\endfoot
15762 }

```

onTabularHeader

```

15763 \newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
15764 \toprule
15765 \glslongextraHeaderFmt\entryname &
15766 \glslongextraHeaderFmt\descriptionname &
15767 \glslongextraHeaderFmt\symbolname &
15768 \glslongextraHeaderFmt\pagelistname\tabularnewline
15769 \midrule
15770 }

```

onTabularFooter

```

15771 \newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
15772 \bottomrule
15773 }

```

me-desc-sym-loc Four columns: name, description and location

```

15774 \newglossarystyle{long-name-desc-sym-loc}%
15775 {%
15776 \ifGlsLongExtraUseTabular
15777 \renewenvironment{theglossary}%
15778   {%
15779     \glslongextraSymLocSetDescWidth
15780     \edef\@glslongextra@begintab{%
15781       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%

```

```

15782         \expandonce\glslongextraNameAlign
15783         \expandonce\glslongextraDescAlign
15784         \expandonce\glslongextraSymbolAlign
15785         \expandonce\glslongextraLocationAlign
15786     }%
15787     \@glslongextra@begintab
15788 }%
15789 {%
15790     \glslongextraNameDescSymLocationTabularFooter
15791     \end{tabular}%
15792 }%
15793 \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationTabularHeader}%
15794 \else
15795 \renewenvironment{theglossary}%
15796 {%
15797     \glspatchLToutput
15798     \glslongextraSymLocSetDescWidth
15799     \edef\@glslongextra@begintab{%
15800         \noexpand\begin{longtable}{%
15801             \expandonce\glslongextraNameAlign
15802             \expandonce\glslongextraDescAlign
15803             \expandonce\glslongextraSymbolAlign
15804             \expandonce\glslongextraLocationAlign
15805         }%
15806         \@glslongextra@begintab
15807     }%
15808     {\end{longtable}}%
15809     \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationHeader}%
15810 \fi
15811 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
15812 \renewcommand{\glossentry}[2]{%
15813     \glslongextraNameFmt{##1} &
15814     \glslongextraDescFmt{##1} &
15815     \glslongextraSymbolFmt{##1}&
15816     \glslongextraLocationFmt{##1}{##2}\tabularnewline
15817 }%
15818 \renewcommand{\subglossentry}[3]{%
15819     \glslongextraSubNameFmt{##1}{##2} &
15820     \glslongextraSubDescFmt{##1}{##2} &
15821     \glslongextraSubSymbolFmt{##1}{##2}&
15822     \glslongextraSubLocationFmt{##1}{##2}{##3}%
15823     \tabularnewline
15824 }%
15825 \ifglsnogroupskip
15826     \renewcommand*\glsgroupskip{}%
15827 \else
15828     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15829 \fi
15830 }

```

meSymDescHeader

```
15831 \newcommand{\glslongextraNameSymDescHeader}{%
15832 \glslongextraNameSymDescTabularHeader\endhead
15833 \glslongextraNameSymDescTabularFooter\endfoot
15834 }
```

scTabularHeader

```
15835 \newcommand{\glslongextraNameSymDescTabularHeader}{%
15836 \toprule
15837 \glslongextraHeaderFmt\entryname &
15838 \glslongextraHeaderFmt\symbolname &
15839 \glslongextraHeaderFmt\descriptionname\tabularnewline
15840 \midrule
15841 }
```

scTabularFooter

```
15842 \newcommand{\glslongextraNameSymDescTabularFooter}{%
15843 \bottomrule
15844 }
```

g-name-sym-desc Three column style with symbol in the second column.

```
15845 \newglossarystyle{long-name-sym-desc}%
15846 {%
15847 \ifGlsLongExtraUseTabular
15848 \renewenvironment{theglossary}%
15849   {%
15850     \glslongextraSymSetDescWidth
15851     \edef\@glslongextra@begintab{%
15852       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15853         \expandonce\glslongextraNameAlign
15854         \expandonce\glslongextraSymbolAlign
15855         \expandonce\glslongextraDescAlign
15856       }}%
15857     \@glslongextra@begintab
15858   }%
15859   {%
15860     \glslongextraNameSymDescTabularFooter
15861     \end{tabular}%
15862   }%
15863 \renewcommand*\glossaryheader{\glslongextraNameSymDescTabularHeader}%
15864 \else
15865 \renewenvironment{theglossary}%
15866   {%
15867     \glspatchLToutput
15868     \glslongextraSymSetDescWidth
15869     \edef\@glslongextra@begintab{%
15870       \noexpand\begin{longtable}{%
15871         \expandonce\glslongextraNameAlign
15872         \expandonce\glslongextraSymbolAlign
```

