

glossaries-extra.sty v1.38: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-12-01

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	42
1.3.1	Existence Checks	47
1.3.2	Document Definitions	56
1.3.3	Existing Glossary Style Modifications	62
1.3.4	Entry Formatting, Hyperlinks and Indexing	67
1.3.5	Entry Counting	104
1.3.6	Acronym Modifications	120
1.3.7	Indexing and Displaying Glossaries	122
1.4	Link Counting	160
1.5	Integration with glossaries-accsupp	162
1.6	Categories	175
1.7	Abbreviations	202
1.7.1	Abbreviation Styles Setup	222
1.7.2	Predefined Styles (Default Font)	225
1.7.3	Predefined Styles (Small Capitals)	242
1.7.4	Predefined Styles (Fake Small Capitals)	256
1.7.5	Predefined Styles (Emphasized)	270
1.7.6	Predefined Styles (User Parentheses Hook)	292
1.7.7	Predefined Styles (Hyphen)	301
1.7.8	Predefined Styles (No Short on First Use)	315
1.8	Using Entries in Headings	318
1.9	Multi-Lingual Support	337
1.10	glossaries-extra-bib2gls.sty	338
2	Style Adjustments (glossaries-extra-stylemods.sty)	383
2.1	Package Initialisation	383
2.2	List-Like Styles	384
2.3	Longtable Styles	387
2.4	Long Ragged Styles	389
2.5	Supertabular Styles	391
2.6	Super Ragged Styles	393
2.7	Inline Style	395
2.8	Tree Styles	395
2.9	Multicolumn Styles	413

3	bookindex style (glossary-bookindex.sty)	419
3.1	Package Initialisation and Options	419
4	longextra styles (glossary-longextra.sty)	425
4.1	Package Initialisation and Options	425
	Glossary	448
	Change History	449
	Index	470

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/12/01 1.38 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

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

```
6 {%
```

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

```
7 \newcommand{\glxstr@doption}[1]{\setupglossaries{#1}}%
```

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

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

```
11 \newcommand{\glxstr@doption}[1]{%
```

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

```
13 }%
```

Set the defaults.

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

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

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

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

```
18 {}%
```

```
19 {%
```

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

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

```
22 {}%
```

```
23 }%
```

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

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

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

```
27 }
```

```
28 }
```

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

```

undefaction

```

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

\@glsxtr@record Does nothing by default.

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

\glsxtr@recordsee Does nothing by default.

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

\gltnumberformat

```
79 \newcommand*{\@glsxtr@defaultnumberformat}{\glsnnumberformat}%

```

\gltnumberFormat

```

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

```

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

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

```

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

Save the label in case it's needed.

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

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

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

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

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

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

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

Implement any pre-key settings.

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

Assign keys.

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

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

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

Check post-key hook.

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

Increment associated counter.

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

Check if noindex option has been used.

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

Check if thevalue has been set.

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

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

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

```

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

```

Save the entry counter.

```

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

```

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

```

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

```

No need to escape special characters.

```

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

```

glslink@prekeys

```

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

```

glslink@postkeys

```

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

```

glossadd@prekeys

```

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

```

glossadd@postkeys

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`r@recordcounter`

```

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

```

`p@recordcounter`

```

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

p@recordcounter

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

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

srtglossaryunit

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

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

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

addloclistfield

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

```

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

```

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

Add a key to store the group heading.

```

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

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

```

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

alsoindex

```

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

only

```

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

nameref

```

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

only

```

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

```

318 \fi
319 }

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

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

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

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

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

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

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

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

```

```

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

```

Only record (don't index).

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

Need to provide conditional commands that are backward compatible:

`\if@glxstrdocdef`

```

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

```

`\glxstrdocdeftrue`

```

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

```


sxtrdocdeffalse

```
383 \newcommand*{\@glxtrdocdeffalse}{\def\@glxtr@docdefval{0}}
```

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

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

ocdefrestricted

```
394 \newcommand*{\if@glxtrdocdefrestricted}{\ifnum\@glxtr@docdefval>1 }
```

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

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

indexcrossrefs Automatically index cross references at the end of the document

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

Switch off since this can increase the build time.

```
402 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
403 \newcommand*{\@glxtr@autoindexcrossrefs}{\@glxtrindexcrossrefstrue}
```

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

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

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

```

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

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

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

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

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

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

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

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

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

```

```

436 }%
437 }

```

nopostdot Needs to redefine \@glsxtr@defpostpunc

```

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

```

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

```

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

```

glsxtrabbrvtype Glossary type for abbreviations.

```

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

```

bbreviationsdef Set by abbreviations option.

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

bbreviationsdef

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

   If the acronym option hasn't been used, change \acronymtype to \glxtrabbrvtype.
488   \ifglssacronym
489   \else
490     \renewcommand*{\acronymtype}{\glxtrabbrvtype}%
491   \fi
492 }%
```

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

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

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses \newcommand instead of \let as a safety feature (except for \newabbr which is also provided with \GlsXtrDefineAcShortcuts).

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

```

```

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

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

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

```

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

```

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

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

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

```

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

```

552 \newcommand*{\GlsXtrDefineOtherShortcuts}{%

```

```

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

```

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

@setupshortcuts Command used to set the shortcuts option.

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

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

```
565 \newcommand*{\@glstr@shortcutsval}{\ifglscrshortcuts acro\else none\fi}%

```

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

```

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

```

```

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

```

Leave none and false as last option.

```

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

```

lsxtr@doaccsupp

```

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

```

accsupp If accsupp, load glossaries-accsupp package.

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

xtr@redefstyles

```

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

```

stylemods

```

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

```



```

666 }%
667 }

```

glsxtr@do@style

```

668 \newcommand*{\@glsxtr@do@style}{}

```

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

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

672 \setglossarystyle{#1}%

```

```

673 }%

```

```

674 }

```

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

```

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

```

ocationHyperlink

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

The first two arguments are always control sequences.

```

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

```

```

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

```

```

678 }

```

cationhyperlink

```

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

```

```

680 \pageref{wrglossary.#3}%

```

```

681 }

```

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

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

```

682 \@glsxtr@declareoption{indexcounter}{%

```

```

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

```

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

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

```

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

```

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

```

`\sxttrwrglossmark` Does nothing by default.

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

```

`debug` Provide extra debug options.

```

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

```

```

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

```

Pass all other options to glossaries.

```

729 \DeclareOptionX*{%
730 \expandafter\glstr@dooption\expandafter{\CurrentOption}}

```

Process options.

```
731 \ProcessOptionsX
```

Load glossaries if not already loaded.

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

Load the glossaries-accsupp package if required.

```
733 \@glstr@doaccsupp
```

Redefine \glspostdescription if required.

```
734 \@glstr@defpostpunc
```

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

```

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

```

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

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

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

```

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

```

```

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

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

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

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

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

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

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

                  Setup shortcuts if required.
784 \@glsxtr@setupshortcuts

```

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

```

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

```

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

```

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

```

Now define the user command:

```

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

```

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

```

798 \let\glxtr@org@@do@wrglossary@@do@wrglossary

```

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

```

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

```

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

```

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

```

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

```

805 \let\@gl@s@saveentrycounter\glxtr@indexonly@saveentrycounter

```

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

```

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

```

```

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

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

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

```

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

sxtrdialecthook

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

```

Set up record option if required.

```
836 \glsxtr@setup@record

```

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

```

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

```

1.2 Extra Utilities

nusedOrUndefined

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

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

```

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

```

xtrifemptyglossary

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

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

```

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

```

xtrifkeydefined

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

```

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

```

providestoragekey

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

```

859 \newcommand*{\glsxtrprovidestoragekey}{%
860   \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
861 }

```

vide@storagekey

Unstarred version.

```

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

```

```

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

```

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

```

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

```

Provide the no-link command if not already defined.

```

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

```

`\provide@storagekey` Starred version.

```

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

```

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

`\GlsXtrFmtField`

```

897 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

898 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```


`\@glsxtrfmt` Unstarred form.

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

`\s@glsxtrfmt` Starred form.

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

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

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

`\@@glsxtrfmt` Actual inner working.

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

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

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

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

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

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

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

```

\glxstrentryfmt No link or indexing.
930 \ifdef\teorpdfstring
931 {
932   \newcommand*\glxstrentryfmt[2]{%
933     \teorpdfstring{\@glxstrentryfmt{#1}{#2}}{#2}%
934   }
935 }
936 {
937   \newcommand*\glxstrentryfmt{\@glxstrentryfmt}
938 }

```

```

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

```

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

```

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

```

glxstrfieldlistgadd Similarly but uses `\listcsgadd`.

```

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

```

glxstrfieldlistseadd Similarly but uses `\listcseadd`.

```

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

```

glxstrfieldlistxadd Similarly but uses `\listcsxadd`.

```

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

```

Now provide commands to iterate over these lists.

fielddolistloop

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

fieldforlistloop

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

List element tests:

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

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

rfieldxifinlist Expands item.

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

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

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

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

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

lsxtrifhasfield Unstarred version adds grouping.

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

lsxtrifhasfield Starred version omits grouping.

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

rIfFieldNonZero Designed for numeric fields.

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

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

Designed for numeric fields.

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

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

Designed for numeric fields.

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

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

Just uses \ifcsundef.

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

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

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

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

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

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

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

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

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

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

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

`etfieldifexists`

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

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

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

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

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

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

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

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

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

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

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

`xGlsXtrSetField`

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

`eGlsXtrSetField`

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

`XtrIfFieldEqStr`

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

```

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

```

`rIfFieldEqXpStr` Like the above but first expands the string.

```

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

```

`fXpFieldEqXpStr` Like the above but also expands the field value.

```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

No tracked dialect found for the root language.

```

1127      }%
1128      {%

```

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

```

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

```

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

```

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

```

Does a caption hook exist for this?

```

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

```

No mapping. Try root language label instead.

```

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

```

No mapping. Try root language label instead.

```

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

```



```

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

```

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

```

1172 \newcommand*{\GlsXtrForeignTextField}{userii}

```

`\nDialectWarning`

```

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

```

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

```

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

```

```

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

```

lossary preamble

```

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

```

lossary preamble

```

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

```

1.3 Modifications to Commands Provided by glossaries

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

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

In the event that undefined entries should trigger a warning rather than an error, `\ifglused` needs to be modified to check for existence. If the boolean variable is undefined, then its state is indeterminate and is neither true nor false, so neither *<true part>* nor *<false>* part will be performed if *<label>* is undefined.

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

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

`\newglossaryentry`

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

`\newglossaryentry` Starred version.

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

`\newglossaryentry` Unstarred version.

```
1251 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1252   \glstoifnoexists{#1}%
1253   {%
1254     \bgroup
1255     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1256     \long\def\@newglossaryentryprehook{%
1257       \long\def\@glo@desc{#3\glstrpostlongdescription}%
1258     }
1259   }
```

```

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

```

The following is different from the base glossaries.sty:

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

```

1268 \newcommand*{\glsxtrpostlongdescription}{\leavevmode\unskip\nopostdesc}

```

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

`\ignoredglossary` Redefine to check for star.

```

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

```

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

```

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

```

```

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

```

ignoredglossary Starred form.

```

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

```

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

```

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

```

```

1340 }
1341 }

```

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

```

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

```

ignoredglossary Unstarred version.

```

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

```

ignoredglossary Starred form.

```

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

```

```

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

```

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

```

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

```

1.3.1 Existence Checks

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

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

```


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

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

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

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

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

`\glxtr@usesee`

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

`\@glxtr@usesee`

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

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

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

`lsseeitemformat` glossaries originally defined `\glsseeitemformat` to use `\glsentryname` but in v3.0 this was switched to use `\glsentrytext` due to problems occurring with the name field being sanitized. Since this is no longer a problem, glossaries-extra restores the original definition as it

makes more sense to use the name in the cross-reference list. This still uses `\glsaccesstext` for abbreviations.

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

`\glsxtrhiername` `\glsxtrhiername{\label}`

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

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

`\Glsxtrhiername` `\Glsxtrhiername{\label}`

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

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

`\GlsXtrhiername` `\GlsXtrhiername{\label}`

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

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

```

1519  {%
1520    \glxtrifhasfield{parent}{#1}%
1521    {\GlsXtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1522    }%
1523    \ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}%
1524  }%
1525 }

```

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

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

```

1526 \newcommand*{\GlsXtrhiername}[1]{%
1527   \glsdoidexists{#1}%
1528   {%
1529     \glxtrifhasfield{parent}{#1}%
1530     {%
1531       \GlsXtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1532       \ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}%
1533     }%
1534     {\ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}}%
1535   }%
1536 }

```

`\GlsXTRhiername` `\GlsXTRhiername{<label>}`

As above but displays all names in all-caps.

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

1564 \providecommand{\seealso}{see also}

```

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

```

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

```

Append to class list.

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

```

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

```

Add to the key mappings.

```

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

```

Set the default value.

```

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

```

Assign the field values.

```

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

```

```

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

```

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

```

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

```

Provide user-level commands to access the values.

`\glstralias`

```

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

```

`trseealsolabels`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

trseealsolabels
1649 \glsaddstoragekey*{seealso}{\glxtrseealsolabels}

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

ryentryposthook Append to the hook to check for the alias and seealso keys.
1650 \appto\@newglossaryentryposthook{%
1651 \ifcsvoid{glo@\@glo@label @alias}%
1652 {%
1653 \ifcsvoid{glo@\@glo@label @seealso}%
1654 }%
1655 {%
1656 \edef\@do@glsee{\noexpand\glxtrindexseealso
1657 {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1658 \@do@glsee
1659 }%
1660 }%
1661 {%

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

1671 }

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

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

```

```

1681 }%
1682 }

```

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

```

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

```

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

```

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

```

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

```

1700 \def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
1701   \@for\@glsxtr@label:=#1\do
1702   {%
1703     \ifglsused{\@glsxtr@label}{}%
1704     {%
1705       \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1706       \glsunset{\@glsxtr@label}%
1707       \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1708     }%
1709   }%
1710 }

```

glsxtrunusedformat

```

1711 \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.

```

1712 \ifdef\gls@begindocdefs
1713 {%

```



```

1714 \renewcommand*{\gls@begindocdefs}{%
1715   \ifnum\@glxtr@docdefval=1\relax
1716     \@gls@enablesavenonumberlist
1717     \edef\@gls@restoreat{%
1718       \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1719     \makeatletter
1720     \InputIfFileExists{\jobname.glsdefs}{\relax}%
1721     \@gls@restoreat
1722     \undef\@gls@restoreat
1723     \gls@defdocnewglossaryentry
1724   \else
1725     \ifnum\@glxtr@docdefval=3\relax

```

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

```

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

```

Write all currently defined entries.

```

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

```

m@glossaryentry

```

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

```

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

```

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

```

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

```

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

```

Check for docdefs=restricted:

```

1764 \ifglsxtrdocdefrestricted

```

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

```

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

```

Disable document definitions.

```

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

```

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

```

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

```

```

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

```

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

```

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

```

Restricted mode just needs to allow the see value.

```

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

```

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

`rEnableOnTheFly`

```

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

```

`rEnableOnTheFly`

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

```

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

```

```

1823 \fi
1824 }

```

sxtrstarflywarn

```

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

```

rEnableOnTheFly

```

1830 \newcommand*{\@GlsXtrEnableOnTheFly}{%

```

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

These definitions are all assigned the category given by:

\glxtrcat

```

1831 \newcommand*{\glxtrcat}{general}

```

\glxtr

```

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

```

\@glxtr

```

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

```

\Glsxtr

```

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

```

\@Glsxtr

```

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

```

```

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

```

\glstrpl

```

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

```

\@glstrpl

```

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

```

\Glsxtrpl

```

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

```

\@Glsxtrpl

```

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

```

\GlsXtrWarning

```

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

```

Disable commands after the glossary:

```

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

```

abledflycommand

```

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

```

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

```

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

```

1.3.3 Existing Glossary Style Modifications

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

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

```

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

```

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

etglossarystyle

```

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

```

```

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

```

Only set the current style if it exists.

```

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

```

In case we have an old version of glossaries:

```

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

```

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

```

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

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

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

```

```

1963 }
1964 {}%

```

and for \glspagelistwidth:

lspagelistwidth

```

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

```

aryentrynumbers Has the nonnumberlist option been used?

```

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

```

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

```

1998 \newcommand*{\GlsXtrFormatLocationList}[1]{#1}

```

Sometimes users want to prefix the location list with “page”/“pages”. The simplest way to determine if the location list consists of a single location is to check for instances of \delimN

or `\delimR`, but this isn't so easy to do as they might be embedded inside the argument of formatting commands. With a bit of trickery we can find out by adjusting `\delimN` and `\delimR` to set a flag and then save information to the auxiliary file for the next run.

`ePreLocationTag`

```

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

```

`glstrpreloctag`

```

2019 \newcommand*{\@glstrpreloctag}{%
2020   \let\@glstr@org@delimN\delimN
2021   \let\@glstr@org@delimR\delimR
2022   \let\@glstr@org@glsignore\glsignore
2023   \gdef is required as the delimiters may occur inside a scope.
2024   \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2025   \renewcommand*{\delimN}{%
2026     \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2027     \@glstr@org@delimN}%
2028   \renewcommand*{\delimR}{%
2029     \gdef\@glstr@thisloctag{\@glstr@pagetag}%
2030     \@glstr@org@delimR}%
2031   \renewcommand*{\glsignore}[1]{%
2032     \gdef\@glstr@thisloctag{\relax}%
2033     \@glstr@org@glsignore{##1}}%
2034   \@glstr@doloctag
2035 }

```

`glstrpreloctag`

```

2035 \newcommand*{\@glstrpreloctag}{}

```

@glsxtr@pagetag

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

glsxtr@pagetag

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

lsxtrpostloctag

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

lsxtrpostloctag

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

lsxtr@preloctag

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

glsxtr@doloctag

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

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

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

1.3.4 Entry Formatting, Hyperlinks and Indexing

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

```

2067 \renewcommand*{\glsentryfmt}{%
2068   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
2069   \glsifregular{\glslabel}%
2070   {\glstrregularfont{\glsgenentryfmt}}%
2071   {%
2072     \ifglshasshort{\glslabel}%
2073     {\glstrabbreviationfont{\glstrgenabbrvfmt}}%
2074     {\glstrregularfont{\glsgenentryfmt}}%
2075   }%
2076 }
```

`glstrregularfont` Font used for regular entries.

```
2077 \newcommand*{\glstrregularfont}[1]{#1}
```

`glstrabbreviationfont` Font used for abbreviation entries.

```
2078 \newcommand*{\glstrabbreviationfont}[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 `\glstrifwasfirstuse` etc to allow the postlink hook to work better. This now has an optional argument that sets up the defaults.

```
2079 \renewcommand{\@gls@field@link}[4][]{%}
```

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

```

2080 \glstr@record{#2}{#3}{glslink}%
2081 \glstoifexists{#3}%
2082 {%
```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

Post-link hook commands need initialising.

```

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

```

sxtrinitwrgloss Set the default if the wrgloss is omitted.

```

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

```

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

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

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

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

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

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

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

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

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

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

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

nithyperoutside Set the default if the hyperoutside is omitted.

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

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

```
2199 \newcommand*{\glstr@inc@linkcount}{}

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

sXtrExpandedFmt Helper command that (protected) fully expands second argument and then applies it to the
first, which must be a command that takes a single argument.
2201 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2202   \protected@edef\@glstr@tmp{#2}%
2203   \expandafter#1\expandafter{\@glstr@tmp}%
2204 }

tion@counter@or If in a numbered equation, change the counter to equation. This can be overridden by explic-
itly setting the counter in the optional argument of commands like \gls and \glslink.
2205 \newcommand*{\@glstr@use@equation@counter}{%
2206   \@glstr@ifnum@mmode{\def\@gls@counter{equation}}{}}%
2207 }

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

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

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

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

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

```

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

```

Save current value of `\glolinkprefix`:

```

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

```

Initialise `\@glsxtr@local@textformat`

```

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

```

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

```

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

```

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

```

2236 \glsxtrinitwrgloss

```

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

```

2237 \glsxtrinithyperoutside

```

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

```

2238 \@gls@setdefault@glslink@opts

```

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

```

2239 \glsxtr@inc@linkcount

```

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

```

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

```

As the original definition.

```

2243 \do@glsdisablehyperinlist
2244 \do@gls@link@checkfirsthyper

```

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

```

2245 \glslinkpresetkeys

```

Set options.

```

2246 \setkeys{glslink}{#1}%

```

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

```

2247 \glsxtr@do@autoadd{glslink}%

```


User hook after options are set:

```
2248 \glslinkpostsetkeys
```

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

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

Check if the textformat key has been used.

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

Check textformat attribute (new to v1.21).

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

Do write if it should occur before the link text:

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

Do the link text:

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

```

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

```

Do write if it should occur after the link text:

```

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

```

Restore original value of \glolinkprefix:

```

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

```

As the original definition:

```

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

```

```

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

```

```

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

```

\saddpresetkeys

```

2304 \newcommand*{\glsaddpresetkeys}{ }

```

\saddpostsetkeys

```

2305 \newcommand*{\glsaddpostsetkeys}{ }

```

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

```

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

```

Implement any default settings (before options are set)

```

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

```

Implement any default settings (after options are set)

```

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

```

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

```

2330 \@gls@setsort{#2}%

```

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

```

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

```

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

```

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

```

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

```

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

```

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

assignfieldfont

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`\ecknohyperfirst`

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

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

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

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

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

\@glsdesc@

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

\@Glsdesc@ First letter uppercase version.

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

\@GLSdesc@ All uppercase version.

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

@glsdescplural@ No case-changing version.

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

```

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

```

@Glsdescplural@ First letter uppercase version.

```

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

```

@GLSdescplural@ All uppercase version.

```

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

```

\@glssymbol@

```

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

```

\@Glsymbol@ First letter uppercase version.

```

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

```

\@GLSsymbol@ All uppercase version.

```

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

```

lssymbolplural@ No case-changing version.

```

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

```



```

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

```

lssymbolplural@ First letter uppercase version.

```

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

```

LSsymbolplural@ All uppercase version.

```

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

```

\@Glsuseri@ First letter uppercase version.

```

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

```

\@GLSuseri@ All uppercase version.

```

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

```

\@Glsuserii@ First letter uppercase version.

```

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

```

\@GLSuserii@ All uppercase version.

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

\@Glsuseriii@ First letter uppercase version.

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

\@GLSuseriii@ All uppercase version.

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

\@Glsuseriv@ First letter uppercase version.

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

\@GLSuseriv@ All uppercase version.

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

\@Glsuserv@ First letter uppercase version.

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

\@GLSuserv@ All uppercase version.

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

```

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

```

\@Glsuservi@ First letter uppercase version.

```

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

```

\@GLSuservi@ All uppercase version.

```

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

```

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

\@acrshort No case change.

```

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

```

\@Acrshort First letter uppercase.

```

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

```

```

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

```

\@ACRshort All uppercase.

```

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

```

\@acrshortpl No case change.

```

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

```

\@Acrshortpl First letter uppercase.

```

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

```

```

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

```

\@ACRshortpl All uppercase.

```

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

```

\@acrlong No case change.

```

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

```

\@Acrlong First letter uppercase.

```

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

```

```

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

```

\@ACRlong All uppercase.

```

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

```

\@acrlongpl No case change.

```

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

```

\@Acrlongpl First letter uppercase.

```

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

```

```

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

```

\@ACRlongpl All uppercase.

```

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

```

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

\@glsaddkey

```

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

```

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

```

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

```

```

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

```

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

```

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

```

Finally the all caps version (modified):

```

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

```



```

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

```

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

```

2878 \providecommand*{\@gls@link@nocheckfirsthyper}{}

```

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

```

2879 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2880 \renewcommand*{\@gls@link@checkfirsthyper}{%

```

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

```

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

```

Store the category label for convenience.

```

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

```

```

2893 }%
2894 \glslinkcheckfirsthyperhook
2895 }

```

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

```

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

```

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

```

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

```

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

lt@gl:link@opts

```

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

```

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

```

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

```

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

```

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

```

setaliasnoindex

```

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

```

xtrindexaliased

```

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

```

xtrindexaliased

```

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

```

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

```

2949 \let\glxtrindexaliased\@no@glxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

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

```

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

```

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

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

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

```

```

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

```

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

```

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

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

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

```

`xtr@do@@wrindex`

```

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

```

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

```

2985 \newcommand*{\glxtrdowrglossaryhook}[1]{%

```

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

```

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

```

```

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

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

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

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

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

org@dohyperlink
3015 \let\glsxtr@org@dohyperlink\glsdohyperlink

glsnavhyperlink  Now that \glsdohyperlink (used by \@glslink) references \glslabel it's necessary to
                  patch \glsnavhyperlink to avoid using it (since \glslabel won't be defined). This means
                  temporarily redefining \glsdohyperlink to its original definition.
                  This command is provided by glossary-hypernav so it may not exist.
3016 \ifdef\glsnavhyperlink
3017 {
3018   \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
3019     \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
      Scope:
3020     {%
3021       \let\glsdohyperlink\glsxtr@org@dohyperlink

```

```

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

```

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

```

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

```

Check for alias.

```

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

```

Redirect link to the alias target.

```

3058   \glxtrhyperlink

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`gl:sdonohyperlink` This command was only introduced in glossaries v4.20, so it may not be defined (therefore use `\def`). For older glossaries versions, this won't be used if `hyperref` hasn't been loaded,

which means the indexing will still take place. The generated text is scoped (the link text in `\hyperlink` is also scoped, so it's consistent).

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

lsxtr@p@longpl@
3181 \def\@Lsxtr@p@longpl@#1#2[#3]{\Glsentrylongpl{#2}#3}

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

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

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

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

```

```

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

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

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

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

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

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

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

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

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

Commands to minimise conflict.

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

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

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

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

```

Add scope.

```

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

```

\@glsxtrp

```

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

```

\@Glsxtrp

```

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

```

\@GLSxtrp

```

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

```

```

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

```

\glsxtr@entry@p

```

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

```

\glsxtrp Not robust as it needs to expand somewhat.

```

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

```

```

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

```

Provide short synonyms for the most common option.

`\glsp`

```

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

```

`\glsp`

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

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

```

1.3.5 Entry Counting

The (use) entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine \gls etc to use \cgl instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceeding the specified threshold. For link counting, see Section 1.4.

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be

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

`\@glsxtr@unset` Global unset.

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

`\@glsunset` Global unset.

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

`glsxtrpostunset`

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

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

`tUnsetBuffering`

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

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

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

`tUnsetBuffering` Starred version checks for duplicates.

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

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

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

`fer@nodup@unset` Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using `\xifinlist` as the added complexity might cause problems that the buffering is trying to overcome.)

```

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

```

pUnsetBuffering

```

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

```

pUnsetBuffering Unstarred form (global unset).

```

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

```

pUnsetBuffering Starred form (local unset).

```

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

```

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

```

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

```

\@glsllocalunset Local unset.

```

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

```

rpostlocalunset

```

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

```

\@glrsreset Global reset.

```

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

```

glxtrpostreset

```

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

```

`\@glslocalreset` Local reset.

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

`trpostlocalreset`

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

`sllocalreseteach` Locally reset a list of entries.

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

`sllocalunseteach` Locally unset a list of entries.

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

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

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

Enable entry counting:

```
3498   \glsenableentrycount
```

Redefine `\gls` etc:

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

Set the entrycount attribute:

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

In case this command is used again:

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

ycountunsetattr

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

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

nableentrycount

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

Enable new fields:

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

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

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

New commands to access new fields:

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

Adjust post unset and reset:

```

3540 \let\@glxstr@entrycount@org@unset\glxstrpostunset
3541 \renewcommand*{\glxstrpostunset}[1]{%
3542   \@glxstr@entrycount@org@unset{##1}%
3543   \@gls@increment@currcount{##1}%
3544 }%
3545 \let\@glxstr@entrycount@org@localunset\glxstrpostlocalunset
3546 \renewcommand*{\glxstrpostlocalunset}[1]{%
3547   \@glxstr@entrycount@org@localunset{##1}%
3548   \@gls@local@increment@currcount{##1}%
3549 }%
3550 \let\@glxstr@entrycount@org@reset\glxstrpostreset
3551 \renewcommand*{\glxstrpostreset}[1]{%
3552   \@glxstr@entrycount@org@reset{##1}%
3553   \csgdef{glo\@glsdetoklabel{##1}@currcount}{0}%
3554 }%
3555 \let\@glxstr@entrycount@org@localreset\glxstrpostlocalreset
3556 \renewcommand*{\glxstrpostlocalreset}[1]{%
3557   \@glxstr@entrycount@org@localreset{##1}%
3558   \csdef{glo\@glsdetoklabel{##1}@currcount}{0}%
3559 }%

```

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

```

3560 \let\@cgls@\@cgls@
3561 \let\@cglspl@\@cglspl@

3562 \let\@cGls@\@cGls@
3563 \let\@cGlspl@\@cGlspl@
3564 \let\@cGLS@\@cGLS@
3565 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

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

```

`ite@entrycounts` Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```

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

```

```

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

```

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

```

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

```

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

`\@@cgl@`

```

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

```

```

3619  {%
3620    \@gls@{#1}{#2}[#3]%
3621  }%
3622 }%

```

\@@cglsp1@

```

3623 \def\@@cglsp1@#1#2[#3]{%
3624   \glstrifcounttrigger{#2}%
3625   {%
3626     \cglsp1format{#2}{#3}%
3627     \glunset{#2}%
3628   }%
3629   {%
3630     \@glsp1@{#1}{#2}[#3]%
3631   }%
3632 }%

```

\@@cGls@

```

3633 \def\@@cGls@#1#2[#3]{%
3634   \glstrifcounttrigger{#2}%
3635   {%
3636     \cGlsformat{#2}{#3}%
3637     \glunset{#2}%
3638   }%
3639   {%
3640     \@Gls@{#1}{#2}[#3]%
3641   }%
3642 }%

```

\@@cGlsp1@

```

3643 \def\@@cGlsp1@#1#2[#3]{%
3644   \glstrifcounttrigger{#2}%
3645   {%
3646     \cGlsp1format{#2}{#3}%
3647     \glunset{#2}%
3648   }%
3649   {%
3650     \@Glsp1@{#1}{#2}[#3]%
3651   }%
3652 }%

```

\@@cGLS@

```

3653 \def\@@cGLS@#1#2[#3]{%
3654   \glstrifcounttrigger{#2}%
3655   {%
3656     \cGLSformat{#2}{#3}%
3657     \glunset{#2}%
3658   }%
3659   {%

```

```

3660 \cGLS@{#1}{#2}[#3]%
3661 }%
3662 }%

```

\cGLSpl@

```

3663 \def\cGLSpl@#1#2[#3]{%
3664 \glxtrifcounttrigger{#2}%
3665 {%
3666 \cGLSplformat{#2}{#3}%
3667 \glset{#2}%
3668 }%
3669 {%
3670 \cGLSpl@{#1}{#2}[#3]%
3671 }%
3672 }%

```

Remove default warnings from \cglset etc so that it can be used interchangeable with \glset etc.

\cglset@

```

3673 \def\cglset@#1#2[#3]{\cglset@{#1}{#2}[#3]}

```

\cGLset@

```

3674 \def\cGLset@#1#2[#3]{\cGLset@{#1}{#2}[#3]}

```

\cglspl@

```

3675 \def\cglspl@#1#2[#3]{\cglspl@{#1}{#2}[#3]}

```

\cGLspl@

```

3676 \def\cGLspl@#1#2[#3]{\cGLspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

\cGLS

```

3677 \newrobustcmd*{\cGLS}{\cglsethyp@opt\cGLS}

```

\cGLS Defined the un-starred form. Need to determine if there is a final optional argument

```

3678 \newcommand*{\cGLS}[2][{}]{%
3679 \new@ifnextchar[\cGLS@{#1}{#2}]{\cGLS@{#1}{#2}[{}]}%
3680 }

```

\cGLS@

```

3681 \def\cGLS@#1#2[#3]{\cGLS@{#1}{#2}[#3]}

```

\cGLSformat Format used by \cGLS if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```

3682 \newcommand*{\cGLSformat}[2]{%
3683 \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsetformat{#1}{#2}}%
3684 }

```



```
\cGLSp1
3685 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\cGLSp1}
```

\cGLSp1 Defined the un-starred form. Need to determine if there is a final optional argument

```
3686 \newcommand*{\cGLSp1}[2] [] {%
3687   \new@ifnextchar[{\cGLSp1@{#1}{#2}}{\cGLSp1@{#1}{#2} []}%
3688 }
```

\cGLSp1@

```
3689 \def\cGLSp1@#1#2[#3]{\cGLSp1@{#1}{#2}[#3]}
```

\cGLSplformat Format used by \cGLSp1 if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
3690 \newcommand*{\cGLSplformat}[2] {%
3691   \expandafter\mfirstuc\MakeUppercase\expandafter{\cGLsplformat{#1}{#2}}%
3692 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3693 \renewcommand*{\cglformat}[2] {%
3694   \glsifregular{#1}
3695   {\glsentryfirst{#1}}%
3696   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3697 }
```

\cGlsformat

```
3698 \renewcommand*{\cGlsformat}[2] {%
3699   \glsifregular{#1}
3700   {\Glsentryfirst{#1}}%
3701   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3702 }
```

\cglsplformat

```
3703 \renewcommand*{\cglsplformat}[2] {%
3704   \glsifregular{#1}
3705   {\glsentryfirstplural{#1}}%
3706   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3707 }
```

\cGlsplformat

```
3708 \renewcommand*{\cGlsplformat}[2] {%
3709   \glsifregular{#1}
3710   {\Glsentryfirstplural{#1}}%
3711   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%#2%
3712 }
```

New code similar to above for unit counting.

defunitcounters

```
3713 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3714   \edef\@glo@countunit{\csuse{@glxstr@categoryattr@@\@glo@category @unitcount}}%
3715   \ifdefvoid\@glo@countunit
3716   {}%
3717   {%
3718     \@glxstr@ifunitcounter{\@glo@countunit}%
3719     {}%
3720     {\expandafter\@glxstr@addunitcounter\expandafter{\@glo@countunit}}%
3721   }%
3722 }
```

r@unitcountlist List to keep track of which counters are being used by the entry unit count facility.

```
3723 \newcommand*{\@glxstr@unitcountlist}{}
```

@addunitcounter

```
3724 \newcommand*{\@glxstr@addunitcounter}[1]{%
3725   \listadd{\@glxstr@unitcountlist}{#1}%
3726   \ifcsundef{glxstr@theunit@#1}
3727   {%
3728     \ifcsdef{theH#1}%
3729     {\csdef{glxstr@theunit@#1}{\csuse{theH#1}}}%
3730     {\csdef{glxstr@theunit@#1}{\csuse{the#1}}}%
3731   }%
3732   {}%
3733 }
```

r@ifunitcounter

```
3734 \newcommand*{\@glxstr@ifunitcounter}[3]{%
3735   \xifinlist{#1}{\@glxstr@unitcountlist}{#2}{#3}%
3736 }
```

urrentunitcount

```
3737 \newcommand*\@glxstr@currentunitcount[1]{%
3738   glo@\glsdetoklabel{#1}@currunit@glsggetattribute{#1}{unitcount}.%
3739   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3740 }
```

eviousunitcount

```
3741 \newcommand*\@glxstr@previousunitcount[1]{%
3742   glo@\glsdetoklabel{#1}@prevunit@glsggetattribute{#1}{unitcount}.%
3743   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3744 }
```

t@currunitcount

```
3745 \newcommand*{\@gls@increment@currunitcount}[1]{%
3746   \glshasattribute{#1}{unitcount}%
3747   {%
```

```

3748 \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3749 \ifcsundef{\@glxstr@csname}%
3750 {%
3751 \csgdef{\@glxstr@csname}{1}%
3752 \listcsxadd
3753 {glo@\glsdetoklabel{#1}@unitlist}%
3754 {\glsggetattribute{#1}{unitcount}.%
3755 \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3756 }%
3757 }%
3758 {%
3759 \csxdef{\@glxstr@csname}%
3760 {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3761 }%
3762 }%
3763 {}%
3764 }

```

t@currunitcount

```

3765 \newcommand*{\@glx@local@increment@currunitcount}[1]{%
3766 \glshasattribute{#1}{unitcount}%
3767 {%
3768 \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3769 \ifcsundef{\@glxstr@csname}%
3770 {%
3771 \csdef{\@glxstr@csname}{1}%
3772 \listcseadd
3773 {glo@\glsdetoklabel{#1}@unitlist}%
3774 {\glsggetattribute{#1}{unitcount}.%
3775 \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3776 }%
3777 }%
3778 {%
3779 \csedef{\@glxstr@csname}%
3780 {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3781 }%
3782 }%
3783 {}%
3784 }

```

r@currunitcount

```

3785 \newcommand*{\@glxstr@currunitcount}[2]{%
3786 \ifcsundef
3787 {glo@\glsdetoklabel{#1}@currunit@#2}%
3788 {0}%
3789 {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
3790 }%

```

r@prevunitcount

```

3791 \newcommand*{\@glxstr@prevunitcount}[2]{%
3792   \ifcsundef
3793     {glo@\glsdetoklabel{##1}@prevunit@#2}%
3794     {0}%
3795     {\csuse{glo@\glsdetoklabel{##1}@prevunit@#2}}%
3796 }%

```

entryunitcount

```

3797 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3798   \appto@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3799   \renewcommand*{\gls@defdocnewglossaryentry}{%
3800     \renewcommand*{\newglossaryentry}[2]{%
3801       \PackageError{glossaries}{\string\newglossaryentry\space
3802         may only be used in the preamble when entry counting has
3803         been activated}{If you use \string\glsenableentryunitcount\space
3804         you must place all entry definitions in the preamble not in
3805         the document environment}%
3806     }%
3807 }%

```

New commands to access new fields:

```

3808 \newcommand*{\glsentrycurrcount}[1]{%
3809   \@glxstr@currunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3810   \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3811 }%
3812 \newcommand*{\glsentryprevcount}[1]{%
3813   \@glxstr@prevunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3814   \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3815 }%

```

Access total count:

```

3816 \newcommand*{\glsentryprevtotalcount}[1]{%
3817   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3818     {0}%
3819     {%
3820       \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3821     }%
3822 }%

```

Access max value:

```

3823 \newcommand*{\glsentryprevmaxcount}[1]{%
3824   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3825     {0}%
3826     {%
3827       \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3828     }%
3829 }%

```

Adjust post unset and reset:

```

3830 \let\@glxstr@entryunitcount@org@unset\glxstrpostunset
3831 \renewcommand*{\glxstrpostunset}[1]{%
3832   \@glxstr@entryunitcount@org@unset{##1}%
3833   \@glx@increment@currunitcount{##1}%
3834 }%
3835 \let\@glxstr@entryunitcount@org@localunset\glxstrpostlocalunset
3836 \renewcommand*{\glxstrpostlocalunset}[1]{%
3837   \@glxstr@entryunitcount@org@localunset{##1}%
3838   \@glx@local@increment@currunitcount{##1}%
3839 }%
3840 \let\@glxstr@entryunitcount@org@reset\glxstrpostreset
3841 \renewcommand*{\glxstrpostreset}[1]{%
3842   \glshasattribute{##1}{unitcount}%
3843   {%
3844     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3845     \ifcsundef{\@glxstr@csname}%
3846     {%
3847       {\csgdef{\@glxstr@csname}{0}}%
3848     }%
3849   }%
3850 }%
3851 \let\@glxstr@entryunitcount@org@localreset\glxstrpostlocalreset
3852 \renewcommand*{\glxstrpostlocalreset}[1]{%
3853   \@glxstr@entryunitcount@org@localreset{##1}%
3854   \glshasattribute{##1}{unitcount}%
3855   {%
3856     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3857     \ifcsundef{\@glxstr@csname}%
3858     {%
3859       {\csdef{\@glxstr@csname}{0}}%
3860     }%
3861   }%
3862 }%

```

Modifications to take into account the attributes that govern whether the entry should be unset.

```

3863 \let\@cgl@\@cgl@
3864 \let\@cgl@spl@\@cgl@spl@

3865 \let\@cGl@\@cGl@
3866 \let\@cGl@spl@\@cGl@spl@
3867 \let\@cGL@\@cGL@
3868 \let\@cGL@spl@\@cGL@spl@

```

Write information to the aux file.

```

3869 \AtEndDocument{\@glx@write@entryunitcounts}%
3870 \renewcommand*{\@glx@entry@unitcount}[3]{%
3871   \csgdef{glo@glx@detoklabel{##1}@prevunit@##3}{##2}%
3872   \ifcsundef{glo@glx@detoklabel{##1}@prevunittotal}%

```

```

3873 {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
3874 {%
3875   \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
3876     \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
3877   }%
3878   \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3879   {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
3880   {%
3881     \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3882     \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3883     \fi
3884   }%
3885 }%
3886 \let\glsenableentryunitcount\relax
3887 \renewcommand*{\glsenableentrycount}{%
3888   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3889     can't be used with \string\glsenableentryunitcount}%
3890   {Use one or other but not both commands}%
3891 }%
3892 }
3893 \onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3894 \newcommand*{\@gls@entry@unitcount}[3]{%

```

ryunitcounts@do

```

3895 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3896   \immediate\write\@auxout
3897   {\string\@gls@entry@unitcount
3898     {\@glsentry}%
3899     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3900     }%
3901     {#1}}%
3902 }

```

entryunitcounts

```

3903 \newcommand*{\@gls@write@entryunitcounts}{%
3904   \immediate\write\@auxout
3905   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3906   \count@=0\relax
3907   \forallglsentries{\@glsentry}{%
3908     \glsattribute{\@glsentry}{unitcount}%
3909     {%
3910       \ifglsused{\@glsentry}%
3911       {%
3912         \forlistcsloop
3913         {\@gls@write@entryunitcounts@do}%
3914         {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3915       }%

```

```

3916      {}%
3917      \advance\count@ by \@ne
3918    }%
3919    {}%
3920  }%
3921  \ifnum\count@=0
3922    \GlossariesExtraWarningNoLine{Entry counting has been enabled
3923      \MessageBreak with \string\glsenableentryunitcount\space but the
3924      \MessageBreak attribute ‘unitcount’ hasn’t
3925      \MessageBreak been assigned to any of the defined
3926      \MessageBreak entries}%
3927  \fi
3928 }

```

`tryUnitCounting` The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

```

3929 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3930 \glsenableentryunitcount

```

Redefine `\gls` etc:

```

3931 \renewcommand*{\gls}{\cgl}%
3932 \renewcommand*{\Gls}{\cGls}%
3933 \renewcommand*{\glspl}{\cglspl}%
3934 \renewcommand*{\Glspl}{\cGlspl}%
3935 \renewcommand*{\GLS}{\cGLS}%
3936 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3937 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3938 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
3939 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3940 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3941 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3942 {Use one or other but not both commands}}%
3943 }

```

`tcountunsetattr`

```

3944 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3945 \@for\@glsxtr@cat:=#1\do
3946 {%
3947 \ifdefempty{\@glsxtr@cat}{}%
3948 {%
3949 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3950 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3951 }%
3952 }%
3953 }

```

1.3.6 Acronym Modifications

It's more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package's custom acronym format. (For example, user may already have defined some acronym styles with `\newacronymstyle` which they would like to continue to use.) The original glossaries acronym code can be restored with `\RestoreAcronyms`, but adjust `\SetGenericNewAcronym` so that `\newacronym` adds the category.

`nericNewAcronym`

```

3954 \renewcommand*{\SetGenericNewAcronym}{%
3955   \let\@Gls@entryname\@Gls@acrentryname
3956   \renewcommand{\newacronym}[4][]{%
3957     \ifdefempty{\@glsacronymlists}%
3958     {%
3959       \def\@glo@type{\acronymtype}%
3960       \setkeys{glossentry}{##1}%
3961       \DeclareAcronymList{\@glo@type}%
3962     }%
3963   }%
3964   \glskeylisttok{##1}%
3965   \glslabeltok{##2}%
3966   \glsshorttok{##3}%
3967   \glslongtok{##4}%
3968   \newacronymhook
3969   \protected@edef\@do@newglossaryentry{%
3970     \noexpand\newglossaryentry{\the\glslabeltok}%
3971     {%
3972       type=\acronymtype,%
3973       name={\expandonce{\acronymentry{##2}}},%
3974       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3975       text={\the\glsshorttok},%
3976       short={\the\glsshorttok},%
3977       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3978       long={\the\glslongtok},%
3979       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3980       category=acronym,%
3981       \GenericAcronymFields,%
3982       \the\glskeylisttok
3983     }%
3984   }%
3985   \@do@newglossaryentry
3986 }%
3987 \renewcommand*{\acrfullfmt}[3]{%
3988   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}%
3989 \renewcommand*{\Acrfullfmt}[3]{%
3990   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}%
3991 \renewcommand*{\ACRfullfmt}[3]{%
3992   \glslink[##1]{##2}{%

```



```

3993     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3994 \renewcommand*{\acrfullplfmt}[3]{%
3995   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3996 \renewcommand*{\Acrfullplfmt}[3]{%
3997   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3998 \renewcommand*{\ACRfullplfmt}[3]{%
3999   \glslink[##1]{##2}{%
4000     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
4001 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
4002 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
4003 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
4004 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
4005 }

```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```

4006 \let\@glsxtr@org@setacronymstyle\setacronymstyle
4007 \let\@glsxtr@org@newacronymstyle\newacronymstyle

```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbreviationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```

4008 \newcommand*{\MakeAcronymsAbbreviations}{%
4009   \renewcommand*{\newacronym}[4][]{%
4010     \glsxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4011   }%
4012   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4013   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
4014   \renewcommand*{\setacronymstyle}[1]{%
4015     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
4016       unavailable.
4017       Use \string\setabbreviationstyle\space instead.
4018       The original acronym interface can be restored with
4019       \string\RestoreAcronyms}{}%
4020   }%
4021   \renewcommand*{\newacronymstyle}[1]{%
4022     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4023       available unless you restore the original acronym interface with
4024       \string\RestoreAcronyms}%
4025     \@glsxtr@org@newacronymstyle{##1}%
4026   }%
4027 }

```

Switch acronyms to abbreviations:

```

4028 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

4029 \newcommand*\RestoreAcronyms}{%
4030   \SetGenericNewAcronym
4031   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
4032   \renewcommand{\acronymfont}[1]{##1}%
4033   \let\setacronymstyle\@glxtr@org@setacronymstyle
4034   \let\newacronymstyle\@glxtr@org@newacronymstyle

```

Need to restore the original definition of `\@gls@link@checkfirsthyper` but `\glxtrifwasfirstuse` still needs setting for the benefit of the post-link hook.

```

4035   \renewcommand*\@gls@link@checkfirsthyper{%
4036     \ifglused{\glslabel}%
4037     {\let\glxtrifwasfirstuse\@secondoftwo}
4038     {\let\glxtrifwasfirstuse\@firstoftwo}%
4039     \@glxtr@org@checkfirsthyper
4040   }
4041   \glssetcategoryattribute{acronym}{regular}{false}%
4042   \setacronymstyle{long-short}%
4043 }

```

`\glsacspace` Allow the user to customise the maximum value.

```

4044 \renewcommand*\glsacspace}[1]{%
4045   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4046   \ifdim\dimen@<\glsacspacemax~\else\space\fi
4047 }

```

`\glsacspacemax` Value used in the above.

```

4048 \newcommand*\glsacspacemax{3em}

```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the “noidx” commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require `makeindex/xindy`.

`\reg@glosslist`

```

4049 \newcommand*\@glxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

4050 \let\@glxtr@org@makeglossaries\makeglossaries

```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with record.

`\makeglossaries`

```

4051 \renewcommand*\makeglossaries}[1] []{%
4052   \@glxtr@if@record@only

```

```

4053 {%
4054   \PackageError{glossaries-extra}{\string\makeglossaries\space
4055     not permitted\MessageBreak with record=\@glxtr@record@setting\space
4056     package option}%
4057   {You may only use \string\makeglossaries\space with
4058     record=off or record=alsoindex options}%
4059 }%
4060 {%
4061   \ifblank{#1}%
4062   {\@glxtr@org@makeglossaries}%
4063   {%
4064     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
4065     \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4066       not permitted\MessageBreak with record=alsoindex package option}%
4067     {You may only use the hybrid \string\makeglossaries[...]\space with
4068       record=off option}%
4069   }else
4070     \edef\@glxtr@reg@glosslist{#1}%
4071     \ifundef{\glswrite}{\newwrite\glswrite}{}%
4072     \protected@write\@auxout{}{\string\providecommand
4073       \string\@glxtr@reg@glosslist{#1}}
4074     \protected@write\@auxout{}{\string\providecommand
4075       \string\@glxtr@reg@glosslist{#1}}
4076     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4077     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4078     \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
4079     \write\@auxout{\string\providecommand\string\@glxtr@reg@glosslist{#1}}

```

Iterate through each supplied glossary type and activate it.

```

4080   \@for\@glo@type:=#1\do{%
4081     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4082   }%

```

New glossaries must be created before \makeglossaries:

```

4083   \renewcommand*\newglossary[4][]{%
4084     \PackageError{glossaries}{New glossaries
4085       must be created before \string\makeglossaries}{You need
4086       to move \string\makeglossaries\space after all your
4087       \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

4088   \let\@makeglossary\relax
4089   \let\makeglossary\relax
4090   \renewcommand\makeglossaries[1][]{%

```

Disable all commands that have no effect after \makeglossaries

```

4091   \@disable@onlypremakeg

```

Allow see key:

```

4092   \let\gls@checkseeallowed\relax

```

Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```

4093     \renewcommand*{\@do@seeglossary}[2]{%
4094         \glsdoifexists{##1}%
4095         {%
4096             \edef\@gls@label{\glsdetoklabel{##1}}%
4097             \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4098             \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4099             {\@glsxtr@org@doseeglossary{##1}{##2}}%
4100             {%
4101                 \@@glsxtrwrglossmark
4102                 \protected@write\@auxout{}{%
4103                     \string\@gls@reference
4104                     {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
4105                 }%
4106             }%
4107         }%
4108     }%

```

Adjust \@do@@wrglossary

```

4109     \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
4110     \def\@do@@wrglossary{%
4111         \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4112         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4113         {\@glsxtr@@do@@wrglossary}%
4114         {\@gls@noidxglossary}%
4115     }%

```

Suppress warning about no \makeglossaries

```

4116     \let\warn@nomakeglossaries\relax
4117     \def\warn@noprintglossary{%
4118         \GlossariesWarningNoLine{No \string\printglossary\space
4119             or \string\printglossaries\space
4120             found.^^J(Remove \string\makeglossaries\space if you don't want
4121             any glossaries.)^^JThis document will not have a glossary}%
4122     }%

```

Only warn for glossaries not listed.

```

4123     \renewcommand{\@gls@noref@warn}[1]{%
4124         \edef\@gls@type{##1}%
4125         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4126         {%
4127             \GlossariesExtraWarning{Can't use
4128                 \string\printnoidxglossary[type={\@gls@type}]
4129                 when '\@gls@type' is listed in the optional argument of
4130                 \string\makeglossaries}%
4131         }%
4132     {%
4133         \GlossariesWarning{Empty glossary for
4134             \string\printnoidxglossary[type={##1}].