```

15873         \expandonce\glslongextraDescAlign
15874     }}%
15875     \@glslongextra@begintab
15876     }%
15877     {\end{longtable}}%
15878     \renewcommand*\glossaryheader{\glslongextraNameSymDescHeader}%
15879     \fi
15880     \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15881     \renewcommand\glossentry[2]{%
15882         \glslongextraNameFmt{##1} &
15883         \glslongextraSymbolFmt{##1} &
15884         \glslongextraDescFmt{##1}\tabularnewline
15885     }%
15886     \renewcommand\subglossentry[3]{%
15887         \glslongextraSubNameFmt{##1}{##2} &
15888         \glslongextraSubSymbolFmt{##1}{##2} &
15889         \glslongextraSubDescFmt{##1}{##2}\tabularnewline
15890     }%
15891     \ifglsnogroupskip
15892         \renewcommand*\glsgroupskip{}%
15893     \else
15894         \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15895     \fi
15896 }

```

LocationHeader

```

15897 \newcommand{\glslongextraNameSymDescLocationHeader}{%
15898     \glslongextraNameSymDescLocationTabularHeader\endhead
15899     \glslongextraNameSymDescLocationTabularFooter\endfoot
15900 }

```

onTabularHeader

```

15901 \newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
15902     \toprule
15903     \glslongextraHeaderFmt\entryname &
15904     \glslongextraHeaderFmt\symbolname &
15905     \glslongextraHeaderFmt\descriptionname &
15906     \glslongextraHeaderFmt\pagelistname\tabularnewline
15907     \midrule
15908 }

```

onTabularFooter

```

15909 \newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
15910     \bottomrule
15911 }

```

me-sym-desc-loc Four column style with symbol in the second column.

```

15912 \newglossarystyle{long-name-sym-desc-loc}%
15913 {%

```

```

15914 \ifGlsLongExtraUseTabular
15915   \renewenvironment{theglossary}%
15916     {%
15917       \glslongextraSymLocSetDescWidth
15918       \edef\@glslongextra@begintab{%
15919         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15920           \expandonce\glslongextraNameAlign
15921           \expandonce\glslongextraSymbolAlign
15922           \expandonce\glslongextraDescAlign
15923           \expandonce\glslongextraLocationAlign
15924         }}%
15925       \@glslongextra@begintab
15926     }%
15927     {%
15928       \glslongextraNameSymDescLocationTabularFooter
15929       \end{tabular}%
15930     }%
15931   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationTabularHeader}%
15932 \else
15933   \renewenvironment{theglossary}%
15934     {%
15935       \glspatchLToutput
15936       \glslongextraSymLocSetDescWidth
15937       \edef\@glslongextra@begintab{%
15938         \noexpand\begin{longtable}{%
15939           \expandonce\glslongextraNameAlign
15940           \expandonce\glslongextraSymbolAlign
15941           \expandonce\glslongextraDescAlign
15942           \expandonce\glslongextraLocationAlign
15943         }}%
15944       \@glslongextra@begintab
15945     }%
15946     {\end{longtable}}%
15947   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationHeader}%
15948 \fi
15949 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
15950 \renewcommand\glossentry[2]{%
15951   \glslongextraNameFmt{##1} &
15952   \glslongextraSymbolFmt{##1} &
15953   \glslongextraDescFmt{##1} &
15954   \glslongextraLocationFmt{##1}{##2}\tabularnewline
15955 }%
15956 \renewcommand\subglossentry[3]{%
15957   \glslongextraSubNameFmt{##1}{##2} &
15958   \glslongextraSubSymbolFmt{##1}{##2} &
15959   \glslongextraSubDescFmt{##1}{##2} &
15960   \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
15961 }%
15962 \ifglsnogroupskip

```

```

15963   \renewcommand*{\glsgroupskip}{}%
15964   \else
15965     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15966   \fi
15967 }

```

mDescNameHeader

```

15968 \newcommand{\glslongextraSymDescNameHeader}{%
15969   \glslongextraSymDescNameTabularHeader\endhead
15970   \glslongextraSymDescNameTabularFooter\endfoot
15971 }

```

meTabularHeader