```

```

4135         Rerun may be required (or you may have forgotten to use
4136         commands like \string\gls))}%
4137     }%
4138 }%

```

Adjust display number list to check for type:

```

4139     \renewcommand*\glsdisplaynumberlist}[1]{%
4140     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4141     {\@glsxtr@idx@displaynumberlist{##1}}}%
4142     {\@glsxtr@noidx@displaynumberlist{##1}}}%
4143 }%

```

Adjust entry list:

```

4144     \renewcommand*\glsentrynumberlist}[1]{%
4145     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4146     {\@glsxtr@idx@entrynumberlist{##1}}}%
4147     {\@glsxtr@noidx@entrynumberlist{##1}}}%
4148 }%

```

Adjust number list loop

```

4149     \renewcommand*\glsnumberlistloop}[2]{%
4150     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4151     {%
4152         \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4153         not available for glossary ‘##1’}{}%
4154     }%
4155     {\@glsxtr@noidx@numberlistloop{##1}{##2}}}%
4156 }%

```

Only sanitize sort for normal indexing glossaries.

```

4157     \renewcommand*\glsprestandardsort}[3]{%
4158     \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4159     {%
4160         \glsdosanitizesort
4161     }%
4162     {%
4163         \ifglssanitizesort
4164         \@gls@noidx@sanitizesort
4165         \else
4166         \@gls@noidx@nosanitizesort
4167         \fi
4168     }%
4169 }%

```

Unlike \makenoidxglossaries we can't automatically set sanitizesort=false. All entries must be defined in the preamble.

```

4170     \renewcommand*\new@glossaryentry[2]{%
4171     \PackageError{glossaries-extra}{Glossary entries must be defined
4172     in the preamble\MessageBreak when you use the optional argument
4173     of \string\makeglossaries}{Either move your definitions to the
4174     preamble or don't use the optional argument of

```

```

4175         \string\makeglossaries}%
4176     }%

```

Only activate sort key for glossaries that aren't listed in #1 (glossary label is stored in \@glo@type but this defaults to \glsdefaulttype so some expansion is required).

```

4177         \let\@glo@assign@sortkey\@glstr@mixed@assign@sortkey
4178         \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

4179         \expandafter\@glstr@gettype\expandafter,\@glstr@printglossopts,%
4180         type=\glsdefaulttype,\@end@glstr@gettype
4181         \def\@glo@sorttype{\@glo@default@sorttype}%
4182     }%

```

Check automake setting:

```

4183         \ifglautomake
4184         \renewcommand*{\@gls@doautomake}{%
4185             \@for\@gls@type:=\@glstr@reg@glosslist\do{%
4186                 \ifdefempty{\@gls@type}}{\@gls@automake{\@gls@type}}}%
4187         }%
4188     }%
4189 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

4190         \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4191     \fi
4192 }%
4193 }%
4194 }

```

The optional argument version of \makeglossaries needs an adjustment to \@printglossary to allow \@glo@assign@sortkey to pick up the glossary type.

rgprintglossary This no longer simply saves \@printglossary with \let but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes \provideignoredglossary to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

4195 \newcommand{\@glstr@orgprintglossary}[2]{%
4196     \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

4197     \def\glossarytitle{%
4198         \ifcsdef{\@glo@type \@glo@type @title}%
4199         {\csuse{\@glo@type \@glo@type @title}}%
4200         {\glossaryname}}%
4201     \def\glossarytoctitle{\glossarytitle}%
4202     \let\org@glossarytitle\glossarytitle
4203     \def\@glossarystyle{%
4204         \ifx\@glossary@default@style\relax
4205             \GlossariesWarning{No default glossary style provided \MessageBreak
4206                 for the glossary '\@glo@type'. \MessageBreak

```

```

4207      Using deprecated fallback. \MessageBreak
4208      To fix this set the style with \MessageBreak
4209      \string\setglossarystyle\space or use the \MessageBreak
4210      style key=value option}%
4211  \fi
4212 }%
4213 \def\gls@dotocitle{\glssettocitle{\@glo@type}}%
4214 \let\org@glossaryentrynumbers\glossaryentrynumbers
4215 \bgroup
4216   \@printgloss@setsort
4217   \setkeys{printgloss}{#1}%
4218   \ifx\glossarytitle\org@glossarytitle
4219   \else
4220     \cslet{\glo@type@\@glo@type @title}{\glossarytitle}%
4221   \fi
4222   \let\currentglossary\@glo@type
4223   \let\org@glossaryentrynumbers\glossaryentrynumbers
4224   \let\glsnonextpages\@glsnonextpages
4225   \let\glsnextpages\@glsnextpages

4226   \glsxtractivatenopost
4227   \gls@dotocitle
4228   \@glossarystyle
4229   \let\gls@org@glossaryentryfield\glossentry
4230   \let\gls@org@glossarysubentryfield\subglossentry
4231   \renewcommand{\glossentry}[1]{%
4232     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4233     \gls@org@glossaryentryfield{##1}%
4234   }%
4235   \renewcommand{\subglossentry}[2]{%
4236     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4237     \gls@org@glossarysubentryfield{##1}{##2}%
4238   }%
4239   \@gls@preglossaryhook
4240   #2%
4241 \egroup
4242 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
4243 \global\let\warn@noprintglossary\relax
4244 }

```

ractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

4245 \newcommand*{\glsxtractivatenopost}{%
4246   \let\nopostdesc\@nopostdesc
4247   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4248 }

```

lsxtrnopostpunc

```

4249 \newrobustcmd*{\glsxtrnopostpunc}{%

```

gsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

4250 \newcommand{\@gsxtr@nopostpunc}{%
4251   \let\@gsxtr@org@postdescription\glspostdescription
4252   \ifglsnopostdot
4253     \renewcommand{\glspostdescription}{%
4254       \glsnopostdottrue
4255       \let\glspostdescription\@gsxtr@org@postdescription
4256       \let\glxtrrestorepostpunc\@gsxtr@restore@postpunc
4257       \glxtrpostdescription
4258       \@gsxtr@nopostpunc@postdesc}%
4259   \else
4260     \renewcommand{\glspostdescription}{%
4261       \let\glspostdescription\@gsxtr@org@postdescription
4262       \let\glxtrrestorepostpunc\@gsxtr@restore@postpunc
4263       \glxtrpostdescription
4264       \@gsxtr@nopostpunc@postdesc}%
4265   \fi
4266   \glsnopostdotfalse
4267 }

```

stpunc@postdesc

```

4268 \newcommand*{\@gsxtr@nopostpunc@postdesc}{}

```

estore@postpunc

```

4269 \newcommand*{\@gsxtr@restore@postpunc}{%
4270   \def\@gsxtr@nopostpunc@postdesc{%
4271     \@gsxtr@org@postdescription
4272     \let\@gsxtr@nopostpunc@postdesc\@empty
4273     \let\glxtrrestorepostpunc\@empty
4274   }%
4275 }

```

restorepostpunc Does nothing outside of glossary.

```

4276 \newcommand*{\glxtrrestorepostpunc}{}

```

\@printglossary Redefine.

```

4277 \renewcommand{\@printglossary}[2]{%
4278   \def\@gsxtr@printglossopts{#1}%
4279   \@gsxtr@orgprintglossary{#1}{#2}%
4280 }

```

Add a key that switches off the entry targets:

```

4281 \define@choicekey{printgloss}{target}
4282 [\@gsxtr@printglossval\@gsxtr@printglossnr]%
4283 {true,false}[true]%
4284 {%
4285   \ifcase\@gsxtr@printglossnr

```



```

4286 \def\@glstarget{\@glsdohypertarget}%
4287 \else
4288 \let\@glstarget\@secondoftwo
4289 \fi
4290 }

```

hypernameprefix

```

4291 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:
4292 \define@key{printgloss}{targetnameprefix}{%
4293 \renewcommand{\@glxtrhypernameprefix}{#1}%
4294 }

4295 \define@key{printgloss}{prefix}{%
4296 \renewcommand{\@glo@linkprefix}{#1}%
4297 }

```

glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```

4298 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4299 \renewcommand{\glsdohypertarget}[2]{%
4300 \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4301 }

```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```

4302 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4303 \def\@glstarget{\glsdohypertarget}%
4304 \fi

```

@makeglossaries For the benefit of makeglossaries

```

4305 \newcommand*{\@glxtr@makeglossaries}[1]{}

```

@glxtr@gettype Get just the type.

```

4306 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4307 \def\@glo@type{#2}%
4308 }

```

@assign@sortkey Assign the sort key.

```

4309 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4310 \edef\@glo@type{\@glo@type}%
4311 \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4312 {%
4313 \@glo@no@assign@sortkey{#1}%
4314 }%
4315 {%
4316 \@glo@assign@sortkey{#1}%
4317 }%
4318 }%

```

Display number list for the regular version:

splaynumberlist

```
4319 \let\@glstr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4320 \newcommand*{\@glstr@noidx@displaynumberlist}[1]{%
4321   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4322   \ifdef\@gls@loclist
4323   {%
4324     \def\@gls@noidx@loclist@sep{%
4325       \def\@gls@noidx@loclist@sep{%
4326         \def\@gls@noidx@loclist@sep{%
4327           \glsnumlistsep
4328         }%
4329         \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4330       }%
4331     }%
4332     \def\@gls@noidx@loclist@finalsep{}%
4333     \def\@gls@noidx@loclist@prev{}%
4334     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4335     \@gls@noidx@loclist@finalsep
4336     \@gls@noidx@loclist@prev
4337   }%
4338   {%
4339     \glstrundeftag
4340     \glsdoifexists{#1}%
4341     {%
4342       \GlossariesWarning{Missing location list for ‘#1’. Either
4343         a rerun is required or you haven’t referenced the entry.}%
4344     }%
4345   }%
4346 }%
4347
```

And for the number list loop:

@numberlistloop

```
4348 \newcommand*{\@glstr@noidx@numberlistloop}[3]{%
4349   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4350   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4351   \let\@gls@org@glsseeformat\glsseeformat
4352   \let\glsnoidxdisplayloc#2\relax
4353   \let\glsseeformat#3\relax
4354   \ifdef\@gls@loclist
4355   {%
4356     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4357   }%
```

```

4358 {%
4359     \glxtrundeftag
4360     \glsdoifexists{#1}%
4361     {%
4362         \GlossariesWarning{Missing location list for ‘##1’. Either
4363             a rerun is required or you haven’t referenced the entry.}%
4364     }%
4365 }%
4366 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4367 \let\glsseeformat\@gls@org@glsseeformat
4368 }%

```

Same for entry number list.

entrynumberlist

```

4369 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4370     \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4371     \ifdef\@gls@loclist
4372     {%
4373         \glsnoidxloclist{\@gls@loclist}%
4374     }%
4375     {%
4376         \glxtrundeftag
4377         \glsdoifexists{#1}%
4378         {%
4379             \GlossariesWarning{Missing location list for ‘#1’. Either
4380                 a rerun is required or you haven’t referenced the entry.}%
4381         }%
4382     }%
4383 }%

```

entrynumberlist

```

4384 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\@glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4385 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4386     \protected@edef\@glsxtr@titlelabel{#1}%
4387     \ifdefvoid\@glsxtr@titlelabel
4388     {}%
4389     {%
4390         \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4391     }%
4392     \ifdefvoid{\@glsxtr@titlelabel}%
4393     {%
4394         \DTLifint{#1}%
4395         {%
4396             \ifnum#1<256\relax
4397                 \edef#2{\char#1\relax}%

```

```

4398     \else
4399     \edef#2{#1}%
4400     \fi
4401 }%
4402 {%
4403     \ifcsundef{#1groupname}%
4404     {\def#2{#1}}%
4405     {\letcs#2{#1groupname}}%
4406 }%
4407 }%
4408 {%
4409     \let#2\@glsxtr@titlelabel
4410 }%
4411 }

```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```

4412 \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.

```

4413 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4414     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4415     \@onelevel@sanitize\@glsxtr@titlelabel
4416     \ifcsdef{\@glsxtr@titlelabel}
4417     {\letcs{#2}{\@glsxtr@titlelabel}}%
4418     {\glsxtr@org@getgrouptitle{#1}{#2}}%
4419 }
4420 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

4421 \newcommand{\glsxtrsetgrouptitle}[2]{%
4422     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4423     \@onelevel@sanitize\@glsxtr@titlelabel
4424     \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4425 }

```

alsetgrouptitle As above put only locally defines the title.

```

4426 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4427     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4428     \@onelevel@sanitize\@glsxtr@titlelabel
4429     \protected@csedef{\@glsxtr@titlelabel}{#2}%
4430 }

```

\glsnavigation Redefine to use new user-level command.

```

4431 \renewcommand*{\glsnavigation}{%
4432     \def\@gls@between{}%
4433     \ifcsundef{\@gls@hypergroup@list@{\@gls@type}}%
4434     {%

```

```

4435 \def\@gls@list{%
4436 }%
4437 {%
4438 \expandafter\let\expandafter\@gls@list
4439 \csname @gls@hypergroup@list@\@glo@type\endcsname
4440 }%
4441 \@for\@gls@tmp:=\@gls@list\do{%
4442 \@gls@between
4443 \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4444 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4445 \let\@gls@between\glshypernavsep
4446 }%
4447 }

```

`@noidx@glossary`

```

4448 \renewcommand*{\@print@noidx@glossary}{%
4449 \ifcsdef{\@gls@ref@\@glo@type}%
4450 {%
4451 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4452 {%
4453 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4454 }%
4455 {%
4456 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4457 }%
4458 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4459 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4460 \def\@gls@currentlettergroup{%
4461 \begin{theglossary}%
4462 \glossaryheader
4463 \glsresetentrylist
4464 \forlistcsloop{\@gls@noidx@do}{\@gls@ref@\@glo@type}%
4465 \end{theglossary}%
4466 \glossarypostamble
4467 }%
4468 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4469 \glsxtrifemptyglossary{\@glo@type}%
4470 }%
4471 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4472 \@gls@noref@warn{\@glo@type}%
4473 }%
4474 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4475 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4476   \setentrycounter[#1]{#2}%
4477   \@glsxtr@display@loc#3\empty\end@glsxtr@display@loc{#4}%
4478 }

```

`xtr@display@loc` Patch to check for range formations.

```

4479 \def\@glsxtr@display@loc#1#2\end@glsxtr@display@loc#3{%
4480   \ifx#1(\relax
4481     \glsxtrdisplaystartloc{#2}{#3}%
4482   \else
4483     \ifx#1)\relax
4484       \glsxtrdisplayendloc{#2}{#3}%
4485     \else
4486       \glsxtrdisplaysingleloc{#1#2}{#3}%
4487     \fi
4488   \fi
4489 }

```

`isplayingleloc` Single location.

```

4490 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4491   \csuse{#1}{#2}%
4492 }

```

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.

```

4493 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4494   \edef\glsxtrlocrangefmt{#1}%
4495   \ifx\glsxtrlocrangefmt\empty
4496     \def\glsxtrlocrangefmt{\glsnumberformat}%
4497   \fi
4498   \expandafter\glsxtrdisplaysingleloc
4499   \expandafter{\glsxtrlocrangefmt}{#2}%
4500 }

```

`trdisplayendloc` End of a location range.

```

4501 \newcommand*{\glsxtrdisplayendloc}[2]{%
4502   \edef\@glsxtr@tmp{#1}%
4503   \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{\glsnumberformat}}{}}%
4504   \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4505   \else
4506     \GlossariesExtraWarning{Mismatched end location range
4507       (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
4508   \fi
4509   \expandafter\glsxtrdisplayendloohook\expandafter{\@glsxtr@tmp}{#2}%
4510   \expandafter\glsxtrdisplaysingleloc
4511   \expandafter{\glsxtrlocrangefmt}{#2}%
4512   \def\glsxtrlocrangefmt{}%
4513 }

```

`splayendlochook` Allow the user to hook into the end of range command.

```
4514 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```
4515 \newcommand*{\glsxtrlocrangefmt}{}%
```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```
4516 \renewcommand*{\setentrycounter}[2][{}]{%
4517   \def\glsxtrcounterprefix{#1}%
4518   \ifx\glsxtrcounterprefix\@empty
4519     \def\@glo@counterprefix{.}%
4520   \else
4521     \def\@glo@counterprefix{.#1.}%
4522   \fi
4523   \def\glsetentrycounter{#2}%
4524 }
```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```
4525 \def\@gls@removespaces#1 #2\@nil{%
4526   \toks@=\expandafter{\the\toks@#1}%
4527   \ifx\@#2\@nil
4528     \edef\x{\the\toks@}%
4529     \ifx\x\empty
4530     \else
```

Expand location (just in case `\toks@` is needed for something else).

```
4531   \expandafter\glxtrlocationhyperlink\expandafter
4532   \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4533   \fi
4534 \else
4535   \@gls@ReturnAfterFi{%
4536     \@gls@removespaces#2\@nil
4537   }%
4538 \fi
4539 }
```

`locationhyperlink` `\glxtrlocationhyperlink{<counter>}{<prefix>}{<location>}`

```
4540 \newcommand*{\glxtrlocationhyperlink}[3]{%
4541   \ifdefined\glxtrsupplocationurl
4542   {%
4543     \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4544   }%
4545   {%
4546     \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
4547   }%
4548 }
```

supphypernumber

```
4549 \newcommand*{\glxtrsupphypernumber}[1]{%
4550 {%
4551   \glshasattribute{\glscurrententrylabel}{externallocation}%
4552   {%
4553     \def\glxtrsupplocationurl{%
4554       \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4555   }%
4556   {%
4557     \def\glxtrsupplocationurl{}%
4558   }%
4559   \glshypernumber{#1}%
4560 }%
4561 }
```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```
4562 \renewcommand{\@print@glossary}{%
4563   \makeatletter
4564   \@input@{\jobname.\csname @glo@type @in\endcsname}%
4565   \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
4566   {}%
4567   {\glxtrNoGlossaryWarning{\@glo@type}}%
4568   \ifglxindy
4569     \ifcsundef{@xdy@\@glo@type @language}%
4570     {%
4571       \edef\@do@auxoutstuff{%
4572         \noexpand\AtEndDocument{%
4573           \noexpand\immediate\noexpand\write\@auxout{%
4574             \string\providecommand\string\@xdylanguage[2]{}}%
4575           \noexpand\immediate\noexpand\write\@auxout{%
4576             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4577         }%
4578       }%
4579     }%
4580     {%
4581       \edef\@do@auxoutstuff{%
4582         \noexpand\AtEndDocument{%
4583           \noexpand\immediate\noexpand\write\@auxout{%
4584             \string\providecommand\string\@xdylanguage[2]{}}%
4585           \noexpand\immediate\noexpand\write\@auxout{%
4586             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4587               @language\endcsname}}%
4588         }%
4589       }%
4590     }%
4591     \@do@auxoutstuff
```



```

4592 \edef\@do@auxoutstuff{%
4593 \noexpand\AtEndDocument{%
4594 \noexpand\immediate\noexpand\write\@auxout{%
4595 \string\providecommand\string\@gls@codepage[2]{}}%
4596 \noexpand\immediate\noexpand\write\@auxout{%
4597 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4598 }%
4599 }%
4600 \@do@auxoutstuff
4601 \fi
4602 \renewcommand*{\@warn@nomakeglossaries}{%
4603 \GlossariesWarningNoLine{\string\makeglossaries\space
4604 hasn't been used,^^Jthe glossaries will not be updated}%
4605 }%
4606 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4607 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4608 This document is incomplete. The external file associated with
4609 the glossary '#1' (which should be called \texttt{#2})
4610 hasn't been created.%
4611 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4612 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4613 This has probably happened because there are no entries defined
4614 in this glossary.%
4615 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4616 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4617 If you don't want this glossary,
4618 add \texttt{nomain} to your package option list when you load
4619 \texttt{glossaries-extra.sty}. For example:%
4620 }

```

`\GlsWarningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4621 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4622 Did you forget to use \texttt{type=#1} when you defined your
4623 entries? If you tried to load entries into this glossary with
4624 \texttt{\string\loadglsentries} did you remember to use
4625 \texttt{[#1]} as the optional argument? If you did, check that
4626 the definitions in the file you loaded all had the type set
4627 to \texttt{\string\glsdefaulttype}.%
4628 }

```

`\GlsWarningCheckFile` Advisory message to check the file contents.

```

4629 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4630   Check the contents of the file \texttt{#1}. If
4631   it's empty, that means you haven't indexed any of your entries in this
4632   glossary (using commands like \texttt{\string\gls} or
4633   \texttt{\string\glsadd}) so this list can't be generated.
4634   If the file isn't empty, the document build process hasn't been
4635   completed.%
4636 }

```

WarningAutoMake Message when automake option has been used.

```

4637 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4638   You may need to rerun \LaTeX. If you already have, it may be that
4639   \TeX's shell escape doesn't allow you to run
4640   \ifglxindy xindy\else makeindex\fi. Check the
4641   transcript file \texttt{\jobname.log}. If the shell escape is
4642   disabled, try one of the following:
4643
4644   \begin{itemize}
4645     \item Run the external (Lua) application:
4646
4647       \texttt{makeglossaries-lite.lua \string\jobname\string}
4648
4649     \item Run the external (Perl) application:
4650
4651       \texttt{makeglossaries \string\jobname\string}
4652   \end{itemize}
4653
4654   Then rerun \LaTeX\ on this document.
4655   \GlossariesExtraWarning{Rerun required to build the
4656   glossary '#1' or check TeX's shell escape allows
4657   you to run \ifglxindy xindy\else makeindex\fi}%
4658 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4659 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4660   You need to either replace \texttt{\string\makenoidxglossaries}
4661   with \texttt{\string\makeglossaries} or replace
4662   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4663   \texttt{\string\printnoidxglossary}
4664   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4665   this document.%
4666 }

```

WarningBuildInfo Build advice.

```

4667 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4668   Try one of the following:
4669   \begin{itemize}
4670     \item Add \texttt{automake} to your package option list when you load

```

```

4671         \texttt{glossaries-extra.sty}. For example:
4672
4673         \texttt{\string\usepackage[automake]%
4674             \glsopenbrace glossaries-extra\glsclosebrace}
4675
4676     \item Run the external (Lua) application:
4677
4678         \texttt{makeglossaries-lite.lua \string\jobname\string}
4679
4680     \item Run the external (Perl) application:
4681
4682         \texttt{makeglossaries \string\jobname\string}
4683 \end{itemize}
4684
4685 Then rerun \LaTeX\ on this document.%
4686 }

```

trRecordWarning Paragraph for record=only.

```

4687 \newcommand{\GlsXtrRecordWarning}[1]{%
4688 \texttt{\string\printglossary} doesn't work
4689 with the \texttt{record=only} package option
4690 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4691 instead (or change the package option).%
4692 }

```

oGlsWarningTail Final paragraph.

```

4693 \newcommand{\GlsXtrNoGlsWarningTail}{%
4694 This message will be removed once the problem has been fixed.%
4695 }

```

GlsWarningNoOut No out file created. Build advice.

```

4696 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4697 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4698 \texttt{\string\makeglossaries} or you have used
4699 \texttt{\string\nofiles}. If this is just a draft version of the
4700 document, you can suppress this message using the
4701 \texttt{nomissinggls\text} package option.%
4702 }

```

glossarywarning

```

4703 \newcommand*{@glsxtr@defaultnoglossarywarning}[1]{%
4704 \glossarysection[\glossarytoctitle]{\glossarytitle}
4705 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glo@type @in\endcsname}
4706 \par
4707 \glsxtrifemptyglossary{#1}%
4708 {%
4709 \GlsXtrNoGlsWarningEmptyStart\space
4710 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4711 \medskip

```

```

4712 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4713 \glsopenbrace glossaries-extra\glsclosebrace}
4714 \medskip
4715 }%
4716 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4717 }%
4718 {%
4719 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4720 {%
4721 \GlsXtrNoGlsWarningCheckFile
4722 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4723
4724 \ifglsautomake
4725
4726 \GlsXtrNoGlsWarningAutoMake{#1}
4727
4728 \else
4729
4730 \ifthenelse{\equal{#1}{main}}%
4731 {%
4732 \GlsXtrNoGlsWarningEmptyMain\par
4733 \medskip
4734 \noindent\texttt{\string\usepackage[nomain]%
4735 \glsopenbrace glossaries-extra\glsclosebrace}
4736 \medskip
4737 }%
4738 {}%
4739
4740 \ifdefequal\makeglossaries\@no@makeglossaries
4741 {%
4742 \GlsXtrNoGlsWarningMisMatch
4743 }%
4744 {%
4745 \GlsXtrNoGlsWarningBuildInfo
4746 }%
4747 \fi
4748 }%
4749 {%
4750 \GlsXtrNoGlsWarningNoOut
4751 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4752 }%
4753 }%
4754 \par
4755 \GlsXtrNoGlsWarningTail
4756 }

```

glossarywarning Warn about using `\printglossary` with record

```

4757 \newcommand*{\@glsxtr@record@noglossarywarning}[1]{%
4758 \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak

```

```

4759 with record=only package option\MessageBreak(use
4760 \string\printunsrtglossary[type=#1])\MessageBreak
4761 instead (or change the package option)}%
4762 \glossarysection[\glossarytoctitle]{\glossarytitle}
4763 \GlsXtrRecordWarning{#1}
4764 \GlsXtrNoGlsWarningTail
4765 }

```

Provide some commands to accompany the record option for use with **bib2gls**.

xtrresourcefile Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```

4766 \newcommand*{\glsxtrresourcefile}[2] [] {%

```

The record option can't be set after this command.

```

4767 \disable@keys{glossaries-extra.sty}{record}%
4768 \glsxtr@writefields
4769 \protected@write\@auxout{\glsxtrresourceinit}{\string\glsxtr@resource{#1}{#2}}%
4770 \let\@glsxtr@org@see@noindex\@gl@see@noindex
4771 \let\@gl@see@noindex\relax
4772 \IfFileExists{#2.glstex}%
4773 {%

```

Can't scope \input so save and restore the category code of @ to allow for internal commands in the location list.

```

4774 \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4775 \makeatletter
4776 \@input{#2.glstex}%
4777 \@bibgls@restoreat

```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```

4778 \glsxtr@check@bibgls@nameref
4779 }%
4780 {%
4781 \GlossariesExtraWarning{No file '#2.glstex'}%
4782 }%
4783 \let\@gl@see@noindex\@glsxtr@org@see@noindex
4784 }
4785 \onlypreamble\glsxtrresourcefile

```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```

4786 \newcommand{\glsxtr@check@bibgls@nameref}{%
4787 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4788 \ifdef\bibglshrefchar
4789 {}%
4790 {%
4791 \GlossariesExtraWarning{record=nameref requires at least
4792 version 1.8 of bib2gls}%
4793 }%

```

```

4794 \fi
4795 \let\@glxstr@check@bibgls@nameref\relax
4796 }

xtrresourceinit Code used during the protected write operation.
4797 \newcommand*{\glxtrresourceinit}{%

trresourcecount
4798 \newcount\glxtrresourcecount

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.
4799 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
4800 \ifnum\glxtrresourcecount=0\relax
4801 \glxtrresourcefile[#1]{\jobname}%
4802 \else
4803 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4804 \fi
4805 \advance\glxtrresourcecount by 1\relax
4806 }

glxstr@resource
4807 \newcommand*{\glxstr@resource}[2]{%

\glxstr@fields
4808 \newcommand*{\glxstr@fields}[1]{%

xtr@texencoding
4809 \newcommand*{\glxstr@texencoding}[1]{%

\glxstr@langtag
4810 \newcommand*{\glxstr@langtag}[1]{%

@pluralsuffixes
4811 \newcommand*{\glxstr@pluralsuffixes}[4]{%

tr@shortcutsval
4812 \newcommand*{\glxstr@shortcutsval}[1]{%

sxtr@linkprefix
4813 \newcommand*{\glxstr@linkprefix}[1]{%

xtr@writefields This information only needs to be written once, so disable it after it's been used.
4814 \newcommand*{\glxstr@writefields}{%

```

```

4815 \protected@write\@auxout{}%
4816   {\string\providecommand*{\string\glsxtr@fields}[1]{}}%
4817 \protected@write\@auxout{}%
4818   {\string\providecommand*{\string\glsxtr@resource}[2]{}}%
4819 \protected@write\@auxout{}%
4820   {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}}%
4821 \protected@write\@auxout{}%
4822   {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}}%
4823 \protected@write\@auxout{}%
4824   {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}}%
4825 \protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%

4826 \protected@write\@auxout{}%
4827   {\string\providecommand*{\string\glsxtr@record}[5]{}}%

4828 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
4829   \protected@write\@auxout{}%
4830     {\string\providecommand*{\string\glsxtr@record@nameref}[8]{}}%
4831 \fi

```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by `tracklang`). For multilingual documents, the required locale will have to be indicated in the sort key when using `\glsxtrresourcefile`.

```

4832 \ifdef\CurrentTrackedLanguageTag
4833   {%
4834     \protected@write\@auxout{}{%
4835       \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
4836   }%
4837   {%
4838     \protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
4839       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4840       {\glsxtrabbrvpluralsuffix}}%
4841   \ifdef\inputencodingname
4842     {%
4843       \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
4844     }%
4845   {%

```

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.)

```

4846     \@ifpackageloaded{fontspec}%
4847     {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
4848     {}%
4849   }%
4850 \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.