```

15972 \newcommand{\glslongextraSymDescNameTabularHeader}{%
15973   \toprule
15974   \glslongextraHeaderFmt\symbolname &
15975   \glslongextraHeaderFmt\descriptionname &
15976   \glslongextraHeaderFmt\entryname\tabularnewline
15977   \midrule
15978 }

```

meTabularFooter

```

15979 \newcommand{\glslongextraSymDescNameTabularFooter}{%
15980   \bottomrule
15981 }

```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```

15982 \newglossarystyle{long-sym-desc-name}{%
15983 {%
15984   \ifGlsLongExtraUseTabular
15985     \renewenvironment{theglossary}{%
15986       {%
15987         \glslongextraSymSetDescWidth
15988         \edef\@glslongextra@begintab{%
15989           \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15990             \expandonce\glslongextraSymbolAlign
15991             \expandonce\glslongextraDescAlign
15992             \expandonce\glslongextraNameAlign
15993           }}%
15994         \@glslongextra@begintab
15995       }%
15996     }%
15997     \glslongextraSymDescNameTabularFooter
15998     \end{tabular}%
15999   }%
16000   \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
16001   \else
16002     \renewenvironment{theglossary}{%

```

```

16003   {%
16004     \glspatchLToutput
16005     \glslongextraSymSetDescWidth
16006     \edef\@glslongextra@begintab{%
16007       \noexpand\begin{longtable}{%
16008         \expandonce\glslongextraSymbolAlign
16009         \expandonce\glslongextraDescAlign
16010         \expandonce\glslongextraNameAlign
16011       }}%
16012     \@glslongextra@begintab
16013   }%
16014   {\end{longtable}}%
16015   \renewcommand*\glossaryheader{\glslongextraSymDescNameHeader}%
16016   \fi
16017   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16018   \renewcommand\glossentry[2]{%
16019     \glslongextraSymbolFmt{##1} &
16020     \glslongextraDescFmt{##1} &
16021     \glslongextraNameFmt{##1}\tabularnewline
16022   }%
16023   \renewcommand\subglossentry[3]{%
16024     \glslongextraSubSymbolFmt{##1}{##2} &
16025     \glslongextraSubDescFmt{##1}{##2} &
16026     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16027   }%
16028   \ifglsnogroupskip
16029     \renewcommand*\glsgroupskip{}%
16030   \else
16031     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
16032   \fi
16033 }

```

mDescNameHeader

```

16034 \newcommand{\glslongextraLocationSymDescNameHeader}{%
16035   \glslongextraLocationSymDescNameTabularHeader\endhead
16036   \glslongextraLocationSymDescNameTabularFooter\endfoot
16037 }

```

mTabularHeader

```

16038 \newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
16039   \toprule
16040   \glslongextraHeaderFmt\pagelistname &
16041   \glslongextraHeaderFmt\symbolname &
16042   \glslongextraHeaderFmt\descriptionname &
16043   \glslongextraHeaderFmt\entryname\tabularnewline
16044   \midrule
16045 }

```

mTabularFooter


```

16046 \newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
16047 \bottomrule
16048 }

```

c-sym-desc-name Four column style with location list, symbol, description and name.

```

16049 \newglossarystyle{long-loc-sym-desc-name}%
16050 {%
16051 \ifGlsLongExtraUseTabular
16052 \renewenvironment{theglossary}%
16053 {%
16054 \glslongextraSymLocSetDescWidth
16055 \edef\@glslongextra@begintab{%
16056 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16057 \expandonce\glslongextraLocationAlign
16058 \expandonce\glslongextraSymbolAlign
16059 \expandonce\glslongextraDescAlign
16060 \expandonce\glslongextraNameAlign
16061 }}%
16062 \@glslongextra@begintab
16063 }%
16064 {%
16065 \glslongextraLocationSymDescNameTabularFooter
16066 \end{tabular}%
16067 }%
16068 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameTabularHeader}%
16069 \else
16070 \renewenvironment{theglossary}%
16071 {%
16072 \glspatchLToutput
16073 \glslongextraSymLocSetDescWidth
16074 \edef\@glslongextra@begintab{%
16075 \noexpand\begin{longtable}{%
16076 \expandonce\glslongextraLocationAlign
16077 \expandonce\glslongextraSymbolAlign
16078 \expandonce\glslongextraDescAlign
16079 \expandonce\glslongextraNameAlign
16080 }}%
16081 \@glslongextra@begintab
16082 }%
16083 {\end{longtable}}%
16084 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameHeader}%
16085 \fi
16086 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16087 \renewcommand{\glossentry}[2]{%
16088 \glslongextraLocationFmt{##1}{##2} &
16089 \glslongextraSymbolFmt{##1} &
16090 \glslongextraDescFmt{##1} &
16091 \glslongextraNameFmt{##1}\tabularnewline
16092 }%