```

4851 \AtBeginDocument
4852   {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
4853 \let\glsxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. The double-quotes around \jobname have been removed (v1.19) since \jobname will include double-quotes if the file name has spaces.

```
4854 \ifglautomake
4855 \IfFileExists{\jobname.aux}%
4856 {\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.

```
4857 \ifx\@gls@doautomake\@gls@doautomake@err
4858 \let\@gls@doautomake\relax
4859 \fi
4860 \fi
4861 }
```

do@automake@err

```
4862 \newcommand*{\@gls@doautomake@err}{%
4863 \PackageError{glossaries}{You must use
4864 \string\makeglossaries\space with automake=true}
4865 {%
4866 Either remove the automake=true setting or
4867 add \string\makeglossaries\space to your document preamble.%
4868 }%
4869 }
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```
4870 \newcommand*{\glsxtr@record}[5] {}
```

@record@nameref Used with record=nameref to include current label information.

```
4871 \newcommand*{\glsxtr@record@nameref}[8] {}
```

r@counterrecord Aux file command.

```
4872 \newcommand*{\glsxtr@counterrecord}[3] {%
4873 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4874 }
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
4875 \newcommand*{\@glsxtr@counterrecordhook} {}
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4876 \newcommand*{\GlsXtrRecordCounter}[1] {%
4877 \@glsxtr@recordcounter{#1}%
4878 }
4879 \@onlypreamble\GlsXtrRecordCounter
```


docounterrecord

```
4880 \newcommand*{\@glstr@docounterrecord}[1]{%
4881   \protected@write\@auxout{}\string\glstr@counterrecord
4882     {\@glsl@label}{#1}{\csuse{the#1}}}%
4883 }
```

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.

```
4884 \newcommand*{\glstrglossentry}[1]{%
4885   \glstrtitleorpdforheading
4886   {\@glstrglossentry{#1}}%
4887   {\glsentryname{#1}}%
4888   {\glstrheadname{#1}}%
4889 }
```

lsxtrglossentry Another test is needed in case `\@glstrglossentry` has been written to the table of contents.

```
4890 \newrobustcmd*{\@glstrglossentry}[1]{%
4891   \glstrtitleorpdforheading
4892   {%
4893     \glsoifexists{#1}%
4894     {%
4895       \begingroup
4896       \edef\glscurrententrylabel{\glsetoklabel{#1}}%
4897       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4898       \ifglshasparent{#1}%
4899       {\GlsXtrStandaloneSubEntryItem{#1}}%
4900       {\glsentryitem{#1}}%
4901       \GlsXtrStandaloneEntryName{#1}%
4902     \endgroup
4903   }%
4904   }%
4905   {\glsentryname{#1}}%
4906   {\glstrheadname{#1}}%
4907 }
```

daloneEntryName

```
4908 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
4909   \glstarget{#1}{\glossentryname{#1}}%
4910 }
```

oneGlossaryType To make it easier to adjust the definition of `\currentglossary` within `\glstrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4911 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}

oneSubEntryItem  Used for sub-entries in standalone format. The argument is the entry's label.
4912 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4913   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
4914 }

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.
4915 \newcommand*{\glsxtrglossentryother}[3]{%
4916   \ifstrempy{#1}%
4917   {%
4918     \ifcsdef{glsxtrhead#3}%
4919     {%
4920       \glsxtrtitleorpdforheading
4921       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4922       {\@gls@entry@field{#2}{#3}}%
4923       {\csuse{glsxtrhead#3}{#2}}%
4924     }%
4925     {%
4926       \glsxtrtitleorpdforheading
4927       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4928       {\@gls@entry@field{#2}{#3}}%
4929       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4930     }%
4931   }%
4932   {%
4933     \glsxtrtitleorpdforheading
4934     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4935     {\@gls@entry@field{#2}{#3}}%
4936     {#1}%
4937   }%
4938 }

glossentryother  As \@glsxtrglossentry but uses a different field.
4939 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
4940   \glsxtrtitleorpdforheading
4941   {%
4942     \glsdoifexists{#1}%
4943     {%
4944       \begingroup
4945       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4946       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4947       \ifglshasparent{#1}%
4948       {\GlsXtrStandaloneSubEntryItem{#1}}%
4949       {\glsentryitem{#1}}%
4950       \GlsXtrStandaloneEntryOther{#1}%

```

```

4951      \endgroup
4952    }%
4953  }%
4954  {\@gls@entry@field{#1}{#2}}%
4955  {#3}%
4956 }

```

aloneEntryOther

```

4957 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
4958   \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4959 }

```

ntunsrtglossary Similar to \printnoidxglossary but it displays all entries defined for the given glossary without sorting.

```

4960 \newcommand*{\printunsrtglossary}{%
4961   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4962 }

```

ntunsrtglossary Unstarred version.

```

4963 \newcommand*{\@printunsrtglossary}[1][ ]{%
4964   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4965 }

```

ntunsrtglossary Starred version.

```

4966 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4967   \begingroup
4968     #2%
4969     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4970   \endgroup
4971 }

```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```

4972 \newcommand*{\printunsrtglossaries}{%
4973   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4974 }

```

@unsrt@glossary

```

4975 \newcommand*{\@print@unsrt@glossary}{%
4976   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4977   \glossarypreamble
4978   check for empty list
4979   \glstrifemptyglossary{\@glo@type}%
4980   {\%
4981     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4982   }%

```

```

4983 \key@ifundefined{glossentry}{group}%
4984 {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4985 {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4986 \def\@gls@currentlettergroup{}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

4987 \def\@glsxtr@doglossary{%
4988   \begin{theglossary}%
4989   \glossaryheader
4990   \glsresetentrylist
4991 }%
4992 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4993   :\expandafter=\csname glo@list@\@glo@type\endcsname\do{%
4994   \ifdefempty{\glscurrententrylabel}
4995   }%
4996   {%

```

Provide a hook (for example to measure width).

```

4997   \let\glsxtr@process\@firstofone
4998   \let\printunsrtglossaryskipentry
4999   \@glsxtr@printunsrtglossaryskipentry
5000   \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5001   \glsxtr@process
5002   {%
5003     \ifglshasparent{\glscurrententrylabel}{}%
5004     {%
5005       \@glsxtr@checkgroup\glscurrententrylabel
5006       \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
5007         {\@glsxtr@groupheading}%
5008     }%
5009     \eappto\@glsxtr@doglossary{%
5010       \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5011   }%
5012 }%
5013 }%
5014 \appto\@glsxtr@doglossary{\end{theglossary}}%
5015 \printunsrtglossarypredoglossary
5016 \@glsxtr@doglossary
5017 }%
5018 \glossarypostamble
5019 }

```

entryprocesshook

```

5020 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{%

```

ossaryskipentry

```

5021 \newcommand*{\printunsrtglossaryskipentry}{%
5022   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space

```

```

5023 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5024 }

```

entryprocesshook

```

5025 \newcommand*{\@glstr@printunsrtglossaryskipentry}{%
5026   \let\glstr@process\@gobble
5027 }

```

rypredoglossary

```

5028 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```

5029 \newcommand{\@printunsrt@glossary@handler}[1]{%
5030   \xdef\glscurrententrylabel{#1}%
5031   \printunsrtglossaryhandler\glscurrententrylabel
5032 }

```

glossaryhandler

```

5033 \newcommand{\printunsrtglossaryhandler}[1]{%
5034   \glstrunsrtdo{#1}%
5035 }

```

striflabelinlist

```
\glstriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```

5036 \newrobustcmd*{\glstriflabelinlist}[4]{%
5037   \protected@edef\@glstr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
5038   \@glstr@doiflabelinlist{#3}{#4}%
5039 }

```

srtglossaryunit

```

5040 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5041   \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
5042     \printunsrtglossaryunitsetup{#2}%
5043   }%
5044 }

```

ossaryunitsetup

```

5045 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
5046   \renewcommand{\printunsrtglossaryhandler}[1]{%
5047     \glstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5048     {\glstrunsrtdo{##1}}%
5049   }%
5050 }%

```

Only the target names should have the prefixes adjusted as \gls etc need the original \glslinkprefix. The \@gobble part discards \glslinkprefix.

```

5051 \ifcsundef{theH#1}%
5052 {%
5053 \renewcommand*{\@glstrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
5054 }%
5055 {%
5056 \renewcommand*{\@glstrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
5057 }%
5058 \renewcommand*{\glossarysection}[2][{}]{%
5059 \appto\glossarypostamble{\glspar\medskip\glspar}%
5060 }

```

unsrtglossaryunit

```

5061 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
5062 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5063 requires the record=only or record=alsoindex package option}{}%
5064 }

```

t@getgrouptitle

```

5065 \newrobustcmd*{\@glstr@unsrt@getgrouptitle}[2]{%
5066 \protected@edef\@glstr@titlelabel{\glstr@grouptitle@#1}%
5067 \@onelevel@sanitize\@glstr@titlelabel
5068 \ifcsdef{\@glstr@titlelabel}
5069 {\letcs{#2}{\@glstr@titlelabel}}%
5070 {\def#2{#1}}%
5071 }

```

\glstrunsrtdo Provide a user-level call to \@glstr@noidx@do to make it easier to define a new handler.

```

5072 \newcommand{\glstrunsrtdo}{\@glstr@noidx@do}

```

lsxtrgroupfield bib2gls provides a supplementary field labelled secondarygroup for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```

5073 \newcommand*{\glstrgroupfield}{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 \@glstr@doglossary is being constructed rather than in the handler.

glstr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glstr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glstr@groupheading, which will be empty if no heading is required.

```

5074 \newcommand*{\@glstr@checkgroup}[1]{%
5075 \def\@glstr@groupheading{}%
5076 \key@ifundefined{glossentry}{group}%
5077 {%

```

```

5078 \letcs{\@gls@sort}{glo@glstetoklabel{#1}@sort}%
5079 \expandafter\glo@grabfirst\@gls@sort{}\@nil
5080 }%
5081 {%
5082 \protected@edef\@glo@thislettergrp{%
5083 \csuse{glo@glstetoklabel{#1}@glstxtrgroupfield}}%
5084 }%
5085 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5086 {}%
5087 {%
5088 \ifdefempty{\@gls@currentlettergroup}{}%
5089 {\def\@glstxtr@groupheading{\glsgroupskip}}%
5090 \eappto\@glstxtr@groupheading{%
5091 \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
5092 }%
5093 }%
5094 \let\@gls@currentlettergroup\@glo@thislettergrp
5095 }

```

trLocationField Stores the internal name of the location field.

```

5096 \newcommand*\@GlsXtrLocationField{location}

```

glstxtr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present, but also need to check for the group field.

```

5097 \newcommand{\@glstxtr@noidx@do}[1]{%
5098 \ifglstentryexists{#1}%
5099 {%
5100 \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
5101 \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@GlsXtrLocationField}%
5102 \ifglshasparent{#1}%
5103 {%
5104 \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
5105 \ifdefvoid{\@gls@location}%
5106 {%
5107 \ifdefvoid{\@gls@loclist}%
5108 {%
5109 \subglossentry{\gls@level}{#1}{}%
5110 }%
5111 {%
5112 \subglossentry{\gls@level}{#1}%
5113 {%
5114 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5115 }%
5116 }%
5117 }%
5118 {%
5119 \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5120 }%

```

```

5121 }%
5122 {%

5123 \ifdefvoid{\@gls@location}%
5124 {%
5125 \ifdefvoid{\@gls@loclist}
5126 {%
5127 \glossentry{#1}{}%
5128 }%
5129 {%
5130 \glossentry{#1}%
5131 {%
5132 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5133 }%
5134 }%
5135 }%
5136 {%
5137 \glossentry{#1}%
5138 {%
5139 \glossaryentrynumbers{\@gls@location}%
5140 }%
5141 }%
5142 }%
5143 }%
5144 {}%
5145 }

```

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`

```

5146 \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`

```

5147 \newcommand*{\@glsxtr@providenewgls}{%
5148 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
5149 \let\@glsxtr@providenewgls\relax
5150 }

```

`identifyglslike` Identify the command given in the second argument for the benefit of **bib2gls**.

```

5151 \newcommand{\@glsxtridentifyglslike}[2]{%
5152 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
5153 }%
5154 {%

```



```

5155 \@glxstr@providenewgls
5156 \protected@write\@auxout{}\string\@glxstr@newglslike{#1}\string#2}}%
5157 }%
5158 }

```

`\@glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}{<inner cs name>}`

```

5159 \newcommand*{\@glxstrnewgls}[4]{%
5160 \ifdef{#3}%
5161 {%
5162 \PackageError{glossaries-extra}{Command \string#3\space already
5163 defined}{}%
5164 }%
5165 {%

```

Write information to the aux file for bib2gls.

```

5166 \glxstridentifyglslike{#2}{#3}%
5167 \ifcsdef{@#4like@#2}%
5168 {%
5169 \advance\@glxstrnewgls@inner by \@ne
5170 \def\@glxstrnewgls@innercsname{@#4like\number\@glxstrnewgls@inner @#2}%
5171 }%
5172 {\def\@glxstrnewgls@innercsname{@#4like@#2}}%
5173 \expandafter\newrobustcmd\expandafter*\expandafter
5174 #3\expandafter{\expandafter\@glshyp@opt\csname\@glxstrnewgls@innercsname\endcsname}%
5175 \ifstrempy{#1}%
5176 {%
5177 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][]{%
5178 \new@ifnextchar[%
5179 {\csname @#4@\endcsname{##1}{#2##2}}%
5180 {\csname @#4@\endcsname{##1}{#2##2}[]}%
5181 }%
5182 }%
5183 {%
5184 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][]{%
5185 \new@ifnextchar[%
5186 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5187 {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
5188 }%
5189 }%
5190 }%
5191 }

```

`\glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```
5192 \newrobustcmd*{\glstrnewgls}[3] [] {%
5193   \@glstrnewgls{#1}{#2}{#3}{gls}%
5194 }
```

glstrnewglslike Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5195 \newrobustcmd*{\glstrnewglslike}[6] [] {%
5196   \@glstrnewgls{#1}{#2}{#3}{gls}%
5197   \@glstrnewgls{#1}{#2}{#4}{glspl}%
5198   \@glstrnewgls{#1}{#2}{#5}{Gls}%
5199   \@glstrnewgls{#1}{#2}{#6}{Glspl}%
5200 }
```

glstrnewGLSlike Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5201 \newrobustcmd*{\glstrnewGLSlike}[4] [] {%
5202   \@glstrnewgls{#1}{#2}{#3}{GLS}%
5203   \@glstrnewgls{#1}{#2}{#4}{GLSpl}%
5204 }
```

\glstrnewrgls As `\glstrnewgls` but for `\rgls`.

```
5205 \newrobustcmd*{\glstrnewrgls}[3] [] {%
5206   \@glstrnewgls{#1}{#2}{#3}{rgls}%
5207 }
```

glstrnewrglslike As `\glstrnewglslike` but for `\rgls` etc.

```
5208 \newrobustcmd*{\glstrnewrglslike}[6] [] {%
5209   \@glstrnewgls{#1}{#2}{#3}{rgls}%
5210   \@glstrnewgls{#1}{#2}{#4}{rglspl}%
5211   \@glstrnewgls{#1}{#2}{#5}{rGls}%
5212   \@glstrnewgls{#1}{#2}{#6}{rGlspl}%
5213 }
```

glstrnewrGLSlike As `\glstrnewGLSlike` but for `\rGLS` etc.

```
5214 \newrobustcmd*{\glstrnewrGLSlike}[4] [] {%
5215   \@glstrnewgls{#1}{#2}{#3}{rGLS}%
5216   \@glstrnewgls{#1}{#2}{#4}{rGLSpl}%
5217 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
5218 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
5219   \ifcsdef{glo@glsdetoklabel{#1}@recordcount}%
5220   {\csname glo@glsdetoklabel{#1}@recordcount\endcsname}%
5221   {0}%
5222 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
5223 \newcommand*{\GlsXtrRecordCount}[2]{%
5224 \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2}%
5225 {\csname glo@glstetoklabel{#1}@recordcount.#2\endcsname}%
5226 {0}%
5227 }
```

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.

```
5228 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
5229 \ifcsdef{glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}}%
5230 {\csname glo@glstetoklabel{#1}@recordcount.#2.\glstetoklocation{#3}\endcsname}%
5231 {0}%
5232 }
```

trdetoklocation

```
5233 \newcommand*{\glstetoklocation}[1]{#1}
```

ablerecordcount

```
5234 \newcommand*{\glstrenablerecordcount}{%
5235 \renewcommand*{\gls}{\rgls}%
5236 \renewcommand*{\Gls}{\rGls}%
5237 \renewcommand*{\glspl}{\rglspl}%
5238 \renewcommand*{\Glspl}{\rGlspl}%
5239 \renewcommand*{\GLS}{\rGLS}%
5240 \renewcommand*{\GLSpl}{\rGLSpl}%
5241 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
5242 \newcommand*{\glstxrrecordtriggervalue}[1]{%
5243 \GlsXtrTotalRecordCount{#1}%
5244 }
```

dCountAttribute

```
5245 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
5246 \@for\@glstxr@cat:=#1\do
5247 {%
5248 \ifdefempty{\@glstxr@cat}{}%
5249 {%
5250 \glsssetcategoryattribute{\@glstxr@cat}{recordcount}{#2}%
5251 }%
5252 }%
5253 }
```

trifrecordtrigger

<code>\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}</code>

```
5254 \newcommand*{\glxtrifrecordtrigger}[3]{%
5255   \glshasattribute{#1}{recordcount}%
5256   {%
5257     \ifnum\glxtrrecordtriggervalue{#1}>\glsetattribute{#1}{recordcount}\relax
5258     #3%
5259   \else
5260     #2%
5261   \fi
5262 }%
5263 {#3}%
5264 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
5265 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
5266   \edef\glslabel{\glsdetoklabel{#2}}%
5267   \let\@gls@link@label\glslabel
5268   \def\@glxtr@thevalue{%
5269     \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5270     \def\@glsnumberformat{glstriggerrecordformat}%
5271     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5272     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5273     \def\@glxtr@thevalue{%
5274       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5275       \glxtrinitwrgloss
5276       \glslinkpresetkeys
5277       \setkeys{glslink}{#1}%
5278       \glslinkpostsetkeys
5279       \ifdefempty{\@glxtr@thevalue}%
5280       {%
5281         \@gls@saveentrycounter
5282       }%
5283       {%
5284         \let\thegl Sentrycounter\@glxtr@thevalue
5285         \def\theHglSentrycounter{\@glxtr@theHvalue}%
5286       }%
5287       \ifglxtrinitwrglossbefore
5288         \@do@wrglossary{#2}%
5289       \fi
5290       #3%
5291       \ifglxtrinitwrglossbefore
5292       \else
5293         \@do@wrglossary{#2}%
5294       \fi
5295       \ifKV@glslink@local
5296         \glslocalunset{#2}%

```

```

5297 \else
5298   \glsunset{#2}%
5299 \fi
5300 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5301 \newcommand*{\glstriggerrecordformat}[1]{

```

```

\rgls

```

```

5302 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rgls

```

```

5303 \newcommand*{\@rgls}[2][{}]{%
5304   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5305 }

```

```

\@rgls@

```

```

5306 \def\@rgls@#1#2[#3]{%
5307   \glstrifrecordtrigger{#2}%
5308   {%
5309     \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5310   }%
5311   {%
5312     \@gls@{#1}{#2}[#3]%
5313   }%
5314 }%

```

```

\rglspl

```

```

5315 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

```

```

\@rglspl

```

```

5316 \newcommand*{\@rglspl}[2][{}]{%
5317   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5318 }

```

```

\@rglspl@

```

```

5319 \def\@rglspl@#1#2[#3]{%
5320   \glstrifrecordtrigger{#2}%
5321   {%
5322     \@glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5323   }%
5324   {%
5325     \@glspl@{#1}{#2}[#3]%
5326   }%
5327 }%

```

```

\rgls

```

```

5328 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rGls
5329 \newcommand*{\@rGls}[2][{}]{%
5330   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5331 }

\@rGls@
5332 \def\@rGls@#1#2[#3]{%
5333   \glstrifrecordtrigger{#2}%
5334   {%
5335     \@glstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5336   }%
5337   {%
5338     \@Gls@{#1}{#2}[#3]%
5339   }%
5340 }%

\rGlspl
5341 \newrobustcmd*{\rGlspl}{\@glshyp@opt\@rGlspl}

\@rGlspl
5342 \newcommand*{\@rGlspl}[2][{}]{%
5343   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5344 }

\@rGlspl@
5345 \def\@rGlspl@#1#2[#3]{%
5346   \glstrifrecordtrigger{#2}%
5347   {%
5348     \@glstr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5349   }%
5350   {%
5351     \@Glspl@{#1}{#2}[#3]%
5352   }%
5353 }%

\rGLS
5354 \newrobustcmd*{\rGLS}{\@glshyp@opt\@rGLS}

\@rGLS
5355 \newcommand*{\@rGLS}[2][{}]{%
5356   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5357 }

\@rGLS@
5358 \def\@rGLS@#1#2[#3]{%
5359   \glstrifrecordtrigger{#2}%
5360   {%
5361     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%

```

```

5362 }%
5363 {%
5364 \@GLS@{#1}{#2}[#3]%
5365 }%
5366 }%

```

\rGLSpl

```
5367 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}
```

\@rGLSpl

```

5368 \newcommand*{\@rGLSpl}[2][{}]{%
5369 \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[{}]}%
5370 }

```

\@rGLSpl@

```

5371 \def\rGLSpl@#1#2[#3]{%
5372 \glsxtrifrecordtrigger{#2}%
5373 {%
5374 \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5375 }%
5376 {%
5377 \@GLSpl@{#1}{#2}[#3]%
5378 }%
5379 }%

```

\rglsformat

```

5380 \newcommand*{\rglsformat}[2]{%
5381 \glsifregular{#1}
5382 {\glsentryfirst{#1}}%
5383 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5384 }

```

\rglsplformat

```

5385 \newcommand*{\rglsplformat}[2]{%
5386 \glsifregular{#1}
5387 {\glsentryfirstplural{#1}}%
5388 {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5389 }

```

\rGlsformat

```

5390 \newcommand*{\rGlsformat}[2]{%
5391 \glsifregular{#1}
5392 {\Glsentryfirst{#1}}%
5393 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5394 }

```

\rGlsplformat

```
5395 \newcommand*{\rGlsplformat}[2]{%
```

```

5396 \glsifregular{#1}
5397 {\Glsentryfirstplural{#1}}%
5398 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5399 }

```

\rGLSformat

```

5400 \newcommand*{\rGLSformat}[2]{%
5401 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5402 }

```

\rGLSplformat

```

5403 \newcommand*{\rGLSplformat}[2]{%
5404 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5405 }

```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into \@gls@link (through \glsxtr@inc@linkcount) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The “link” part of the name refers to \@gls@link not \hyperlink.)

\@inc@linkcount This performs the actual incrementing and counter definition. The counter is given by \c@glxtr@linkcount@<label> where <label> is the entry’s label. Since this is performed within \@gls@link the label can be accessed with \glslabel.

```

5406 \newcommand{\@glsxtr@do@inc@linkcount}{%
    Does this entry have the linkcount attribute set?
5407 \glsifattribute{\glslabel}{linkcount}{true}%
5408 {%
    Does the counter exist?
5409 \ifcsdef{c@glxtr@linkcount@\glslabel}{}%
5410 {%
    Counter doesn't exist, so define it.
5411 \newcounter{glxtr@linkcount@\glslabel}%
    If linkcountmaster is set, add to counter reset.
5412 \glshasattribute{\glslabel}{linkcountmaster}%
5413 {%
    Need to ensure values are fully expanded.
5414 \begingroup
5415 \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@\glslabel}%
5416 {\glsggetattribute{\glslabel}{linkcountmaster}}}%
5417 \x
5418 }%

```



```

5419     {}%
5420   }%
    Increment counter:
5421   \glxstrinclinkcounter{glxstr@linkcount@glslabel}%
5422 }%
5423 {}%
5424 }

rlinkcounter   May be redefined to use \refstepcounter if required.
5425 \newcommand*{\glxstrinclinkcounter}[1]{\stepcounter{#1}}

linkCounterValue   Expands to the associated link counter register or 0 if not defined.
5426 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5427   \ifcsundef{c@glxstr@linkcount@#1}{0}{\csname c@glxstr@linkcount@#1\endcsname}%
5428 }

rTheLinkCounter   Expands to the display value of the associated link counter or 0 if not defined.
5429 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5430   \ifcsundef{theglsxtr@linkcount@#1}{0}%
5431   {\csname theglxstr@linkcount@#1\endcsname}%
5432 }

fLinkCounterDef   Tests if the counter has been defined
5433 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5434   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5435 }

LinkCounterName   Expands to the associated link counter name. (No check for existence.)
5436 \newcommand*{\GlsXtrLinkCounterName}[1]{glxstr@linkcount@#1}

ableLinkCounting   \GlsXtrEnableLinkCounting[master counter]{categories}

    Enable link counting for the given categories.
5437 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5438   \let\glxstr@inc@linkcount\@glxstr@do@inc@linkcount
5439   \@for\@glxstr@label:=#2\do
5440   {%
5441     \glsssetcategoryattribute{\@glxstr@label}{linkcount}{true}%
5442     \ifstrempy{#1}{}%
5443     {%
5444       \ifcsundef{c@#1}%
5445       {\@nocounterr{#1}}%
5446       {\glsssetcategoryattribute{\@glxstr@label}{linkcountmaster}{#1}}%
5447     }%
5448   }%
5449 }
5450 \@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.

```
5451 \@ifpackageloaded{glossaries-accsupp}
5452 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5453 \newcommand*{\glsaccessname}[1]{%
5454 \glsnameaccessdisplay
5455 {%
5456 \glstryname{#1}%
5457 }%
5458 {#1}%
5459 }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
5460 \newcommand*{\Glsaccessname}[1]{%
5461 \glsnameaccessdisplay
5462 {%
5463 \Glsstryname{#1}%
5464 }%
5465 {#1}%
5466 }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
5467 \newcommand*{\GLSaccessname}[1]{%
5468 \glsnameaccessdisplay
5469 {%
5470 \mfirstucMakeUppercase{\glstryname{#1}}%
5471 }%
5472 {#1}%
5473 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
5474 \newcommand*{\glsaccesstext}[1]{%
5475 \glstextaccessdisplay
5476 {%
5477 \glstrytext{#1}%
5478 }%
5479 {#1}%
5480 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
5481 \newcommand*{\Glsaccesstext}[1]{%
5482   \glstextaccessdisplay
5483   {%
5484     \Glsentrytext{#1}%
5485   }%
5486   {#1}%
5487 }
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```
5488 \newcommand*{\GLSaccesstext}[1]{%
5489   \glstextaccessdisplay
5490   {%
5491     \mfirstucMakeUppercase{\Glsentrytext{#1}}%
5492   }%
5493   {#1}%
5494 }
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```
5495 \newcommand*{\glsaccessplural}[1]{%
5496   \glspluralaccessdisplay
5497   {%
5498     \Glsentryplural{#1}%
5499   }%
5500   {#1}%
5501 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
5502 \newcommand*{\GLSaccessplural}[1]{%
5503   \glspluralaccessdisplay
5504   {%
5505     \Glsentryplural{#1}%
5506   }%
5507   {#1}%
5508 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) converted to upper case.

```
5509 \newcommand*{\GLSaccessplural}[1]{%
5510   \glspluralaccessdisplay
5511   {%
5512     \mfirstucMakeUppercase{\Glsentryplural{#1}}%
5513   }%
5514   {#1}%
5515 }
```

`\glsaccessfirst` Display the first value (no link and no check for existence).

```

5516 \newcommand*{\glsaccessfirst}[1]{%
5517   \glsfirstaccessdisplay
5518   {%
5519     \glstryfirst{#1}%
5520   }%
5521   {#1}%
5522 }

```

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5523 \newcommand*{\Glsaccessfirst}[1]{%
5524   \glsfirstaccessdisplay
5525   {%
5526     \Glsentryfirst{#1}%
5527   }%
5528   {#1}%
5529 }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5530 \newcommand*{\GLSaccessfirst}[1]{%
5531   \glsfirstaccessdisplay
5532   {%
5533     \mfirstucMakeUppercase{\glstryfirst{#1}}%
5534   }%
5535   {#1}%
5536 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5537 \newcommand*{\glsaccessfirstplural}[1]{%
5538   \glsfirstpluralaccessdisplay
5539   {%
5540     \glstryfirstplural{#1}%
5541   }%
5542   {#1}%
5543 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5544 \newcommand*{\Glsaccessfirstplural}[1]{%
5545   \glsfirstpluralaccessdisplay
5546   {%
5547     \Glsentryfirstplural{#1}%
5548   }%
5549   {#1}%
5550 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5551 \newcommand*{\GLSaccessfirstplural}[1]{%

```

```

5552   \glsfirstpluralaccessdisplay
5553   {%
5554       \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5555   }%
5556   {#1}%
5557   }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

5558   \newcommand*{\glsaccesssymbol}[1]{%
5559       \glssymbolaccessdisplay
5560       {%
5561           \glsentrysymbol{#1}%
5562       }%
5563       {#1}%
5564   }

```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5565   \newcommand*{\Glsaccesssymbol}[1]{%
5566       \glssymbolaccessdisplay
5567       {%
5568           \Glsentrysymbol{#1}%
5569       }%
5570       {#1}%
5571   }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5572   \newcommand*{\GLSaccesssymbol}[1]{%
5573       \glssymbolaccessdisplay
5574       {%
5575           \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5576       }%
5577       {#1}%
5578   }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5579   \newcommand*{\glsaccesssymbolplural}[1]{%
5580       \glssymbolpluralaccessdisplay
5581       {%
5582           \glsentrysymbolplural{#1}%
5583       }%
5584       {#1}%
5585   }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5586   \newcommand*{\Glsaccesssymbolplural}[1]{%
5587       \glssymbolpluralaccessdisplay

```

```

5588   {%
5589     \Glsentrysymbolplural{#1}%
5590   }%
5591   {#1}%
5592 }

```

`\esssymbolplural` Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5593 \newcommand*{\GLSaccesssymbolplural}[1]{%
5594   \glssymbolpluralaccessdisplay
5595   {%
5596     \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5597   }%
5598   {#1}%
5599 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5600 \newcommand*{\glsaccessdesc}[1]{%
5601   \glsdescriptionaccessdisplay
5602   {%
5603     \glentrydesc{#1}%
5604   }%
5605   {#1}%
5606 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5607 \newcommand*{\Glsaccessdesc}[1]{%
5608   \glsdescriptionaccessdisplay
5609   {%
5610     \Glsentrydesc{#1}%
5611   }%
5612   {#1}%
5613 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5614 \newcommand*{\GLSaccessdesc}[1]{%
5615   \glsdescriptionaccessdisplay
5616   {%
5617     \mfirstucMakeUppercase{\glentrydesc{#1}}%
5618   }%
5619   {#1}%
5620 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence).

```

5621 \newcommand*{\glsaccessdescplural}[1]{%
5622   \glsdescriptionpluralaccessdisplay
5623   {%
5624     \glentrydescplural{#1}%

```

```

5625     }%
5626     {#1}%
5627 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5628 \newcommand*\Glsaccessdescplural[1]{%
5629   \glsdescriptionpluralaccessdisplay
5630   {%
5631     \Glsentrydescplural{#1}%
5632   }%
5633   {#1}%
5634 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5635 \newcommand*\GLSaccessdescplural[1]{%
5636   \glsdescriptionpluralaccessdisplay
5637   {%
5638     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5639   }%
5640   {#1}%
5641 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5642 \newcommand*\glsaccessshort[1]{%
5643   \glsshortaccessdisplay
5644   {%
5645     \glsentryshort{#1}%
5646   }%
5647   {#1}%
5648 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5649 \newcommand*\Glsaccessshort[1]{%
5650   \glsshortaccessdisplay
5651   {%
5652     \Glsentryshort{#1}%
5653   }%
5654   {#1}%
5655 }

```

`\GLSAccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5656 \newcommand*\GLSAccessshort[1]{%
5657   \glsshortaccessdisplay
5658   {%
5659     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5660   }%

```

```

5661     {#1}%
5662 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5663 \newcommand*{\lsaccessshortpl}[1]{%
5664   \glshortpluralaccessdisplay
5665   {%
5666     \glentryshortpl{#1}%
5667   }%
5668   {#1}%
5669 }

```

`\lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5670 \newcommand*{\Glsaccessshortpl}[1]{%
5671   \glshortpluralaccessdisplay
5672   {%
5673     \Glsentryshortpl{#1}%
5674   }%
5675   {#1}%
5676 }

```

`\LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5677 \newcommand*{\GLSaccessshortpl}[1]{%
5678   \glshortpluralaccessdisplay
5679   {%
5680     \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5681   }%
5682   {#1}%
5683 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

5684 \newcommand*{\glsaccesslong}[1]{%
5685   \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5686 }

```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

5687
5688 \newcommand*{\Glsaccesslong}[1]{%
5689   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5690 }

```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```

5691 \newcommand*{\GLSaccesslong}[1]{%
5692   \glslongaccessdisplay
5693   {%
5694     \mfirstucMakeUppercase{\glentrylong{#1}}%
5695   }%

```



```

5696     {#1}%
5697 }

glsaccesslongpl  Display the long plural form (no link and no check for existence).
5698   \newcommand*{\glsaccesslongpl}[1]{%
5699     \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5700 }

Glsaccesslongpl  Display the long plural form (no link and no check for existence).
5701
5702   \newcommand*{\Glsaccesslongpl}[1]{%
5703     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5704 }

GLSaccesslongpl  Display the longplural value (no link and no check for existence) converted to upper case.
5705   \newcommand*{\GLSaccesslongpl}[1]{%
5706     \glslongpluralaccessdisplay
5707     {%
5708       \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5709     }%
5710     {#1}%
5711 }

    Keys for accessibility support.
5712   \define@key{glsxtrabrv}{access}{%
5713     \def\@gls@nameaccess{#1}%
5714 }
5715   \define@key{glsxtrabrv}{textaccess}{%
5716     \def\@gls@textaccess{#1}%
5717 }
5718   \define@key{glsxtrabrv}{firstaccess}{%
5719     \def\@gls@firstaccess{#1}%
5720 }
5721   \define@key{glsxtrabrv}{shortaccess}{%
5722     \def\@gls@shortaccess{#1}%
5723 }
5724   \define@key{glsxtrabrv}{shortpluralaccess}{%
5725     \def\@gls@shortaccesspl{#1}%
5726 }

@initaccesskeys
5727   \newcommand*{\@gls@initaccesskeys}{%
5728     \def\@gls@nameaccess{}%
5729     \def\@gls@textaccess{}%
5730     \def\@gls@firstaccess{}%
5731     \def\@gls@shortaccess{}%
5732     \def\@gls@shortaccesspl{}%
5733 }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```
5734 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5735   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5736   {#2}%
5737   {%
5738     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5739     {#3}%
5740     {%
5741       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5742       {#2}%
5743       {#3}%
5744     }%
5745   }%
5746 }
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5747 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
5748 \ifdefempty\@gls@shortaccess
5749 {%
5750   \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5751   {%
5752     \@glsxtr@insertdots\@gls@shortaccess{#1}%
5753     \eappto\ExtraCustomAbbreviationFields{%
5754       shortaccess={\expandonce\@gls@shortaccess},}%
5755     }%
5756   }%
5757 }%
5758 }
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5759 \ifdefempty\@gls@shortaccess
5760 {}%
5761 {%
5762   \ifdefempty\@gls@shortaccesspl
5763   {%
5764     \@gls@ifaccessattribute@set{aposplural}%
5765     {%
5766       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5767         \@gls@shortaccess'\abbrvpluralsuffix}%
5768     }%
5769   }%
5770   \@gls@ifaccessattribute@set{noshortplural}%
5771   {%
```

```

5772         \let\@gls@shortaccesspl\@gls@shortaccess
5773     }%
5774     {%
5775         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5776             \@gls@shortaccess\abbrvpluralsuffix}%
5777     }%
5778 }%
5779 \eappto\ExtraCustomAbbreviationFields{%
5780     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5781 }%
5782 {}%
5783 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5784 \ifdefempty\@gls@nameaccess
5785 {%
5786     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5787     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5788         \ifdefempty\@gls@shortaccess
5789         {}%
5790         {%
5791             \eappto\ExtraCustomAbbreviationFields{%
5792                 access={\expandonce\@gls@shortaccess},%
5793             }%
5794         }%
5795     }%
5796     {}%
5797 }%
5798 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5799 \ifdefempty\@gls@textaccess
5800 {%
5801     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5802     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5803         \ifdefempty\@gls@shortaccess
5804         {}%
5805         {%
5806             \eappto\ExtraCustomAbbreviationFields{%
5807                 textaccess={\expandonce\@gls@shortaccess},%
5808             }%
5809         }%
5810     }%
5811     {}%
5812 }%
5813 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5814 \ifdefempty\@gls@firstaccess
5815 {%
5816 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5817 {%
```

Do nothing if the shortaccess key hasn't been set.

```
5818 \ifdefempty\@gls@shortaccess
5819 {}%
5820 {%
5821 \eappto\ExtraCustomAbbreviationFields{%
5822 firstaccess={\expandonce\@gls@shortaccess},%
5823 }%
5824 }%
5825 }%
5826 {}%
5827 }%
5828 {}%
5829 }
```

End of if accsupp part

```
5830 }
5831 {
```

No accessibility support. Just define these commands to do \glsentry<xxx>

```
\glsaccessname Display the name value (no link and no check for existence).
5832 \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.
5833 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
5834 \newcommand*{\GLSaccessname}[1]{%
5835 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}}

\glsaccesstext Display the text value (no link and no check for existence).
5836 \newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to
upper case.
5837 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
5838 \newcommand*{\GLSaccesstext}[1]{%
5839 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}}

glsaccessplural Display the plural value (no link and no check for existence).
5840 \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.

5841 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.

5842 `\newcommand*{\GLSaccessplural}[1]{%`
5843 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).

5844 `\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.

5845 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

5846 `\newcommand*{\GLSaccessfirst}[1]{%`
5847 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).

5848 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

5849 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

5850 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5851 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

5852 `\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.

5853 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

5854 `\newcommand*{\GLSaccesssymbol}[1]{%`
5855 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

5856 `\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.

5857 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
5858 \newcommand*{\GLSaccesssymbolplural}[1]{%
5859 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
5860 \newcommand*{\glsaccessdesc}[1]{\glentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
5861 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
5862 \newcommand*{\GLSaccessdesc}[1]{%
5863 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
5864 \newcommand*{\glsaccessdescplural}[1]{\glentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
5865 \newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
5866 \newcommand*{\GLSaccessdescplural}[1]{%
5867 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
5868 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
5869 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
5870 \newcommand*{\GLSaccessshort}[1]{%
5871 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
5872 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
5873 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.

```
5874 \newcommand*{\GLSaccessshortpl}[1]{%
5875 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

\glsaccesslong Display the long form (no link and no check for existence).

```
5876 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```
5877 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.

```
5878 \newcommand*{\GLSaccesslong}[1]{%
5879 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5880 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5881 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```
5882 \newcommand*{\GLSaccesslongpl}[1]{%
5883 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
5884 \newcommand*{@gls@initaccesskeys}{}%
```

lt@short@access This does nothing if there's no accessibility support.

```
5885 \newcommand{\@gls@setup@default@short@access}[1]{%
End of else part
5886 }
```

1.6 Categories

\glscategory Add a new storage key that can be used to indicate a category. The default category is general.

```
5887 \glsaddstoragekey{category}{general}{\glscategory}
```

\glsifcategory Convenient shortcut to determine if an entry has the given category.

```
5888 \newcommand{\glsifcategory}[4]{%
5889 \ifglstest{#1}{category}{#2}{#3}{#4}%
5890 }
```

Categories can have attributes.

categoryattribute

```
\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
5891 \newcommand*{\glissetcategoryattribute}[3]{%  
5892   \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%  
5893 }
```

categoryattribute

```
\glsggetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5894 \newcommand*{\glsggetcategoryattribute}[2]{%  
5895   \csuse{@glsxtr@categoryattr@@#1@#2}%  
5896 }
```

categoryattribute

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5897 \newcommand*{\glshascategoryattribute}[4]{%  
5898   \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%  
5899 }
```

\glissetattribute

```
\glissetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5900 \newcommand*{\glissetattribute}[3]{%  
5901   \glissetcategoryattribute{\glscategory{#1}}{#2}{#3}%  
5902 }
```

\glsggetattribute

```
\glsggetattribute{<entry label>}{<attribute-label>}
```

Short cut where the category label is obtained from the entry information.

```
5903 \newcommand*{\glsggetattribute}[2]{%  
5904   \glsggetcategoryattribute{\glscategory{#1}}{#2}%  
5905 }
```


\glshasattribute

\glshasattribute{<entry label>}{<attribute-label>}{<true>}{<false>}

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
5906 \newcommand*{\glshasattribute}[4]{%
5907   \ifglentryexists{#1}%
5908   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5909   {#4}%
5910 }
```

categoryattribute

\glcifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}

True if category has the attribute with the given value.

```
5911 \newcommand{\glcifcategoryattribute}[5]{%
5912   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5913   {#5}%
5914   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5915 }
```

\glcifattribute

\glcifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}

Short cut to determine if the given entry has a category with the given attribute set.

```
5916 \newcommand{\glcifattribute}[5]{%
5917   \ifglentryexists{#1}%
5918   {\glcifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5919   {#5}%
5920 }
```

Set attributes for the default general category:

```
5921 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5922 \glsetcategoryattribute{acronym}{regular}{true}
```

regularcategory

Convenient shortcut to create add the regular attribute.

```
5923 \newcommand*{\glsetregularcategory}[1]{%
5924   \glsetcategoryattribute{#1}{regular}{true}%
5925 }
```

fregularcategory

```
\glsifregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5926 \newcommand{\glsifregularcategory}[3]{%  
5927   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%  
5928 }
```

tregularcategory

```
\glsifnotregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5929 \newcommand{\glsifnotregularcategory}[3]{%  
5930   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%  
5931 }
```

\glsifregular

```
\glsifregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to true.

```
5932 \newcommand{\glsifregular}[3]{%  
5933   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%  
5934 }
```

\glsifnotregular

```
\glsifnotregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to false.

```
5935 \newcommand{\glsifnotregular}[3]{%  
5936   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%  
5937 }
```

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.

```
5938 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```

5939 \forallglossaries[#1]{#3}%
5940 {%
5941     \forallsentries[#3]{#4}%
5942     {%
5943         \glsifcategory{#4}{#2}{#5}{}%
5944     }%
5945 }%
5946 }

```

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.

```

5947 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5948     \forallglossaries[#1]{#4}%
5949     {%
5950         \forallsentries[#4]{#5}%
5951         {%
5952             \glsifattribute{#5}{#2}{#3}{#6}{}%
5953         }%
5954     }%
5955 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glstrpostdescription`.

```

5956 \ifdef\newterm
5957 {%

```

`\newterm`

```

5958     \renewcommand*{\newterm}[2][ ]{%
5959         \newglossaryentry{#2}%
5960         {type=index,category=index,name={#2},%
5961          description={\glstrpostdescription\nopostdesc},#1}%
5962     }

```

Indexed terms are regular by default.

```

5963     \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5964     \newcommand*{\glstrpostdescindex}{}

5965 }
5966 {}

```

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.

```
5967 \ifdef\printsymbols
5968 {%
```

`glsxtrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```
5969 \newcommand*{\glsxtrnewsymbol}[3] [] {%
5970 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5971 }
```

Symbols are regular by default.

```
5972 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
5973 \newcommand*{\glsxtrpostdescsymbol}{}

5974 }
5975 {%
```

Similar for the numbers option.

```
5976 \ifdef\printnumbers
5977 {%
```

`glsxtrnewnumber`

```
5978 \ifdef\printnumbers
5979 \newcommand*{\glsxtrnewnumber}[3] [] {%
5980 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5981 }
```

Numbers are regular by default.

```
5982 \glsssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
5983 \newcommand*{\glsxtrpostdescnumber}{}

5984 }
5985 {%
```

`glsxtrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5986 \newcommand*{\glsxtrsetcategory}[2] {%
5987 \@for\@glsxtr@label:=#1\do
5988 {%
5989 \glsfieldxddef{\@glsxtr@label}{category}{#2}%
5990 }%
5991 }
```

`categoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```

5992 \newcommand*{\glxtrsetcategoryforall}[2]{%
5993   \forallglossaries[#1]{\@glxtr@type}{%
5994     \forallglsentries[\@glxtr@type]{\@glxtr@label}%
5995     {%
5996       \glsfieldxdef{\@glxtr@label}{category}{#2}%
5997     }%
5998   }%
5999 }

```

`trfieldtitlecase` `\glxtrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```

6000 \newcommand*{\glxtrfieldtitlecase}[2]{%
6001   \expandafter\glxtrfieldtitlecasesecs\expandafter
6002   {\csname glo\glsdetoklabel{#1}@#2\endcsname}%
6003 }

```

`fieldtitlecasesecs` The command used by `\glxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```

6004 \newcommand*{\glxtrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}

```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

6005 \ifpackageloaded{glossaries-accsupp}
6006 {
6007   \renewcommand*{\glossentrydesc}[1]{%
6008     \glsdoifexistsorwarn{#1}%
6009     {%
6010       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossdescfont` attribute to determine the font applied.

```

6011   \glshasattribute{#1}{glossdescfont}%
6012   {%
6013     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6014     \ifcsdef{\@glxtr@attrval}%
6015     {%
6016       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6017     }%
6018     {%
6019       \GlossariesExtraWarning{Unknown control sequence name
6020         '\@glxtr@attrval' supplied in glossdescfont attribute

```

```

6021         for entry '#1'. Ignoring}%
6022         \let\@glxtr@glossdescfont\@firstofone
6023     }%
6024 }%
6025 {\let\@glxtr@glossdescfont\@firstofone}%
6026 \glsifattribute{#1}{glossdesc}{firstuc}%
6027 {%
6028     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6029 }%
6030 {%
6031     \glsifattribute{#1}{glossdesc}{title}%
6032     {%
6033         \@glxtr@do@titlecaps@warn
6034         \glsdescriptionaccessdisplay
6035         {%
6036             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6037         }%
6038         {#1}%
6039     }%
6040     {%
6041         \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6042     }%
6043 }%
6044 }%
6045 }
6046 }
6047 {
6048 \renewcommand*{\glossentrydesc}[1]{%
6049     \glsdoifexistsorwarn{#1}%
6050     {%
6051         \glssetabbrvfmt{\glscategory{#1}}%
6052         \glsattribute{#1}{glossdescfont}%
6053         {%
6054             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6055             \ifcsdef{\@glxtr@attrval}%
6056             {%
6057                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6058             }%
6059             {%
6060                 \GlossariesExtraWarning{Unknown control sequence name
6061                     '\@glxtr@attrval' supplied in glossdescfont attribute
6062                     for entry '#1'. Ignoring}%
6063                 \let\@glxtr@glossdescfont\@firstofone
6064             }%
6065         }%
6066         {\let\@glxtr@glossdescfont\@firstofone}%
6067         \glsifattribute{#1}{glossdesc}{firstuc}%
6068         {%
6069             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%

```

```

6070 }%
6071 {%
6072   \glsifattribute{#1}{glossdesc}{title}%
6073   {%
6074     \@glsxtr@do@titlecaps@warn
6075     \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
6076   }%
6077   {%
6078     \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
6079   }%
6080 }%
6081 }%
6082 }
6083 }

```

`\glossentryname` If the `glossname` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

6084 \ifpackageloaded{glossaries-accsupp}
6085 {
6086   \renewcommand*{\glossentryname}[1]{%
6087     \@glsdoifexistsorwarn{#1}%
6088     {%
6089       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

6090   \glsifattribute{#1}{glossnamefont}%
6091   {%
6092     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6093     \ifcsdef{\@glsxtr@attrval}%
6094     {%
6095       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6096     }%
6097     {%
6098       \GlossariesExtraWarning{Unknown control sequence name
6099         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
6100         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6101       \let\@glsxtr@glossnamefont\glsnamefont
6102     }%
6103   }%
6104   {\let\@glsxtr@glossnamefont\glsnamefont}%
6105   \glsifattribute{#1}{glossname}{firstuc}%
6106   {%
6107     \glsnameaccessdisplay
6108     {%
6109       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6110     }%
6111     {#1}%
6112   }%
6113   {%
6114     \glsifattribute{#1}{glossname}{title}%

```

```

6115      {%
6116      \@glstr@do@titlecaps@warn
6117      \glsnameaccessdisplay
6118      {%
6119      \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}}%
6120      }%
6121      {#1}%
6122      }%
6123      {%
6124      \glsifattribute{#1}{glossname}{uc}%
6125      {%
6126      \glsnameaccessdisplay
6127      {%

```

Hide the label from the upper-casing command.

```

6128      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6129      \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6130      }%
6131      {#1}%
6132      }%
6133      {%
6134      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6135      \glsnameaccessdisplay
6136      {%
6137      \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6138      }%
6139      {#1}%
6140      }%
6141      }%
6142      }%

```

Do post-name hook:

```

6143      \glstrpostnamehook{#1}%
6144      }%
6145  }
6146 }
6147 {
6148 \renewcommand*{\glossentryname}[1]{%
6149   \@glsdoifexistsorwarn{#1}%
6150   {%
6151     \glssetabbrvfmt{\glscategory{#1}}%
6152     \glshasattribute{#1}{glossnamefont}%
6153     {%
6154       \edef\@glstr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6155       \ifcsdef{\@glstr@attrval}%
6156       {%
6157         \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6158       }%
6159     }%
6160     \GlossariesExtraWarning{Unknown control sequence name

```



```

6161         ‘\@glxtr@attrval’ supplied in glossnamefont attribute
6162         for entry ‘#1’. Reverting to default \string\glsnamefont}%
6163         \let\@glxtr@glossnamefont\glsnamefont
6164     }%
6165 }%
6166 {\let\@glxtr@glossnamefont\glsnamefont}%
6167 \glsifattribute{#1}{glossname}{firstuc}%
6168 {%
6169     \@glxtr@glossnamefont{\Glsentryname{#1}}%
6170 }%
6171 {%
6172     \glsifattribute{#1}{glossname}{title}%
6173     {%
6174         \@glxtr@do@titlecaps@warn
6175         \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
6176     }%
6177     {%
6178         \glsifattribute{#1}{glossname}{uc}%
6179         {%

```

Hide the label from the upper-casing command.

```

6180         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
6181         \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6182     }%
6183     {%

```

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.

```

6184         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
6185         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
6186     }%
6187 }%
6188 }%

```

Do post-name hook.

```

6189     \glxtrpostnamehook{#1}%
6190 }%
6191 }
6192 }

```

\Glossentryname Redefine to set the abbreviation format and accessibility support.

```

6193 \@ifpackageloaded{glossaries-accsupp}
6194 {
6195     \renewcommand*{\Glossentryname}[1]{%
6196         \@glstdoifexistsorwarn{#1}%
6197         {%
6198             \glsssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

6199         \glshasattribute{#1}{glossnamefont}%
6200         {%

```

```

6201     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6202     \ifcsdef{\@glsxtr@attrval}%
6203     {%
6204         \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6205     }%
6206     {%
6207         \GlossariesExtraWarning{Unknown control sequence name
6208             '\@glsxtr@attrval' supplied in glossnamefont attribute
6209             for entry '#1'. Reverting to default \string\glsnamefont}%
6210         \let\@glsxtr@glossnamefont\glsnamefont
6211     }%
6212 }%
6213 {\let\@glsxtr@glossnamefont\glsnamefont}%
6214 \glsnameaccessdisplay
6215 {%
6216     \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6217 }%
6218 {#1}%

```

Do post-name hook:

```

6219     \glsxtrpostnamehook{#1}%
6220 }%
6221 }
6222 }
6223 {
6224     \renewcommand*{\Glossentryname}[1]{%
6225         \@glsdoifexistsorwarn{#1}%
6226     {%
6227         \glssetabbrvfmt{\glscategory{#1}}%
6228         \glsattribute{#1}{glossnamefont}%
6229     {%
6230         \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6231         \ifcsdef{\@glsxtr@attrval}%
6232         {%
6233             \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6234         }%
6235         {%
6236             \GlossariesExtraWarning{Unknown control sequence name
6237                 '\@glsxtr@attrval' supplied in glossnamefont attribute
6238                 for entry '#1'. Reverting to default \string\glsnamefont}%
6239             \let\@glsxtr@glossnamefont\glsnamefont
6240         }%
6241     }%
6242     {\let\@glsxtr@glossnamefont\glsnamefont}%
6243     \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6244     \glsxtrpostnamehook{#1}%
6245 }%
6246 }

```

6247 }

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`\glstrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
6248 \newcommand*{\glstrpostnamehook}[1]{%
6249   \let\@glsnumberformat\@glstr@defaultnumberformat
6250   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
6251   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
6252   \csuse{glstrpostname\glscategory{#1}}%
6253 }
```

`\glsextrapostnamehook`

```
6254 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
6255 \newcommand*{\glsdefpostname}[2]{%
6256   \csdef{glstrpostname#1}{#2}%
6257 }
```

`\glssetaccessdisplay`

```
6258 \@ifpackageloaded{glossaries-accsupp}
6259 {
6260   \newcommand*{\glstrsetaccessdisplay}[1]{%
6261     \ifcsdef{gls#1accessdisplay}%
6262       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
6263     {%
```

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.