```

```

16093 \renewcommand{\subglossentry}[3]{%
16094   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16095   \glslongextraSubSymbolFmt{##1}{##2} &
16096   \glslongextraSubDescFmt{##1}{##2} &
16097   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16098 }%
16099 \ifglsnogroupskip
16100 \renewcommand*\glsgroupskip{}%
16101 \else
16102 \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
16103 \fi
16104 }

```

scSymNameHeader

```

16105 \newcommand{\glslongextraDescSymNameHeader}{%
16106 \glslongextraDescSymNameTabularHeader\endhead
16107 \glslongextraDescSymNameTabularFooter\endfoot
16108 }

```

meTabularHeader

```

16109 \newcommand{\glslongextraDescSymNameTabularHeader}{%
16110 \toprule
16111 \glslongextraHeaderFmt\descriptionname &
16112 \glslongextraHeaderFmt\symbolname &
16113 \glslongextraHeaderFmt\entryname\tabularnewline
16114 \midrule
16115 }

```

meTabularFooter

```

16116 \newcommand{\glslongextraDescSymNameTabularFooter}{%
16117 \bottomrule
16118 }

```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```

16119 \newglossarystyle{long-desc-sym-name}%
16120 {%
16121 \ifGlsLongExtraUseTabular
16122 \renewenvironment{theglossary}%
16123 {%
16124 \glslongextraSymSetDescWidth
16125 \edef\@glslongextra@begintab{%
16126 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16127 \expandonce\glslongextraDescAlign
16128 \expandonce\glslongextraSymbolAlign
16129 \expandonce\glslongextraNameAlign
16130 }}%
16131 \@glslongextra@begintab
16132 }%

```

```

16133   {%
16134     \glslongextraDescSymNameTabularFooter
16135     \end{tabular}%
16136   }%
16137   \renewcommand*\glossaryheader{\glslongextraDescSymNameTabularHeader}%
16138 \else
16139 \renewenvironment{theglossary}%
16140   {%
16141     \glspatchLToutput
16142     \glslongextraSymSetDescWidth
16143     \edef\@glslongextra@begintab{%
16144       \noexpand\begin{longtable}{%
16145         \expandonce\glslongextraDescAlign
16146         \expandonce\glslongextraSymbolAlign
16147         \expandonce\glslongextraNameAlign
16148       }%
16149     \@glslongextra@begintab
16150   }%
16151   {\end{longtable}}%
16152   \renewcommand*\glossaryheader{\glslongextraDescSymNameHeader}%
16153 \fi
16154 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16155 \renewcommand{\glossentry}[2]{%
16156   \glslongextraDescFmt{##1} &
16157   \glslongextraSymbolFmt{##1} &
16158   \glslongextraNameFmt{##1}\tabularnewline
16159 }%
16160 \renewcommand{\subglossentry}[3]{%
16161   \glslongextraSubDescFmt{##1}{##2} &
16162   \glslongextraSubSymbolFmt{##1}{##2} &
16163   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16164 }%
16165 \ifglsnogroupskip
16166   \renewcommand*\glsgroupskip{}%
16167 \else
16168   \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
16169 \fi
16170 }

```

scSymNameHeader

```

16171 \newcommand{\glslongextraLocationDescSymNameHeader}{%
16172   \glslongextraLocationDescSymNameTabularHeader\endhead
16173   \glslongextraLocationDescSymNameTabularFooter\endfoot
16174 }

```

meTabularHeader

```

16175 \newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
16176   \toprule
16177   \glslongextraHeaderFmt\pagelistname &

```

```

16178 \glslongextraHeaderFmt\descriptionname &
16179 \glslongextraHeaderFmt\symbolname &
16180 \glslongextraHeaderFmt\entryname\tabularnewline
16181 \midrule
16182 }

```

meTabularFooter

```

16183 \newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
16184 \bottomrule
16185 }

```

c-desc-sym-name Four column style with location list, description, symbol and name.

```

16186 \newglossarystyle{long-loc-desc-sym-name}%
16187 {%
16188 \ifGlsLongExtraUseTabular
16189 \renewenvironment{theglossary}%
16190 {%
16191 \glslongextraSymLocSetDescWidth
16192 \edef\@glslongextra@begintab{%
16193 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16194 \expandonce\glslongextraLocationAlign
16195 \expandonce\glslongextraDescAlign
16196 \expandonce\glslongextraSymbolAlign
16197 \expandonce\glslongextraNameAlign
16198 }}%
16199 \@glslongextra@begintab
16200 }%
16201 {%
16202 \glslongextraLocationDescSymNameTabularFooter
16203 \end{tabular}}%
16204 }%
16205 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameTabularHeader}%
16206 \else
16207 \renewenvironment{theglossary}%
16208 {%
16209 \glspatchLToutput
16210 \glslongextraSymLocSetDescWidth
16211 \edef\@glslongextra@begintab{%
16212 \noexpand\begin{longtable}{%
16213 \expandonce\glslongextraLocationAlign
16214 \expandonce\glslongextraDescAlign
16215 \expandonce\glslongextraSymbolAlign
16216 \expandonce\glslongextraNameAlign
16217 }}%
16218 \@glslongextra@begintab
16219 }%
16220 {\end{longtable}}%
16221 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameHeader}%
16222 \fi

```

```

16223 \renewcommand*\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
16224 \renewcommand{\glossentry}[2]{%
16225   \glslongextraLocationFmt{##1}{##2} &
16226   \glslongextraDescFmt{##1} &
16227   \glslongextraSymbolFmt{##1} &
16228   \glslongextraNameFmt{##1}\tabularnewline
16229 }%
16230 \renewcommand{\subglossentry}[3]{%
16231   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16232   \glslongextraSubDescFmt{##1}{##2} &
16233   \glslongextraSubSymbolFmt{##1}{##2} &
16234   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16235 }%
16236 \ifglsnogroupskip
16237   \renewcommand*\glsgroupskip}{}%
16238 \else
16239   \renewcommand*\glsgroupskip}{\glspenaltygroupskip}%
16240 \fi
16241 }

```

5 topic styles (glossary-topic.sty)

5.1 Package Initialisation and Options

Provides “topic” styles where top-level entries are considered a topic.

```
16242 \NeedsTeXFormat{LaTeX2e}
16243 \ProvidesPackage{glossary-topic}[2019/04/09 v1.41 (NLCT)]
```

Load required package.

```
16244 \RequirePackage{multicol}
```

The top-level entries act like headers. If the top-level entry has a description it's placed below the name.

topic

```
16245 \newglossarystyle{topic}{%
16246   \renewenvironment{theglossary}%
16247   {%
16248     \glstopicInit
16249     \def\glstopic@prechildren{}%
16250     \def\glstopic@prevlevel{-1}%
16251   }%
16252   {\par}%
16253   \renewcommand*\glossaryheader{}%
16254   \renewcommand*\glsgroupheading[1]{%
16255     \def\glstopic@prevlevel{-1}%
16256     \glstopicGroupHeading{##1}%
16257   }%
16258   \renewcommand{\glossentry}[2]{%
16259     \hangindent0pt\relax
16260     \parindent\glstopicParIndent\relax
16261     \glstopicItem{##1}{##2}%
16262     \ifglshasdesc{##1}%
16263     {%
16264       \def\glstopic@prechildren{}%
16265     }%
16266     {%
16267       \def\glstopic@prechildren{\nopagebreak}%
16268     }%
16269   }%
16270   \renewcommand{\subglossentry}[3]{%
16271     \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
```

```

16272 \def\glstopic@prevlevel{##1}%
16273 \glstopicAssignSubIndent{##1}%
16274 \glstopicSubItem{##1}{##2}{##3}%
16275 }%
16276 \renewcommand*\glsgroupskip{}%
16277 }

```

`\glstopicGroupHeading` `\glstopicGroupHeading{<group label>}`

May be redefined if letter group headings are required. For example:

```

\renewcommand*\glstopicGroupHeading[1]{%
  \glstrgetgrouptitle{#1}{\thisgrptitle}%
  \section*{\thisgrptitle}%
}
16278 \newcommand*\glstopicGroupHeading[1]{}

```

`\glstopicItem` `\glstopicItem{<label>}{<location list>}`

```

16279 \newcommand*\glstopicItem[2]{%
16280 \glsparskip\glstopicPreSkip\glsparskip\noindent
16281 \glstopicMarker{#1}%
16282 \glstopicTitleFont
16283 {%
16284 \glstryitem{#1}\glstarget{#1}{\glstopicTitle{#1}}%
16285 }%
16286 \ifglshasdesc{#1}%
16287 {\glsparskip\nobreak\glstopicMidSkip\glsparskip\nobreak
16288 \@afterheading\glstopicDesc{#1}\glsparskip\glstopicPostSkip}%
16289 {\glsparskip\nobreak\glstopicPostSkip}%
16290 \glstopicLoc{#1}{#2}%
16291 }

```

`\glstopicMarker` May be used to insert a bookmark etc if required.