```
6264     \edef\@gls@thisval{#1}%
6265     \@for\@gls@map:=\@gls@keymap\do{%
6266       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6267       \ifdefequal{\@this@key}{\@gls@thisval}%
6268       {%
6269         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6270         \@endfortrue
6271       }%
6272     }%
6273   }%
6274   \ifcsdef{gls\@gls@thisval accessdisplay}%
6275   {\letcs\@glstr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```

6276      {\let\@glxtr@accessdisplay\@firstoftwo}%
6277    }%
6278  }
6279 }
6280 {%
6281 \newcommand*{\glxtr@setaccessdisplay}[1]{%
6282   \let\@glxtr@accessdisplay\@firstoftwo}
6283 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6284 \newrobustcmd*{\glossentrynameother}[2]{%
6285   \@glxtr@doifexistsorwarn{#1}%
6286   {%

```

Accessibility support:

```

6287   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6288   \glxtr@setabbrvfmt{\glxtr@category{#1}}%
6289   \glxtr@hasattribute{#1}{\glxtr@glossnamefont}%
6290   {%
6291     \edef\@glxtr@attrval{\glxtr@getattribute{#1}{\glxtr@glossnamefont}}%
6292     \ifcsdef{\@glxtr@attrval}%
6293     {%
6294       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6295     }%
6296     {%
6297       \GlossariesExtraWarning{Unknown control sequence name
6298         '\@glxtr@attrval' supplied in glossnamefont attribute
6299         for entry '#1'. Reverting to default \string\glxtr@glossnamefont}%
6300       \let\@glxtr@glossnamefont\glxtr@glossnamefont
6301     }%
6302   }%
6303   {\let\@glxtr@glossnamefont\glxtr@glossnamefont}%
6304   \glxtr@ifattribute{#1}{\glxtr@glossname}{\firstuc}%
6305   {%
6306     \glxtr@accessdisplay
6307     {\@glxtr@glossnamefont{\@glxtr@entry@field{#1}{#2}}}%
6308     {#1}%
6309   }%
6310   {%
6311     \glxtr@ifattribute{#1}{\glxtr@glossname}{\title}%
6312     {%
6313       \glxtr@do@titlecaps@warn
6314       \glxtr@accessdisplay
6315       {\@glxtr@glossnamefont{\glxtr@field@titlecase{#1}{#2}}}%
6316       {#1}%
6317     }%
6318     {%

```

```

6319      \glsifattribute{#1}{glossname}{uc}%
6320      {%
6321      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6322      \@glsxtr@accessdisplay
6323      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6324      {#1}%
6325      }%
6326      {%
6327      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6328      \@glsxtr@accessdisplay
6329      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6330      {#1}%
6331      }%
6332      }%
6333      }%

  Do post-name hook.
6334      \glsxtrpostnamehook{#1}%
6335      }%
6336  }

```

`format@override` Determines if the `format` key should override the indexing attribute value.

```

6337 \newif\if@glsxtr@format@override
6338 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6339 \@ifpackageloaded{hyperref}
6340 {
  If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
  don't add it.
6341   \ifHy@hyperindex
6342     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6343       \@glsxtr@format@override true
6344       \appto\theindex{\let\glshypernumber\@firstofone}%
6345     }
6346   \else
6347     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6348       \@glsxtr@format@override true
6349       \appto\theindex{\let\glshypernumber\hyperpage}%
6350     }
6351   \fi
6352 }
6353 {
6354   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6355     \@glsxtr@format@override true
6356   }

```

```

6357 }
6358 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6359 \newcommand*{\glstrdoautoindexname}[2]{%
6360   \glshasattribute{#1}{#2}%
6361   {%

```

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.

```

6362   \@glstr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

6363   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6364   \if@glstr@format@override

6365     \ifx\@glstrnumberformat\@glstr@defaultnumberformat
6366     \else
6367       \let\@glstr@attrval\@glstrnumberformat
6368     \fi
6369   \fi
6370   \ifdefstring{\@glstr@attrval}{true}%
6371   {}%
6372   {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6373   \expandafter\glstrautoindex\expandafter{\@glo@name}%
6374   }%
6375   {}%
6376 }

```

glstrautoindex

```

6377 \newcommand*{\glstrautoindex}{\index}

```

xtrautoindexesc

```

6378 \newcommand{\glstrautoindexesc}{%
6379   \@gls@checkmkidxchars\@glo@sort
6380   \@glstr@autoindex@doextra@esc\@glo@sort
6381 }

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6382 \newcommand*{\@glstr@autoindex@setname}[1]{%
6383   \protected@edef\@glo@name{\glstrautoindexentry{#1}}%
6384   \glstrautoindexassignsort{\@glo@sort}{#1}%
6385   \glstrautoindexesc
6386   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
6387 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6388 \newcommand*{\glstrautoindexentry}[1]{\string\glsentryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```
6389 \newcommand*{\glxtrautoindexassignsort}[2]{%
6390   \glslentryfield{#1}{#2}{sort}%
6391 }
```

dex@doextra@esc

```
6392 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6393   \ifx\@glxtr@autoindex@esc\@gl@quotechar
6394   \else
6395     \def\@gl@checkedmkidx{}%
6396     \edef\@glxtr@checkspch{%
6397       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6398       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6399       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6400     \@glxtr@checkspch
6401     \let#1\@gl@checkedmkidx\relax
6402   \fi
```

Escape actual character unless it has already been escaped.

```
6403   \ifx\@glxtr@autoindex@at\@gl@actualchar
6404   \else
6405     \def\@gl@checkedmkidx{}%
6406     \edef\@glxtr@checkspch{%
6407       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6408       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6409       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6410     \@glxtr@checkspch
6411     \let#1\@gl@checkedmkidx\relax
6412   \fi
```

Escape level character unless it has already been escaped.

```
6413   \ifx\@glxtr@autoindex@level\@gl@levelchar
6414   \else
6415     \def\@gl@checkedmkidx{}%
6416     \edef\@glxtr@checkspch{%
6417       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6418       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6419       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6420     \@glxtr@checkspch
6421     \let#1\@gl@checkedmkidx\relax
6422   \fi
```

Escape encap character unless it has already been escaped.

```
6423   \ifx\@glxtr@autoindex@encap\@gl@encapchar
6424   \else
6425     \def\@gl@checkedmkidx{}%
6426     \edef\@glxtr@checkspch{%
6427       \noexpand\@glxtr@autoindex@escencap\expandonce{#1}%
6428       \noexpand\@empty\@glxtr@autoindex@encap\noexpand\@nnil
```

```

6429      \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6430      \@glsxtr@checkspch
6431      \let#1\@gls@checkedmkidx\relax
6432      \fi
6433 }

```

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`.

```
6434 \newcommand*{\@glsxtr@autoindex@at}{}

```

`trSetActualChar` Set the actual character.

```

6435 \newcommand*{\GlsXtrSetActualChar}[1]{%
6436   \gdef\@glsxtr@autoindex@at{#1}%
6437   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6438     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6439   }%
6440 }
6441 \@onlypreamble\GlsXtrSetActualChar
6442 \makeatother
6443 \GlsXtrSetActualChar{@}
6444 \makeatletter

```

`autoindex@encap` Encap character for use with `\index`.

```
6445 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`XtrSetEncapChar` Set the encap character.

```

6446 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6447   \gdef\@glsxtr@autoindex@encap{#1}%
6448   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
6449     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6450   }%
6451 }
6452 \GlsXtrSetEncapChar{||}
6453 \@onlypreamble\GlsXtrSetEncapChar

```

`autoindex@level` Level character for use with `\index`.

```
6454 \newcommand*{\@glsxtr@autoindex@level}{}

```

`XtrSetLevelChar` Set the encap character.

```

6455 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6456   \gdef\@glsxtr@autoindex@level{#1}%
6457   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
6458     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
6459   }%
6460 }
6461 \GlsXtrSetLevelChar{!}
6462 \@onlypreamble\GlsXtrSetLevelChar

```


r@autoindex@esc Escape character for use with \index.

```
6463 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
6464 \newcommand*{\GlsXtrSetEscChar}[1]{%
6465   \gdef\@glsxtr@autoindex@esc{#1}%
6466   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6467     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6468   }%
6469 }
6470 \GlsXtrSetEscChar{"}
6471 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
6472 \ifdef\actualchar
6473 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6474 {}
```

Quote character \quotechar:

```
6475 \ifdef\quotechar
6476 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6477 {}
```

Level character \levelchar:

```
6478 \ifdef\levelchar
6479 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6480 {}
```

Encap character \encapchar:

```
6481 \ifdef\encapchar
6482 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6483 {}
```

leto@endescspch

```
6484 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

<code>\@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}</code>

```
6485 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
6486   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
6487   \toks@={#3}%
6488   \ifx\@nnil#3\relax
6489     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
6490   \else
6491     \ifx\@nnil#4\relax
6492       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
6493       \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```

6494      #4#5\@glxtr@endescspch}%
6495  \else
6496    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
6497      \@glxtr@autoindex@esc#1}%
6498    \def\@@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6499  \fi
6500 \fi
6501 \@glxtr@checkspch
6502 }

```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```

6503 \renewcommand*{\Glossentrydesc}[1]{%
6504   \glsdoifexistsorwarn{#1}%
6505   {%
6506     \glssetabbrvfmt{\glscategory{#1}}%
6507     \Glsaccessdesc{#1}%
6508   }%
6509 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

6510 \renewcommand*{\Glossentrysymbol}[1]{%
6511   \glsdoifexistsorwarn{#1}%
6512   {%
6513     \glssetabbrvfmt{\glscategory{#1}}%
6514     \Glsaccesssymbol{#1}%
6515   }%
6516 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

6517 \renewcommand*{\Glossentrysymbol}[1]{%
6518   \glsdoifexistsorwarn{#1}%
6519   {%
6520     \glssetabbrvfmt{\glscategory{#1}}%
6521     \Glsaccesssymbol{#1}%
6522   }%
6523 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

\GlsXtrEnableInitialTagging Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```

6524 \newcommand*{\GlsXtrEnableInitialTagging}{%
6525   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6526 }
6527 \@onlypreamble\GlsXtrEnableInitialTagging

```

\GlsXtrEnableInitialTagging Starred version undefines command.

```

6528 \newcommand*{\s@glxtr@enabletagging}[2]{%
6529   \undef#2%
6530   \@glxtr@enabletagging{#1}{#2}%
6531 }

```

r@enabletagging Internal command.

```

6532 \newcommand*{\@glxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6533   \@for\@glxtr@cat:=#1\do
6534   {%
6535     \ifdefempty\@glxtr@cat
6536     {}%
6537     {\glsssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%
6538   }%
6539   \newrobustcmd*#2[1]{##1}%
6540   \def\@glxtr@taggingcs{#2}%
6541   \renewcommand*{\@glxtr@activate@initialtagging{%
6542     \let#2\@glxtr@tag
6543   }%
6544   \ifundef\@gl@preglossaryhook
6545   {\GlossariesExtraWarning{Initial tagging requires at least
6546     glossaries.sty v4.19 to work correctly}}%
6547   {}%
6548 }

```

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

```

6549 \ifundef\mfu@checkword@do
6550 {
6551   \newcommand*{\mfu@checkword@do}[1]{%
6552     \ifdefstring{\mfu@checkword@arg}{#1}%
6553     {%
6554       \let\@mfu@domakefirstuc\@firstofone
6555       \listbreak
6556     }%
6557   }%
6558 }

```

\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.

```

6559 \ifundef\mfu@checkword
6560 {
6561   \newcommand{\@glxtr@do@titlecaps@warn}{%
6562     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6563       support not available}%

```

One warning should suffice.

```
6564 \let\@glxtr@do@titlecaps@warn\relax
6565 }
6566 }
6567 {
6568 \renewcommand*{\mfu@checkword}[1]{%
6569 \def\mfu@checkword@arg{#1}%
6570 \let\@mfu@domakefirstuc\makefirstuc
6571 \forlistloop\mfu@checkword@do\@mfu@nocaplist
6572 }
6573 }
6574 }
6575 {}% no patch required
```

@titlecaps@warn Do warning if title case not supported.

```
6576 \newcommand*{\@glxtr@do@titlecaps@warn}{}%
```

@initialtagging Used in \printglossary but at least v4.19 of glossaries required.

```
6577 \newcommand*{\@glxtr@activate@initialtagging}{}%
```

\@glxtr@tag Definition of tagging command when used in glossary.

```
6578 \newrobustcmd*{\@glxtr@tag}[1]{%
6579 \glsifattribute{\glscurrententrylabel}{tagging}{true}%
6580 {\glxtrtagfont{#1}}{#1}%
6581 }
```

\glxtrtagfont Used in the glossary.

```
6582 \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.

```
6583 \ifdef\@gls@preglossaryhook
6584 {
6585 \renewcommand*{\@gls@preglossaryhook}{%
6586 \@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.

```
6587 \ifundef\@glxtr@org@postdescription
6588 {%
6589 \let\@glxtr@org@postdescription\glspostdescription
6590 \renewcommand*{\glspostdescription}{%
6591 \ifglentryexists{\glscurrententrylabel}%
6592 {%
6593 \glxtrpostdescription
6594 \@glxtr@org@postdescription
```

```

6595      }%
6596      {}%
6597      }%
6598      }%
6599      {}%

```

Enable the options used by \@glsxtrp:

```

6600      \glossxtrsetpopts
6601      }%
6602  }
6603 {}

```

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.

```

6604 \newcommand*{\glsxtrpostdescription}{%
6605   \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6606 }

```

postdescgeneral

```

6607 \newcommand*{\glsxtrpostdescgeneral}{}

```

xtrpostdescterm

```

6608 \newcommand*{\glsxtrpostdescterm}{}

```

postdescacronym

```

6609 \newcommand*{\glsxtrpostdescacronym}{}

```

descabbreviation

```

6610 \newcommand*{\glsxtrpostdescabbreviation}{}

```

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.

```

6611 \newcommand*{\glsdefpostdesc}[2]{%
6612   \csdef{glsxtrpostdesc#1}{#2}%
6613 }

```

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.

```

6614 \renewcommand*{\glspostlinkhook}{%
6615   \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6616 }

```

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.

```

6617 \newcommand*{\glsxtrpostlinkhook}{%
6618   \glsxtrdiscardperiod{\glslabel}%

```

```

6619 {\glxtrpostlinkendsentence}%
6620 {\glxtrifcustomdiscardperiod
6621  {\glxtrifperiod{\glxtrpostlinkendsentence}{\glxtrpostlink}}}%
6622  {\glxtrpostlink}%
6623 }%
6624 }

```

`\omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```

6625 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}

```

`\glxtrpostlink`

```

6626 \newcommand*{\glxtrpostlink}{%
6627  \csuse{\glxtrpostlink\glscategory{\glslabel}}}%
6628 }

```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glxtrpostlink`.

```

6629 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6630  \ifthenelse{\equal{#1}{}}{%
6631    {\PackageError{glossaries-extra}
6632     {Invalid empty category label in \string\glxtrpostlink}{}}}%
6633    {\csdef{\glxtrpostlink#1}{#2}}}%
6634 }

```

`\linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```

6635 \newcommand*{\glxtrpostlinkendsentence}{%
6636  \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
6637  {%
6638    \csuse{\glxtrpostlink\glscategory{\glslabel}}}%

```

Put the full stop back.

```

6639  .\spacefactor\sfcode'\. \relax
6640 }%
6641 {%

```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```

6642  \spacefactor\sfcode'\. \relax
6643 }%
6644 }

```

`\dDescOnFirstUse` Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```

6645 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6646  \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}}%
6647 }

```

`\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.

```

6648 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6649   \glxtrifwasfirstuse
6650   {%
6651     \ifglshassymbol{\glslabel}%
6652     {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
6653     {}}%
6654   }%
6655   {}%
6656 }

```

`\DescOnFirstUse` Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```

6657 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6658   \glxtrifwasfirstuse
6659   {%
6660     \space\glxtrparen
6661     {%
6662       \ifglshassymbol{\glslabel}%
6663       {\glssaccesssymbol{\glslabel}, }%
6664       {}}%
6665     \glssaccessdesc{\glslabel}%
6666   }%
6667   }%
6668   {}%
6669 }

```

`\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).

```

6670 \newcommand*{\glxtrdiscardperiod}[3]{%
6671   \glxtrifwasfirstuse
6672   {%
6673     \glusifattribute{#1}{retainfirstuseperiod}{true}%
6674     {#3}%
6675     {%
6676       \glusifattribute{#1}{discardperiod}{true}%
6677       {%
6678         \glusifplural
6679         {%
6680           \glusifattribute{#1}{pluraldiscardperiod}{true}%
6681           {\glxtrifperiod{#2}{#3}}%
6682           {#3}%
6683         }%
6684         {%
6685           \glxtrifperiod{#2}{#3}%

```

```

6686     }%
6687     }%
6688     {#3}%
6689     }%
6690 }%
6691 {%
6692   \glsifattribute{#1}{discardperiod}{true}%
6693   {%
6694     \glsifplural
6695     {%
6696       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6697       {\glsxtrifperiod{#2}{#3}}%
6698       {#3}%
6699     }%
6700     {%
6701       \glsxtrifperiod{#2}{#3}%
6702     }%
6703   }%
6704   {#3}%
6705 }%
6706 }

```

`\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`

```
6707 \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 ' ').

```
6708 \newcommand*{\glsxtr@punclist}{.,:;?!}
```

`punctuationmark` Add character to punctuation list.

```
6709 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`punctuationmarks` Reset the punctuation list.

```
6710 \newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}
```

`\glsxtrifpunc` `\glsxtrifnextpunc{<true part>}{<false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```

6711 \newcommand*{\glsxtrifnextpunc}[2]{%
6712   \def\reserved@a{#1}%
6713   \def\reserved@b{#2}%
6714   \futurelet\@glspunc@token\glsxtr@ifnextpunc
6715 }

```


sxtr@ifnextpunc

```
6716 \newcommand*{\glsxtr@ifnextpunc}{%
6717   \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6718   \reserved@b
6719 }
```

xtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
6720 \newcommand*{\glsxtr@ifpunctoken}[1]{%
6721   \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
6722 }
```

xtr@ifpunctoken

```
6723 \def\@glsxtr@ifpunctoken#1#2{%
6724   \let\reserved@d=#2%
6725   \ifx\reserved@d\@nnil
6726     \let\glsxtr@next\@glsxtr@notfoundinlist
6727   \else
6728     \ifx#1\reserved@d
6729       \let\glsxtr@next\@glsxtr@foundinlist
6730     \else
6731       \let\glsxtr@next\@glsxtr@ifpunctoken
6732     \fi
6733   \fi
6734   \glsxtr@next#1%
6735 }
```

xtr@foundinlist

```
6736 \def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6737 \def\@glsxtr@notfoundinlist#1{\@secondoftwo}
```

glsxtrdopostpunc

```
\glsxtrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
6738 \newcommand{\glsxtrdopostpunc}[1]{%
6739   \glsxtr@ifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
6740 }
```

@glsxtr@swaptwo

```
6741 \newcommand{\@glsxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
6742 \define@key{glstrabbrv}{category}{%
6743   \edef\glscategorylabel{#1}%
6744   \ifcsdef{@glstrabbrv@current@#1}%
6745   {%
```

Warning should already have been issued.

```
6746   \let\@glstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6747   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6748   \glstr@applyabbrvstyle{\csname @glstrabbrv@current@#1\endcsname}%
6749   \let\GlsXtrWarnDeprecatedAbbrStyle\@glstr@orgwarndep
6750 }%
6751 {}%
6752 }
```

Save the short plural form. This may be needed before the entry is defined.

```
6753 \define@key{glstrabbrv}{shortplural}{%
6754   \def\@glstr@shortpl{#1}%
6755 }
```

Similarly for the long plural form.

```
6756 \define@key{glstrabbrv}{longplural}{%
6757   \def\@glstr@longpl{#1}%
6758 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

`\glsshortpltok`

```
6759 \newtoks\glsshortpltok
```

`\glslongpltok`

```
6760 \newtoks\glslongpltok
```

`\glstr@insertdots` Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to `\newabbreviation`. Note that explicitly using the short or shortplural keys will override this.

```
6761 \newcommand*{\@glstr@insertdots}[2]{%
6762   \def#1{%
6763     \@glstr@insert@dots#1#2\@nnil
6764   }
```

xtr@insert@dots

```
6765 \newcommand*{\@glsxtr@insert@dots}[2]{%
6766   \ifx\@nnil#2\relax
6767   \let\@glsxtr@insert@dots@next\@gobble
6768   \else
6769   \ifx\relax#2\relax
6770   \else
6771     \appto#1{#2.}%
6772   \fi
6773   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6774   \fi
6775   \@glsxtr@insert@dots@next#1%
6776 }
```

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be defined:

\glsxtrwordsep

```
6777 \newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

\glsxtrword

```
6778 \newcommand*{\glsxtrword}[1]{#1}
```

tr@markwordseps

```
6779 \newcommand*{\@glsxtr@markwordseps}[2]{%
6780   \def#1{%
6781     \@glsxtr@mark@wordseps#1#2 \@nnil
6782   }}
```

r@mark@wordseps

```
6783 \def\@glsxtr@mark@wordseps#1#2 #3{%
6784   \ifdefempty{#1}%
6785   {\def#1{\protect\glsxtrword{#2}}}%
6786   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
6787   \ifx\@nnil#3\relax
6788   \let\@glsxtr@mark@wordseps@next\relax
6789   \else
6790   \def\@glsxtr@mark@wordseps@next{%
6791     \@glsxtr@mark@wordseps#1#3}%
6792   \fi
6793   \@glsxtr@mark@wordseps@next
6794 }
```

newabbreviation Define a new generic abbreviation.

```
6795 \newcommand*{\newabbreviation}[4][ ]{%
6796   \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
6797 }
```

`\newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This just makes it easier to save and restore the original definition.)

```
6798 \newcommand*{\glxtr@newabbreviation}[4]{%
6799   \glstoklisttok{#1}%
6800   \glstoklabeltok{#2}%
6801   \glstokshorttok{#3}%
6802   \glstoklongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
6803   \def\glxtrorgshort{#3}%
6804   \def\glxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
6805   \def\ExtraCustomAbbreviationFields{}
```

Initialise accessibility settings if required.

```
6806   \@gls@initaccesskeys
```

Get the category.

```
6807   \def\glscategorylabel{abbreviation}%
6808   \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
6809   \setkeys*{\glxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
6810   \def\@gls@longpl{#4\glspluralsuffix}%
6811   \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
6812   \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6813   {%
6814     \@glxtr@markwordseps\@gls@long{#4}%
6815     \expandafter\def\expandafter\@gls@longpl\expandafter
6816       {\@gls@long\glspluralsuffix}%
6817     \let\@gls@default@longpl\@gls@longpl
```

Update `\glslongtok`.

```
6818     \expandafter\glslongtok\expandafter{\@gls@long}%
6819   }%
6820   {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6821   \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6822   {%
6823     \@glxtr@markwordseps\@gls@short{#3}%
6824   }%
6825   {}%
```

Has the insertdots attribute been set?

```
6826   \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6827   {%
6828     \@glxtr@insertdots\@gls@short{#3}%
```

```

6829     \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6830 }%
6831 {\def\@gls@short{#3}}%
6832 }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

6833 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6834 {%
6835     \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6836         '\abbrvpluralsuffix}%
6837 }%
6838 {%

```

Has the noshortplural attribute been set?