```

16292 \newcommand*\glstopicMarker[1]{}

```

`\glstopicName`

```

16293 \newcommand*\glstopicTitle[1]{\Glossentryname{#1}%
16294 \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}}%
16295 }

```

`\glstopicTitleFont`

```

16296 \newcommand*\glstopicTitleFont[1]{\textbf{\large #1}}

```

`\glstopicDesc`
16297 `\newcommand*{\glstopicDesc}[1]{\Glossentrydesc{#1}\glspostdescription}`

`\glstopicLoc`
16298 `\newcommand*{\glstopicLoc}[2]{}`

`stopicParIndent`
16299 `\newlength\glstopicParIndent`
16300 `\setlength\glstopicParIndent{20pt}`

`stopicSubIndent`
16301 `\newlength\glstopicSubIndent`
16302 `\setlength\glstopicSubIndent{20pt}`

`\glstopicInit`
16303 `\newcommand{\glstopicInit}{}`

`AssignSubIndent` `\glstopicAssignSubIndent{<level>}`

Used to set the indentation for sub-levels.

```
16304 \newcommand*{\glstopicAssignSubIndent}[1]{%
16305   \par
16306   \parindent\dimexpr#1\glstopicSubIndent-\glstopicSubIndent\relax
16307   \glstopicAssignWidest{#1}%
16308   \hangindent\dimexpr\parindent+\glstopicwidest\relax
16309 }
```

`\glstopicwidest`
16310 `\newlength\glstopicwidest`

`opicAssignWidest` `\glstopicAssignWidest{<level>}`

Used in the definition of `\glstopicAssignSubIndent` to set the indentation from the widest name for the given level. This will require `glossary-tree` to set the values.

```
16311 \newcommand*{\glstopicAssignWidest}[1]{%
16312   \ifcsundef{@glswidestlength\romannumeral#1}%
16313   {%
16314     \ifcsdef{@glswidestname\romannumeral#1}%
16315     {%
16316       \settowidth{\glstopicwidest}{%
16317         \glstopicSubNameFont{\csuse{@glswidestname\romannumeral#1}}%
16318         \glstopicSubItemSep
```



```

16319     }%
16320     }%
16321     {\setlength{\glstopicwidest}{0pt}}%
        Save the value so that it doesn't have to keep being recalculated.
16322     \csedef{@glswidestlength\romannumeral#1}{\the\glstopicwidest}%
16323     }%
16324     {\setlength{\glstopicwidest}{\csuse{@glswidestlength\romannumeral#1}}}%
16325 }

```

glstopicPreSkip

```
16326 \newcommand*{\glstopicPreSkip}{\medskip}
```

glstopicMidSkip

```
16327 \newcommand*{\glstopicMidSkip}{\smallskip}
```

lstopicPostSkip

```
16328 \newcommand*{\glstopicPostSkip}{\smallskip}
```

\glstopicSubItem

```
\glstopicSubItem{<level>}{<label>}{<location list>}
```

```

16329 \newcommand*{\glstopicSubItem}[3]{%
16330   \glstopicSubItemBox{#1}{\glstopicSubNameFont{\glstentryitem{#2}}%
16331     \glstarget{#2}{\glossentryname{#2}}}%
16332   \glstopicSubItemSep
16333   }%
16334   \ifglshassymbol{#2}{(\glossentrysymbol{#2})\space}{}%
16335   \ifglshasdesc{#2}%
16336     {\glossentrydesc{#2}\glspostdescription\glstopicSubPreLocSep}{}%
16337   \glstopicSubLoc{#2}{#3}%
16338 }

```

topicSubItemSep

```
16339 \newcommand*{\glstopicSubItemSep}{\quad}
```

glstopicSubItemBox

```
\glstopicSubItemBox{<level>}{<text>}
```

```

16340 \newcommand*{\glstopicSubItemBox}[2]{%
16341   \ifdim\glstopicwidest>0pt\relax\makebox[\glstopicwidest][l]{#2}\else#2\fi
16342 }

```