```

6839 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6840 {%
6841     \let\@gls@shortpl\@gls@short
6842 }%
6843 {%
6844     \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6845         '\abbrvpluralsuffix}%
6846 }%
6847 }%

```

Update \glsshorttok:

```

6848 \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

6849 \glstrnewabbrevpresetkeyhook{#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).

```

6850 \setkeys*{\glstrabbrv}[category]{#1}%

```

Has the plural been explicitly set?

```

6851 \ifx\@gls@default@longpl\@gls@longpl
6852 \else

```

Has the markwords attribute been set?

```

6853 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6854 {%
6855     \expandafter\@glstr@markwordseps\expandafter\@gls@longpl\expandafter
6856         {\@gls@longpl}%
6857 }%
6858 {}%
6859 \fi

```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```

6860 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6861 \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```

6862 \@gls@setup@default@short@access{#3}%

```

Do any extra setup provided by hook:

6863 \newabbreviationhook

Define this entry:

```
6864 \protected@edef\@do@newglossaryentry{%
6865   \noexpand\newglossaryentry{\the\glslabeltok}%
6866   {%
6867     type=\glstrabbrvtype,%
6868     category=abbreviation,%
6869     short={\the\glsshorttok},%
6870     shortplural={\the\glsshortpltok},%
6871     long={\the\glslongtok},%
6872     longplural={\the\glslongpltok},%
6873     name={\the\glsshorttok},%
6874     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

6875 \ExtraCustomAbbreviationFields

Any explicit fields set in the optional argument override all other settings.

```
6876   \the\glskeylisttok
6877 }%
6878 }%
6879 \do@newglossaryentry
6880 \GlsXtrPostNewAbbreviation
6881 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

6882 \newcommand*{\glstrnewabbrevpresetkeyhook}[3]{}

NewAbbreviation Hook used by abbreviation styles.

6883 \newcommand*{\GlsXtrPostNewAbbreviation}{}

bbreviationhook Hook for use with \newabbreviation.

6884 \newcommand*{\newabbreviationhook}{}

reviationFields

6885 \newcommand*{\CustomAbbreviationFields}{}

\glstrparen For the parenthetical styles.

6886 \newcommand*{\glstrparen}[1]{(#1)}

lsxtrfullformat Full format without case change.

```
6887 \newcommand*{\glstrfullformat}[2]{%
6888   \glsfirstlongfont{\glsaccesslong{#1}}#2\glstrfullsep{#1}%
6889   \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6890 }
```

`\lsxtrfullformat` Full format with case change.

```
6891 \newcommand*{\Glsxtrfullformat}[2]{%
6892   \glsfirstlongfont{\Glsaccesslong{#1}}#2\gsxtrfullsep{#1}%
6893   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
6894 }
```

`\xtrfullplformat` Plural full format without case change.

```
6895 \newcommand*{\Glsxtrfullplformat}[2]{%
6896   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6897   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6898 }
```

`\xtrfullplformat` Plural full format with case change.

```
6899 \newcommand*{\Glsxtrfullplformat}[2]{%
6900   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
6901   \gsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6902 }
```

`\glxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```
6903 \newcommand*{\glxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glentryfull` (for example, first use suppresses the long form or uses a footnote).

`\nlinefullformat` Full format without case change.

```
6904 \newcommand*{\Glsxtrinlinefullformat}{\glxtrfullformat}
```

`\nlinefullformat` Full format with case change.

```
6905 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\xtrfullplformat` Plural full format without case change.

```
6906 \newcommand*{\Glsxtrinlinefullplformat}{\glxtrfullplformat}
```

`\inefullplformat` Plural full format with case change.

```
6907 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glxtrfull` set of commands instead.

`\glentryfull`

```
6908 \renewcommand*{\glentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
6909 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glentryfullpl`

```
6910 \renewcommand*{\glentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

6911 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

6912 `\newcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvdefaultfont{#1}}`

`\glbbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

6913 `\newcommand*{\glbbrvdefaultfont}[1]{\glbbrvfont{#1}}`

`\glabbrvfont` Font changing command used for the abbreviation on subsequent use.

6914 `\newcommand*{\glabbrvfont}[1]{\glabbrvdefaultfont{#1}}`

`\glbbrvdefaultfont`

6915 `\newcommand*{\glbbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glxtrlong`.

6916 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glxtrlong`.

6917 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glslfirstlongfont` Font changing command used for the long form on first use or in the full format.

6918 `\newcommand*{\glslfirstlongfont}[1]{\glslongfont{#1}}`

`\glslongdefaultfont`

6919 `\newcommand*{\glslfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

6920 `\newcommand*{\glxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glbrvpluralsuffix` Default plural suffix.

6921 `\newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}`

`\glxtrfull` Full form (no case-change).

6922 `\newrobustcmd*{\glxtrfull}{\@glshyp@opt\ns@glxtrfull}`

6923 `\newcommand*\ns@glxtrfull[2][\%`

6924 `\new@ifnextchar[\@glxtr@full{#1}{#2}]{\%`

6925 `{\@glxtr@full{#1}{#2}[]}\%`

6926 `}`

`\@glxtr@full` Low-level macro:

6927 `\def\@glxtr@full#1#2[#3]{\%`

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```

6928 \@glxtr@record{#1}{#2}{glslink}%
6929 \glsdoifexists{#2}%
6930 {%
6931 \glsssetabbrvfmt{\glscategory{#2}}%
6932 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6933 \let\glslifplural\@secondoftwo
6934 \let\glscapscase\@firstofthree
6935 \let\glslinsert\@empty
6936 \def\glscustomtext{\glxtrinlinefullformat{#2}{#3}}%

```

What should \glxtrifwasfirstuse be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```

6937 \glxtrsetupfulldefs
6938 \@glsl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6939 }%
6940 \glspostlinkhook
6941 }

```

trsetupfulldefs

```

6942 \newcommand*{\glxtrsetupfulldefs}{%
6943 \let\glxtrifwasfirstuse\@firstoftwo
6944 }

```

\Glsxtrfull Full form (first letter uppercase).

```

6945 \newrobustcmd*{\Glsxtrfull}{\@glsl@hyp@opt\@ns@Glsxtrfull}
6946 \newcommand*\ns@Glsxtrfull[2][{}]{%
6947 \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
6948 {\@Glsxtr@full{#1}{#2}[]}%
6949 }

```

\@Glsxtr@full Low-level macro:

```

6950 \def\@Glsxtr@full#1#2[#3]{%
6951 \glsdoifexists{#2}%
6952 {%
6953 \glsssetabbrvfmt{\glscategory{#2}}%
6954 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6955 \let\glslifplural\@secondoftwo
6956 \let\glscapscase\@secondofthree
6957 \let\glslinsert\@empty
6958 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6959 \glxtrsetupfulldefs
6960 \@glsl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6961 }%
6962 \glspostlinkhook
6963 }

```

\GLSxtrfull Full form (all uppercase).

```
6964 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
6965 \newcommand*\ns@GLSxtrfull[2][]{\%
6966   \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}]{\%
6967     {\@GLSxtr@full{#1}{#2}[]}%
6968 }
```

\@GLSxtr@full Low-level macro:

```
6969 \def\@GLSxtr@full#1#2[#3]{\%
6970   \glsdoifexists{#2}%
6971   {\%
6972     \glssetabbrvfmt{\glscategory{#2}}%
6973     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6974     \let\glsifplural\@secondoftwo
6975     \let\glsupcase\@thirdofthree
6976     \let\glsinsert\@empty
6977     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6978     \glsxtrsetupfulldefs
6979     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6980   }%
6981   \glspostlinkhook
6982 }
```

\glsxtrfullpl Plural full form (no case-change).

```
6983 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
6984 \newcommand*\ns@glsxtrfullpl[2][]{\%
6985   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}]{\%
6986     {\@glsxtr@fullpl{#1}{#2}[]}%
6987 }
```

\@glsxtr@fullpl Low-level macro:

```
6988 \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).
6989   \@glsxtr@record{#1}{#2}{\glslink}%
6990   \glsdoifexists{#2}%
6991   {\%
6992     \glssetabbrvfmt{\glscategory{#2}}%
6993     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6994     \let\glsifplural\@firstoftwo
6995     \let\glsupcase\@firstofthree
6996     \let\glsinsert\@empty
6997     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6998     \glsxtrsetupfulldefs
6999     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7000   }%
7001   \glspostlinkhook
7002 }
```

```

\Glsxtrfullpl Plural full form (first letter uppercase).
7003 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
7004 \newcommand*\ns@Glsxtrfullpl[2][]{%
7005   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}]{%
7006     {\@Glsxtr@fullpl{#1}{#2}[]}%
7007 }

\@Glsxtr@fullpl Low-level macro:
7008 \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).
7009   \@glsxtr@record{#1}{#2}{glslink}%
7010   \glsdoifexists{#2}%
7011   {%
7012     \glssetabbrvfmt{\gls@category{#2}}%
7013     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7014     \let\glsifplural\@firstoftwo
7015     \let\gls@scapscase\@secondofthree
7016     \let\glsinsert\@empty
7017     \def\gls@customtext{\Glsxtrinlinefullplformat{#2}{#3}}%
7018     \glsxtrsetupfulldefs
7019     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7020   }%
7021   \gls@postlinkhook
7022 }

\GLSxtrfullpl Plural full form (all upper case).
7023 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
7024 \newcommand*\ns@GLSxtrfullpl[2][]{%
7025   \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}]{%
7026     {\@GLSxtr@fullpl{#1}{#2}[]}%
7027 }

\@GLSxtr@fullpl Low-level macro:
7028 \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).
7029   \@glsxtr@record{#1}{#2}{glslink}%
7030   \glsdoifexists{#2}%
7031   {%
7032     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7033     \let\glsifplural\@firstoftwo
7034     \let\gls@scapscase\@thirdofthree
7035     \let\glsinsert\@empty
7036     \def\gls@customtext{%
7037       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
7038     \glsxtrsetupfulldefs

```

```

7039 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7040 }%
7041 \glspostlinkhook
7042 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

7043 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7044 \newcommand*{\ns@glsxtrshort}[2] [] {%
7045 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2} []}%
7046 }

```

Read in the final optional argument:

```

7047 \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).

```

7048 \glsxtr@record{#1}{#2}{glslink}%
7049 \glsdoifexists{#2}%
7050 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

7051 \glssetabbrvfmt{\glscategory{#2}}%
7052 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7053 \let\glsxtrifwasfirstuse\@secondoftwo
7054 \let\glsifplural\@secondoftwo
7055 \let\glscapscase\@firstofthree
7056 \let\glsinsert\@empty
7057 \def\glscustomtext{%
7058 \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7059 \ifglsxtrininsertinside\else#3\fi
7060 }%
7061 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7062 }%
7063 \glspostlinkhook
7064 }

```

`\Glsxtrshort`

```

7065 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7066 \newcommand*{\ns@Glsxtrshort}[2] [] {%
7067 \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
7068 }

```

Read in the final optional argument:

```

7069 \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).

```

7070 \@glstr@record{#1}{#2}{glslink}%
7071 \glsoifexists{#2}%
7072 {%
7073 \glsetabbrvfmt{\glscategory{#2}}%
7074 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7075 \let\glstrifwasfirstuse\@secondoftwo
7076 \let\gl@ifplural\@secondoftwo
7077 \let\glscapscase\@secondofthree
7078 \let\glinsert\@empty
7079 \def\glscustomtext{%
7080 \glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
7081 \ifglstrinsertinside\else#3\fi
7082 }%
7083 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7084 }%
7085 \glspostlinkhook
7086 }

```

\GLSxtrshort

```

7087 \newrobustcmd*{\GLSxtrshort}{\@gl@hyp@opt\ns@GLSxtrshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7088 \newcommand*{\ns@GLSxtrshort}[2] [] {%
7089 \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} []}]%
7090 }

```

Read in the final optional argument:

```

7091 \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).

```

7092 \@glstr@record{#1}{#2}{glslink}%
7093 \glsoifexists{#2}%
7094 {%
7095 \glsetabbrvfmt{\glscategory{#2}}%
7096 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7097 \let\glstrifwasfirstuse\@secondoftwo
7098 \let\gl@ifplural\@secondoftwo
7099 \let\glscapscase\@thirdofthree
7100 \let\glinsert\@empty
7101 \def\glscustomtext{%
7102 \mfirstucMakeUppercase
7103 {\glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
7104 \ifglstrinsertinside\else#3\fi
7105 }%
7106 }%
7107 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7108 }%

```

```

7109 \glspostlinkhook
7110 }

```

\glsxtrlong

```

7111 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\ns@glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7112 \newcommand*{\ns@glsxtrlong}[2] [] {%
7113   \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2} []}%
7114 }

```

Read in the final optional argument:

```

7115 \def\@glsxtrlong#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7116   \@glxtr@record{#1}{#2}{glslink}%
7117   \glsoifexists{#2}%
7118   {%
7119     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7120     \let\glxtrifwasfirstuse\@secondoftwo
7121     \let\glsifplural\@secondoftwo
7122     \let\glscapscase\@firstofthree
7123     \let\glsinsert\@empty
7124     \def\glscustomtext{%
7125       \glslongfont{\glssaccesslong{#2}\ifglxtrininsertinside#3\fi}%
7126       \ifglxtrininsertinside\else#3\fi
7127     }%
7128     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7129   }%
7130   \glspostlinkhook
7131 }

```

\Glsxtrlong

```

7132 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7133 \newcommand*{\ns@Glsxtrlong}[2] [] {%
7134   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
7135 }

```

Read in the final optional argument:

```

7136 \def\@Glsxtrlong#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7137   \@glxtr@record{#1}{#2}{glslink}%
7138   \glsoifexists{#2}%
7139   {%
7140     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7141     \let\glxtrifwasfirstuse\@secondoftwo

```

```

7142 \let\glsifplural\@secondoftwo
7143 \let\glsifscapscase\@secondofthree
7144 \let\glsinsert\@empty
7145 \def\glscustomtext{%
7146   \glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7147   \ifglstrinsertinside\else#3\fi
7148 }%
7149 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7150 }%
7151 \glspostlinkhook
7152 }

```

\GLSxtrlong

```

7153 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7154 \newcommand*{\ns@GLSxtrlong}[2] [] {%
7155   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
7156 }

```

Read in the final optional argument:

```

7157 \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).

```

7158   \@glsxtr@record{#1}{#2}{glslink}%
7159   \glsdoifexists{#2}%
7160   {%
7161     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7162     \let\glsxtrifwasfirstuse\@secondoftwo
7163     \let\glsifplural\@secondoftwo
7164     \let\glsifscapscase\@thirdofthree
7165     \let\glsinsert\@empty
7166     \def\glscustomtext{%
7167       \mfirstucMakeUppercase
7168       {\glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7169       \ifglstrinsertinside\else#3\fi
7170     }%
7171     }%
7172     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7173   }%
7174   \glspostlinkhook
7175 }

```

Plural short forms:

\glsxtrshortpl

```

7176 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7177 \newcommand*{\ns@glsxtrshortpl}[2] [] {%

```

```

7178 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7179 }

```

Read in the final optional argument:

```

7180 \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).

```

7181 \@glsxtr@record{#1}{#2}{glslink}%
7182 \glsdoifexists{#2}%
7183 {%
7184 \glssetabbrvfmt{\glscategory{#2}}%
7185 \let\do@glsc@link@checkfirsthyper\@glsc@link@nocheckfirsthyper
7186 \let\glsxtrifwasfirstuse\@secondoftwo
7187 \let\glsifplural\@firstoftwo
7188 \let\glscapscase\@firstofthree
7189 \let\glsinsert\@empty
7190 \def\glscustomtext{%
7191 \glsabbrvfont{\glssaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7192 \ifglsxtrininsertinside\else#3\fi
7193 }%
7194 \@glsc@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7195 }%
7196 \glspostlinkhook
7197 }

```

\Glsxtrshortpl

```

7198 \newrobustcmd*{\Glsxtrshortpl}{\@glsc@hyp@opt\@ns@Glsxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7199 \newcommand*{\ns@Glsxtrshortpl}[2] []{%
7200 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7201 }

```

Read in the final optional argument:

```

7202 \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).

```

7203 \@glsxtr@record{#1}{#2}{glslink}%
7204 \glsdoifexists{#2}%
7205 {%
7206 \glssetabbrvfmt{\glscategory{#2}}%
7207 \let\do@glsc@link@checkfirsthyper\@glsc@link@nocheckfirsthyper
7208 \let\glsxtrifwasfirstuse\@secondoftwo
7209 \let\glsifplural\@firstoftwo
7210 \let\glscapscase\@secondofthree
7211 \let\glsinsert\@empty
7212 \def\glscustomtext{%
7213 \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7214 \ifglsxtrininsertinside\else#3\fi

```



```

7215 }%
7216 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7217 }%
7218 \glspostlinkhook
7219 }

```

\GLSxtrshortpl

```

7220 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7221 \newcommand*{\ns@GLSxtrshortpl}[2][]{%
7222   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
7223 }

```

Read in the final optional argument:

```

7224 \def\@GLSxtrshortpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7225   \@glsxtr@record{#1}{#2}{glslink}%
7226   \glsdoifexists{#2}%
7227   {%
7228     \glssetabbrvfmt{\glscategory{#2}}%
7229     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7230     \let\glsxtrifwasfirstuse\@secondoftwo
7231     \let\glsifplural\@firstoftwo
7232     \let\glscapscase\@thirdofthree
7233     \let\glsinsert\@empty
7234     \def\glscustomtext{%
7235       \mfirstucMakeUppercase
7236       {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7237       \ifglsxtrininsertinside\else#3\fi
7238     }%
7239   }%
7240   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7241 }%
7242 \glspostlinkhook
7243 }

```

Plural long forms:

\glsxtrlongpl

```

7244 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7245 \newcommand*{\ns@glsxtrlongpl}[2][]{%
7246   \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
7247 }

```

Read in the final optional argument:

```

7248 \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).

```

7249 \@glstr@record{#1}{#2}{glslink}%
7250 \glsoifexists{#2}%
7251 {%
7252 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7253 \let\glstrifwasfirstuse\@secondoftwo
7254 \let\gl@ifplural\@firstoftwo
7255 \let\glscapscase\@firstofthree
7256 \let\glinsert\@empty
7257 \def\glscustomtext{%
7258 \glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7259 \ifglstrinsertinside\else#3\fi
7260 }%
7261 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7262 }%
7263 \glspostlinkhook
7264 }

```

\Glsxtrlongpl

```

7265 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\ns@Glsxtrlongpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

7266 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7267 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7268 }

```

Read in the final optional argument:

```

7269 \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).

```

7270 \@glstr@record{#1}{#2}{glslink}%
7271 \glsoifexists{#2}%
7272 {%
7273 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7274 \let\glstrifwasfirstuse\@secondoftwo
7275 \let\gl@ifplural\@firstoftwo
7276 \let\glscapscase\@secondofthree
7277 \let\glinsert\@empty
7278 \def\glscustomtext{%
7279 \glslongfont{\Glsaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7280 \ifglstrinsertinside\else#3\fi
7281 }%
7282 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7283 }%
7284 \glspostlinkhook
7285 }

```

\GLSxtrlongpl

```

7286 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7287 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7288   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7289 }

    Read in the final optional argument:
7290 \def\@GLSxtrlongpl#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7291   \@glsxtr@record{#1}{#2}{glslink}%
7292   \glsdoifexists{#2}%
7293   {%
7294     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7295     \let\glsxtrifwasfirstuse\@secondoftwo
7296     \let\glsifplural\@firstoftwo
7297     \let\glsapscase\@thirdofthree
7298     \let\glsinsert\@empty
7299     \def\glscustomtext{%
7300       \mfirstucMakeUppercase
7301       {\glsfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7302       \ifglsxtrininsertinside\else#3\fi
7303     }%
7304     }%
7305     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7306     }%
7307   \glspostlinkhook
7308 }

```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

7309 \newcommand*{\glssetabbrvfmt}[1] {%
7310   \ifcsdef{\glsabbrv@current@#1}%
7311   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7312   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7313 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7314 \newrobustcmd*{\glsuseabbrvfont}[2] {{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```

7315 \newrobustcmd*{\glsuselongfont}[2] {{\glssetabbrvfmt{#2}\glsfont{#1}}}

```

sxtrgenabbrvfmt Similar to \glsacfm, but for abbreviations.

```

7316 \newcommand*{\glsxtrgenabbrvfmt} {%
7317   \ifdefempty\glscustomtext
7318   {%
7319     \ifglsused\glslabel
7320     {%

```

Subsequent use:

7321 \glsifplural
7322 {%

Subsequent plural form:

7323 \glscapscase
7324 {%

Subsequent plural form, don't adjust case:

7325 \glstrsubsequentplfmt{\glslabel}{\glsinsert}%
7326 }%
7327 {%

Subsequent plural form, make first letter upper case:

7328 \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7329 }%
7330 {%

Subsequent plural form, all caps:

7331 \mfirstucMakeUppercase
7332 {\glstrsubsequentplfmt{\glslabel}{\glsinsert}}%
7333 }%
7334 }%
7335 {%

Subsequent singular form

7336 \glscapscase
7337 {%

Subsequent singular form, don't adjust case:

7338 \glstrsubsequentfmt{\glslabel}{\glsinsert}%
7339 }%
7340 {%

Subsequent singular form, make first letter upper case:

7341 \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7342 }%
7343 {%

Subsequent singular form, all caps:

7344 \mfirstucMakeUppercase
7345 {\glstrsubsequentfmt{\glslabel}{\glsinsert}}%
7346 }%
7347 }%
7348 }%
7349 {%

First use:

7350 \glsifplural
7351 {%

First use plural form:

7352 \glscapscase
7353 {%

First use plural form, don't adjust case:

```
7354      \glxtrfullplformat{\glslabel}{\glsinsert}%  
7355      }%  
7356      {%
```

First use plural form, make first letter upper case:

```
7357      \Glsxtrfullplformat{\glslabel}{\glsinsert}%  
7358      }%  
7359      {%
```

First use plural form, all caps:

```
7360      \mfirstucMakeUppercase  
7361      {\glxtrfullplformat{\glslabel}{\glsinsert}}%  
7362      }%  
7363      }%  
7364      {%
```

First use singular form

```
7365      \glscapscase  
7366      {%
```

First use singular form, don't adjust case:

```
7367      \glxtrfullformat{\glslabel}{\glsinsert}%  
7368      }%  
7369      {%
```

First use singular form, make first letter upper case:

```
7370      \Glsxtrfullformat{\glslabel}{\glsinsert}%  
7371      }%  
7372      {%
```

First use singular form, all caps:

```
7373      \mfirstucMakeUppercase  
7374      {\glxtrfullformat{\glslabel}{\glsinsert}}%  
7375      }%  
7376      }%  
7377      }%  
7378      }%  
7379      {%
```

User supplied text.

```
7380      \glscustomtext  
7381      }%  
7382 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7383 \newcommand*{\glxtrsubsequentfmt}[2]{%  
7384   \glsabbrvfont{\glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%  
7385   \ifglxtrininsertinside \else#2\fi  
7386 }  
7387 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

```

subsequentplfmt Subsequent use format (plural no case change).
7388 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7389   \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7390   \ifglxtrinsertinside \else#2\fi
7391 }
7392 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
7393 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7394   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7395   \ifglxtrinsertinside \else#2\fi
7396 }
7397 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).
7398 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7399   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7400   \ifglxtrinsertinside \else#2\fi
7401 }
7402 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

```

breviationstyle
7403 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7404   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
7405   {%
7406     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7407   }%
7408   {%
7409     \ifcsstring{@glsabbrv@current@#1}{#2}%
7410     {%
7411       Have abbreviations already been defined for this category?
7412       \def\@glxtr@dostylewarn{}%
7413       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7414       {%
7415         \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7416           style has been switched \MessageBreak
7417           for category ‘#1’, \MessageBreak
7418           but there have already been entries \MessageBreak
7419           defined for this category. Unwanted \MessageBreak
7420           side-effects may result}}%
7421         \@endfortrue
7422       }%
7423       \@glxtr@dostylewarn
7424     }%

```

Set up the style for the given category.

```

7425      \csdef{@glsabbrv@current@#1}{#2}%
7426      \glsxtr@applyabbrvstyle{#2}%
7427      }%
7428      }%
7429 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7430 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7431   \csuse{@glsabbrv@dispstyle@setup@#1}%
7432   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7433 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

7434 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7435   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7436 }

```

`\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.

```

7437 \newcommand*{\newabbreviationstyle}[3]{%
7438   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7439   {%
7440     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7441       defined}{}%
7442   }%
7443   {%
7444     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7445     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7446     #2}%
7447     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7448     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7449     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7450     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7451     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7452     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7453     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7454     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7455     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7456     #3}%
7457   }%
7458 }

```

breivationstyle

```
7459 \newcommand*{\renewabbreviationstyle}[3]{%
7460   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7461   {%
7462     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7463   }%
7464   {%
7465     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
7466       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7467       #2}%
7468     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
7469       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7470       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7471       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7472       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7473       #3}%
7474     }%
7475 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
7476 \newcommand*{\letabbreviationstyle}[2]{%
7477   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7478   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7479 }
```

ecated@abbrstyle

`\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```
7480 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7481   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7482     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7483     \csuse{@glsabbrv@dispstyle@setup@#2}%
7484   }%
7485   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7486 }
```

ecatedAbbrStyle Generate warning for deprecated style use.

```
7487 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7488   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7489   use ‘#2’ instead}%
7490 }
```


eAbbrStyleSetup

```

7491 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7492   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7493   {%
7494     \PackageError{glossaries-extra}%
7495     {Unknown abbreviation style definitions ‘#1’}{}%
7496   }%
7497   {%
7498     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7499   }%
7500 }
```

seAbbrStyleFmts

```

7501 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7502   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7503   {%
7504     \PackageError{glossaries-extra}%
7505     {Unknown abbreviation style formats ‘#1’}{}%
7506   }%
7507   {%
7508     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7509   }%
7510 }
```

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.

```

7511 \newif\ifglsxtrinsertinside
7512 \glsxtrinsertinsidefalse
```

trlongshortname

```

7513 \newcommand*{\glsxtrlongshortname}{%
7514   \protect\glsabbrvfont{\the\glsshorttok}%
7515 }
```

long-short

```

7516 \newabbreviationstyle{long-short}%
7517 {%
```

```

7518 \renewcommand*{\CustomAbbreviationFields}{%
7519     name={\glxtrlongshortname},
7520     sort={\the\glsshorttok},
7521     first={\protect\glsfirstlongfont{\the\glslongtok}%
7522         \protect\glxtrfullsep{\the\glslabeltok}%
7523         \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7524     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7525         \protect\glxtrfullsep{\the\glslabeltok}%
7526         \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7527     plural={\protect\glsabbrvfont{\the\glsshortpltok}}},%
7528     description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7529 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7530     \glshasattribute{\the\glslabeltok}{regular}%
7531     {%
7532         \glssetattribute{\the\glslabeltok}{regular}{false}%
7533     }%
7534     {}%
7535 }%
7536}%
7537{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7538 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7539 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7540 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7541 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7542 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7543 \renewcommand*{\glxtrfullformat}[2]{%
7544     \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7545     \ifglxtrininsertinside\else##2\fi
7546     \glxtrfullsep{##1}%
7547     \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7548 }%
7549 \renewcommand*{\glxtrfullplformat}[2]{%
7550     \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7551     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7552     \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7553 }%
7554 \renewcommand*{\Glsxtrfullformat}[2]{%
7555     \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7556     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7557     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7558 }%
7559 \renewcommand*{\Glsxtrfullplformat}[2]{%
7560     \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7561     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7562 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7563 }%
7564 }

```

Set this as the default style for general abbreviations:

```

7565 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7566 \newcommand*{\glsxtrlongshortdescsort}{%
7567 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7568 }

```

ngshortdescname

```

7569 \newcommand*{\glsxtrlongshortdescname}{%
7570 \protect\glslongfont{\the\glslongtok}
7571 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7572 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7573 \newabbreviationstyle{long-short-desc}%
7574 {%
7575 \renewcommand*{\CustomAbbreviationFields}{%
7576 name={\glsxtrlongshortdescname},
7577 sort={\glsxtrlongshortdescsort},%
7578 first={\protect\glsfirstlongfont{\the\glslongtok}%
7579 \protect\glsxtrfullsep{\the\glslabeltok}%
7580 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7581 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7582 \protect\glsxtrfullsep{\the\glslabeltok}%
7583 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7584 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7585 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7586 }%

```

Unset the regular attribute if it has been set.

```

7587 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7588 \glshasattribute{\the\glslabeltok}{regular}%
7589 {%
7590 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7591 }%
7592 {}%
7593 }%
7594 }%
7595 {%
7596 \GlsXtrUseAbbrStyleFmts{long-short}%
7597 }

```

trshortlongname

```
7598 \newcommand*{\glxtrshortlongname}{%
7599   \protect\glsabbrvfont{\the\glsshorttok}%
7600 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7601 \newabbreviationstyle{short-long}%
7602 {%
7603   \renewcommand*{\CustomAbbreviationFields}{%
7604     name={\glxtrshortlongname},
7605     sort={\the\glsshorttok},
7606     description={\the\glslongtok},%
7607     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7608       \protect\glxtrfullsep{\the\glslabeltok}%
7609       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7610     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7611       \protect\glxtrfullsep{\the\glslabeltok}%
7612       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7613     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7614 }
```

Unset the regular attribute if it has been set.

```
7614 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7615   \glsattribute{\the\glslabeltok}{regular}%
7616   {%
7617     \glssattribute{\the\glslabeltok}{regular}{false}%
7618   }%
7619   {}%
7620 }%
7621 }%
7622 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7623 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7624 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7625 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7626 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7627 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7628 \renewcommand*{\glxtrfullformat}[2]{%
7629   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7630   \ifglxtrininsertinside\else##2\fi
7631   \glxtrfullsep{##1}%
7632   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7633 }%
7634 \renewcommand*{\glxtrfullplformat}[2]{%
7635   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7636   \ifglxtrininsertinside\else##2\fi
7637   \glxtrfullsep{##1}%
7638 }
```

```

7638 \glxstrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7639 }%
7640 \renewcommand*{\Glsxtrfullformat}[2]{%
7641 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7642 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7643 \glxstrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7644 }%
7645 \renewcommand*{\Glsxtrfullplformat}[2]{%
7646 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7647 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7648 \glxstrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7649 }%
7650 }

```

ortlongdescsort

```

7651 \newcommand*{\glxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7652 \newcommand*{\glxtrshortlongdescname}{%
7653 \protect\glsabbrvfont{\the\glsshorttok}
7654 \glxstrparen{\protect\glslongfont{\the\glslongtok}}}%
7655 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7656 \newabbreviationstyle{short-long-desc}%
7657 {%
7658 \renewcommand*{\CustomAbbreviationFields}{%
7659 name={\glxtrshortlongdescname},
7660 sort={\glxtrshortlongdescsort},
7661 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7662 \protect\glxtrfullsep{\the\glslabeltok}}%
7663 \glxstrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7664 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7665 \protect\glxtrfullsep{\the\glslabeltok}}%
7666 \glxstrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7667 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7668 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7669 }%

```

Unset the regular attribute if it has been set.

```

7670 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7671 \glshasattribute{\the\glslabeltok}{regular}%
7672 {%
7673 \glissetattribute{\the\glslabeltok}{regular}{false}%
7674 }%
7675 {}%
7676 }%

```

```

7677 }%
7678 {%
7679   \GlsXtrUseAbbrStyleFmts{short-long}%
7680 }

```

ongfootnotefont Only used by the “footnote” styles.

```

7681 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

ongfootnotefont Only used by the “footnote” styles.

```

7682 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

xtrabbrvfootnote `\glsxtrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

7683 \newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}

```

xtrfootnotename

```

7684 \newcommand*{\glsxtrfootnotename}{%
7685   \protect\glsabbrvfont{\the\glsshorttok}%
7686 }

```

footnote Short form followed by long form in footnote on first use.

```

7687 \newabbreviationstyle{footnote}%
7688 {%
7689   \renewcommand*{\CustomAbbreviationFields}{%
7690     name={\glsxtrfootnotename},
7691     sort={\the\glsshorttok},
7692     description={\the\glslongtok},%

7693     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7694     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7695     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7696     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7697     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7698     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

7699     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.

```

7700 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7701   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7702   \glsattribute{\the\glslabeltok}{regular}%

```

```

7703   {%
7704       \glsetattribute{\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{\glxtrabbrvpluralsuffix}%
7711 \renewcommand*\glssabrvfont[1]{\glssabrvdefaultfont{##1}}%
7712 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7713 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7714 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7715 \renewcommand*\glxtrfullformat[2]{%
7716     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7717     \ifglxtrininsertinside\else##2\fi
7718     \protect\glxtrabbrvfootnote{##1}%
7719     {\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
7720 }%
7721 \renewcommand*\glxtrfullplformat[2]{%
7722     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7723     \ifglxtrininsertinside\else##2\fi
7724     \protect\glxtrabbrvfootnote{##1}%
7725     {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7726 }%
7727 \renewcommand*\Glsxtrfullformat[2]{%
7728     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7729     \ifglxtrininsertinside\else##2\fi
7730     \protect\glxtrabbrvfootnote{##1}%
7731     {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7732 }%
7733 \renewcommand*\Glsxtrfullplformat[2]{%
7734     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7735     \ifglxtrininsertinside\else##2\fi
7736     \protect\glxtrabbrvfootnote{##1}%
7737     {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7738 }%

```

The first use full form and the inline full form use the short (long) style.

```

7739 \renewcommand*\glxtrininlinefullformat[2]{%
7740     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7741     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7742     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
7743 }%
7744 \renewcommand*\glxtrininlinefullplformat[2]{%
7745     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7746     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7747     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%

```

```

7748 }%
7749 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7750   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7751   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7752   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7753 }%
7754 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7755   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7756   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7757   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7758 }%
7759 }

```

short-footnote

```
7760 \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.

```

7761 \newabbreviationstyle{postfootnote}%
7762 {%
7763   \renewcommand*{\CustomAbbreviationFields}{%
7764     name={\glxtrfootnotename},
7765     sort={\the\glsshorttok},
7766     description={\the\glslongtok},%
7767     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7768     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7769     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.

```

7770 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7771   \csdef{glxtrpostlink\glscategorylabel}{%
7772     \glxtrifwasfirstuse
7773     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7774       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7775       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7776       }%
7777     }%
7778   }%
7779   \glshasattribute{\the\glslabeltok}{regular}%
7780   {%
7781     \glssetattribute{\the\glslabeltok}{regular}{false}%
7782   }%
7783   }%
7784 }%

```


The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7785 \renewcommand*{\glxtrsetupfulldefs}{%
7786   \let\glxtrifwasfirstuse\@secondoftwo
7787 }%
7788 }%
7789 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

7790 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7791 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
7792 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
7793 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongfootnotefont{##1}}%
7794 \renewcommand*{\glsslongfont}[1]{\glsslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```

7795 \renewcommand*{\glxtrfullformat}[2]{%
7796   \glssfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7797   \ifglxtrininsertinside\else##2\fi
7798 }%
7799 \renewcommand*{\glxtrfullplformat}[2]{%
7800   \glssfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7801   \ifglxtrininsertinside\else##2\fi
7802 }%
7803 \renewcommand*{\Glsxtrfullformat}[2]{%
7804   \glssfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7805   \ifglxtrininsertinside\else##2\fi
7806 }%
7807 \renewcommand*{\Glsxtrfullplformat}[2]{%
7808   \glssfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7809   \ifglxtrininsertinside\else##2\fi
7810 }%
```

The first use full form and the inline full form use the short (long) style.

```

7811 \renewcommand*{\glxtrininlinefullformat}[2]{%
7812   \glssfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7813   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7814   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
7815 }%
7816 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7817   \glssfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7818   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7819   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7820 }%
7821 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
7822   \glssfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7823   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7824   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
7825 }%
7826 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```

```

7827 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7828 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7829 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7830 }%
7831 }

```

rt-postfootnote

```
7832 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```

7833 \newcommand*{\glxtrshortnolongname}{%
7834 \protect\glsabbrvfont{\the\glsshorttok}%
7835 }

```

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.

```

7836 \newabbreviationstyle{short}%
7837 {%
7838 \renewcommand*{\CustomAbbreviationFields}{%
7839 name={\glxtrshortnolongname},
7840 sort={\the\glsshorttok},
7841 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7842 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7843 text={\protect\glsabbrvfont{\the\glsshorttok}},
7844 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7845 description={\the\glslongtok}}%
7846 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7847 \glssetattribute{\the\glslabeltok}{regular}{true}}%
7848 }%
7849 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7850 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7851 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7852 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7853 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7854 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7855 \renewcommand*{\glxtrinlinefullformat}[2]{%
7856 \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
7857 \ifglxtrinsertinside##2\fi}%
7858 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7859 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7860 }%
7861 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7862 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%

```

```

7863 \ifglxtrinsertinside##2\fi}%
7864 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7865 \glxtrparen{\glsfirslongfont{\glssaccesslongpl{##1}}}%
7866 }%
7867 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7868 \protect\glsfirstabbrvfont{\glssaccessshort{##1}%
7869 \ifglxtrinsertinside##2\fi}%
7870 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7871 \glxtrparen{\glsfirslongfont{\Glsaccesslong{##1}}}%
7872 }%
7873 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7874 \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}%
7875 \ifglxtrinsertinside##2\fi}%
7876 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7877 \glxtrparen{\glsfirslongfont{\Glsaccesslongpl{##1}}}%
7878 }%

```

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.

```

7879 \renewcommand*{\glxtrfullformat}[2]{%
7880 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7881 \ifglxtrinsertinside\else##2\fi
7882 }%
7883 \renewcommand*{\glxtrfullplformat}[2]{%
7884 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7885 \ifglxtrinsertinside\else##2\fi
7886 }%
7887 \renewcommand*{\Glsxtrfullformat}[2]{%
7888 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7889 \ifglxtrinsertinside\else##2\fi
7890 }%
7891 \renewcommand*{\Glsxtrfullplformat}[2]{%
7892 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7893 \ifglxtrinsertinside\else##2\fi
7894 }%
7895 }

```

Set this as the default style for acronyms:

```

7896 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7897 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7898 \newabbreviationstyle{short-nolong-noreg}%
7899 {%
7900 \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7901 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

7902 \glshasattribute{\the\glslabeltok}{regular}%
7903 {%
7904 \glissetattribute{\the\glslabeltok}{regular}{false}%
7905 }%
7906 {}%
7907 }%
7908 }%
7909 {%
7910 \GlsXtrUseAbbrStyleFmts{short-nolong}%
7911 }

```

trshortdescname

```

7912 \newcommand*{\glxtrshortdescname}{%
7913 \protect\glsabbrvfont{\the\glsshorttok}%
7914 }

```

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.

```

7915 \newabbreviationstyle{short-desc}%
7916 {%
7917 \renewcommand*{\CustomAbbreviationFields}{%
7918 name={\glxtrshortdescname},
7919 sort={\the\glsshorttok},
7920 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7921 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7922 text={\protect\glsabbrvfont{\the\glsshorttok}},
7923 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7924 description={\the\glslongtok}}%
7925 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7926 \glissetattribute{\the\glslabeltok}{regular}{true}}%
7927 }%
7928 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7929 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7930 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7931 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7932 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7933 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7934 \renewcommand*{\glxtrinlinefullformat}[2]{%
7935 \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
7936 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7937 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7938 }%
7939 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7940 \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
7941 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7942 \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%

```

```

7943 }%
7944 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7945   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7946   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7947   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7948 }%
7949 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7950   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7951   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7952   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7953 }%

```

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.

```

7954 \renewcommand*{\glxtrfullformat}[2]{%
7955   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7956   \ifglxtrinsertinside\else##2\fi
7957 }%
7958 \renewcommand*{\glxtrfullplformat}[2]{%
7959   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7960   \ifglxtrinsertinside\else##2\fi
7961 }%
7962 \renewcommand*{\Glsxtrfullformat}[2]{%
7963   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7964   \ifglxtrinsertinside\else##2\fi
7965 }%
7966 \renewcommand*{\Glsxtrfullplformat}[2]{%
7967   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7968   \ifglxtrinsertinside\else##2\fi
7969 }%
7970 }

```

ort-nolong-desc

```

7971 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7972 \newabbreviationstyle{short-nolong-desc-noreg}%
7973 {%
7974   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7975 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7976   \glshasattribute{\the\glslabeltok}{regular}%
7977   {%
7978     \glissetattribute{\the\glslabeltok}{regular}{false}%
7979   }%
7980   {}%
7981 }%
7982 }%
7983 {%

```

```

7984 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7985 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7986 \newabbreviationstyle{nolong-short}%
7987 {%
7988 \GlsXtrUseAbbrStyleSetup{short-nolong}%
7989 }%
7990 {%
7991 \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7992 \renewcommand*{\glxtrinlinefullformat}[2]{%
7993 \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7994 \ifglxtrininsertinside##2\fi}%
7995 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7996 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7997 }%
7998 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7999 \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
8000 \ifglxtrininsertinside##2\fi}%
8001 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8002 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8003 }%
8004 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8005 \protect\glsfirstlongfont{\glsaccesslong{##1}}%
8006 \ifglxtrininsertinside##2\fi}%
8007 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8008 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8009 }%
8010 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8011 \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
8012 \ifglxtrininsertinside##2\fi}%
8013 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8014 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8015 }%
8016 }

```

nolong-short-noreg Like nolong-short but doesn't set the regular attribute.

```

8017 \newabbreviationstyle{nolong-short-noreg}%
8018 {%
8019 \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

8020 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8021 \glshasattribute{\the\glslabeltok}{regular}%
8022 {%
8023 \glssetattribute{\the\glslabeltok}{regular}{false}%
8024 }%

```

```

8025     {}%
8026   }%
8027 }%
8028 {%
8029   \GlsXtrUseAbbrStyleFmts{nolong-short}%
8030 }

```

noshortdescname

```

8031 \newcommand*{\glxtrlongnoshortdescname}{%
8032   \protect\glslongfont{\the\glslongtok}%
8033 }

```

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.

```

8034 \newabbreviationstyle{long-desc}%
8035 {%
8036   \renewcommand*{\CustomAbbreviationFields}{%
8037     name={\glxtrlongnoshortdescname},
8038     sort={\the\glslongtok},
8039     first={\protect\glslongfont{\the\glslongtok}},
8040     firstplural={\protect\glslongfont{\the\glslongpltok}},
8041     text={\glslongfont{\the\glslongtok}},
8042     plural={\glslongfont{\the\glslongpltok}}}%
8043   }%
8044   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8045     \glsssetattribute{\the\glslabeltok}{regular}{true}}%
8046 }%
8047 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8048 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8049 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
8050 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
8051 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8052 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8053 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8054   \glslongfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8055   \ifglxtrininsertinside \else##2\fi
8056 }%
8057 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8058   \glslongfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8059   \ifglxtrininsertinside \else##2\fi
8060 }%
8061 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8062   \glslongfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8063   \ifglxtrininsertinside \else##2\fi
8064 }%

```

```

8065 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8066   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
8067   \ifglsxtrinsertinside \else##2\fi
8068 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8069 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8070   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8071   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8072   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8073 }%
8074 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8075   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8076   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8077   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8078 }%
8079 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8080   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8081   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8082   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8083 }%
8084 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8085   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8086   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8087   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8088 }%

```

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.

```

8089 \renewcommand*{\glsxtrfullformat}[2]{%
8090   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8091   \ifglsxtrinsertinside\else##2\fi
8092 }%
8093 \renewcommand*{\glsxtrfullplformat}[2]{%
8094   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8095   \ifglsxtrinsertinside\else##2\fi
8096 }%
8097 \renewcommand*{\Glsxtrfullformat}[2]{%
8098   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8099   \ifglsxtrinsertinside\else##2\fi
8100 }%
8101 \renewcommand*{\Glsxtrfullplformat}[2]{%
8102   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8103   \ifglsxtrinsertinside\else##2\fi
8104 }%
8105 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

8106 \letabbreviationstyle{long-noshort-desc}{long-desc}

```


`short-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```
8107 \newabbreviationstyle{long-noshort-desc-noreg}%
8108 {%
8109   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
      Unset the regular attribute if it has been set.
8110   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8111     \glshasattribute{\the\glslabeltok}{regular}%
8112     {%
8113       \glissetattribute{\the\glslabeltok}{regular}{false}%
8114     }%
8115     {}%
8116   }%
8117 }%
8118 {%
8119   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8120 }
```

`longnoshortname`

```
8121 \newcommand*{\glsxtrlongnoshortname}{%
8122   \protect\glsabbrvfont{\the\glsshorttok}%
8123 }
```

`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.

```
8124 \newabbreviationstyle{long}%
8125 {%
8126   \renewcommand*{\CustomAbbreviationFields}{%
8127     name={\glsxtrlongnoshortname},
8128     sort={\the\glsshorttok},
8129     first={\protect\glsfirstlongfont{\the\glslongtok}},
8130     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8131     text={\glslongfont{\the\glslongtok}},
8132     plural={\glslongfont{\the\glslongpltok}},%
8133     description={\the\glslongtok}%
8134   }%
8135   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8136     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8137 }%
8138 {%
8139   \GlsXtrUseAbbrStyleFmts{long-desc}%
8140 }
```

`long-noshort` Provide a synonym that matches similar styles.

```
8141 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```

8142 \newabbreviationstyle{long-noshort-noreg}%
8143 {%
8144   \GlsXtrUseAbbrStyleSetup{long-noshort}%
      Unset the regular attribute if it has been set.
8145   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8146     \glshasattribute{\the\glslabeltok}{regular}%
8147     {%
8148       \glissetattribute{\the\glslabeltok}{regular}{false}%
8149     }%
8150   }%
8151 }%
8152 }%
8153 {%
8154   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8155 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

```

\glsxtrscfont   Maintained for backward-compatibility.
8156 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}

\glsabbrvscfont Added for consistent naming.
8157 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}

sxtrfirstscfont Maintained for backward-compatibility.
8158 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}

irstabbrvscfont Added for consistent naming.
8159 \newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}

```

and for the default short form suffix:

```

\glsxtrscsuffix
8160 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}

long-short-sc
8161 \newabbreviationstyle{long-short-sc}%
8162 {%
8163   \renewcommand*{\CustomAbbreviationFields}{%
8164     name={\glsxtrlongshortname},
8165     sort={\the\glsshorttok},
8166     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8167       \protect\glsxtrfullsep{\the\glslabeltok}%
8168       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8169     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8170       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

8171 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8172 plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
8173 description={\the\glslongtok}}%
8174 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8175 \glsattribute{\the\glslabeltok}{regular}%
8176 {%
8177 \glssetattribute{\the\glslabeltok}{regular}{false}%
8178 }%
8179 {}%
8180 }%
8181 }%
8182 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8183 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8184 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8185 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

8186 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8187 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8188 \renewcommand*\glxtrfullformat[2]{%
8189 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8190 \ifglxtrinsertinside\else##2\fi
8191 \glxtrfullsep{##1}}%
8192 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8193 }%
8194 \renewcommand*\glxtrfullplformat[2]{%
8195 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8196 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8197 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8198 }%
8199 \renewcommand*\Glsxtrfullformat[2]{%
8200 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8201 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8202 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8203 }%
8204 \renewcommand*\Glsxtrfullplformat[2]{%
8205 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8206 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
8207 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8208 }%
8209 }

```

g-short-sc-desc

```

8210 \newabbreviationstyle{long-short-sc-desc}%
8211 {%
8212 \renewcommand*\CustomAbbreviationFields{%

```

```

8213     name={\glxtrlongshortdescname},
8214     sort={\glxtrlongshortdescsort},%
8215     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8216       \protect\glxtrfullsep{\the\glslabeltok}%
8217       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8218     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8219       \protect\glxtrfullsep{\the\glslabeltok}%
8220       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8221     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8222     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8223 }%

```

Unset the regular attribute if it has been set.

```

8224 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8225   \glshasattribute{\the\glslabeltok}{regular}%
8226   {%
8227     \glissetattribute{\the\glslabeltok}{regular}{false}%
8228   }%
8229   {}%
8230 }%
8231 }%
8232 {%

```

As long-short-sc style:

```

8233 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8234 }

```

Now the short (long) version

```

8235 \newabbreviationstyle{short-sc-long}%
8236 {%
8237   \renewcommand*{\CustomAbbreviationFields}{%
8238     name={\glxtrshortlongname},
8239     sort={\the\glsshorttok},
8240     description={\the\glslongtok},%
8241     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8242       \protect\glxtrfullsep{\the\glslabeltok}%
8243       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8244     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8245       \protect\glxtrfullsep{\the\glslabeltok}%
8246       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8247     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8248 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8249   \glshasattribute{\the\glslabeltok}{regular}%
8250   {%
8251     \glissetattribute{\the\glslabeltok}{regular}{false}%
8252   }%
8253   {}%
8254 }%
8255 }%

```

8256 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8257 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8258 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8259 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8260 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8261 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8262 \renewcommand*{\glxtrfullformat}[2]{%
8263   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8264   \ifglxtrininsertinside\else##2\fi
8265   \glxtrfullsep{##1}%
8266   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8267 }%
8268 \renewcommand*{\glxtrfullplformat}[2]{%
8269   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8270   \ifglxtrininsertinside\else##2\fi
8271   \glxtrfullsep{##1}%
8272   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8273 }%
8274 \renewcommand*{\Glsxtrfullformat}[2]{%
8275   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8276   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8277   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8278 }%
8279 \renewcommand*{\Glsxtrfullplformat}[2]{%
8280   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8281   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8282   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8283 }%
8284 }
```

As before but user provides description

```
8285 \newabbreviationstyle{short-sc-long-desc}%
8286 {%
8287   \renewcommand*{\CustomAbbreviationFields}{%
8288     name={\glxtrshortlongdescname},
8289     sort={\glxtrshortlongdescsort},
8290     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
8291     \protect\glxtrfullsep{\the\glslabeltok}%
8292     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8293     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
8294     \protect\glxtrfullsep{\the\glslabeltok}%
8295     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8296     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8297     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8298   }%
```

Unset the regular attribute if it has been set.

```

8299 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8300   \glshasattribute{\the\glslabeltok}{regular}}%
8301   {%
8302     \glissetattribute{\the\glslabeltok}{regular}{false}}%
8303   }%
8304   {}%
8305 }%
8306 }%
8307 {%

```

As short-sc-long style:

```

8308 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8309 }

```

short-sc

```

8310 \newabbreviationstyle{short-sc}%
8311 {%
8312   \renewcommand*{\CustomAbbreviationFields}{%
8313     name={\glxtrshortnolongname},
8314     sort={\the\glsshorttok},
8315     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8316     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8317     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8318     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8319     description={\the\glslongtok}}%
8320   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8321     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8322   }%
8323 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8324 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8325 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8326 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8327 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8328 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8329 \renewcommand*{\glxtrinlinefullformat}[2]{%
8330   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8331   \ifglxtrininsertinside##2\fi}%
8332   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8333   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8334 }%
8335 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8336   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8337   \ifglxtrininsertinside##2\fi}%
8338   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8339   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8340 }%

```

```

8341 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8342   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
8343     \ifglxtrinsertinside##2\fi}%
8344   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8345   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8346 }%
8347 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8348   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
8349     \ifglxtrinsertinside##2\fi}%
8350   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8351   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8352 }%

```

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.

```

8353 \renewcommand*{\glxtrfullformat}[2]{%
8354   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8355   \ifglxtrinsertinside\else##2\fi
8356 }%
8357 \renewcommand*{\glxtrfullplformat}[2]{%
8358   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8359   \ifglxtrinsertinside\else##2\fi
8360 }%
8361 \renewcommand*{\Glsxtrfullformat}[2]{%
8362   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8363   \ifglxtrinsertinside\else##2\fi
8364 }%
8365 \renewcommand*{\Glsxtrfullplformat}[2]{%
8366   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8367   \ifglxtrinsertinside\else##2\fi
8368 }%
8369 }

```

short-sc-nolong

```

8370 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8371 \newabbreviationstyle{short-sc-desc}%
8372 {%
8373   \renewcommand*{\CustomAbbreviationFields}{%
8374     name={\glxtrshortdesname},
8375     sort={\the\glsshorttok},
8376     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8377     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8378     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8379     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8380     description={\the\glslongtok}}%
8381   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8382     \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

8383 }%
8384 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8385 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8386 \renewcommand*\glssabbrvfont[1]{\glssabbrvscfont{##1}}%
8387 \renewcommand*\glssfirstabbrvfont[1]{\glssfirstabbrvscfont{##1}}%
8388 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
8389 \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8390 \renewcommand*\glxtrinlinefullformat[2]{%
8391   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8392   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8393   \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
8394 }%
8395 \renewcommand*\glxtrinlinefullplformat[2]{%
8396   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8397   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8398   \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8399 }%
8400 \renewcommand*\Glsxtrinlinefullformat[2]{%
8401   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8402   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8403   \glxtrparen{\glssfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8404 }%
8405 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8406   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8407   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8408   \glxtrparen{\glssfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8409 }%
```

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.

```
8410 \renewcommand*\glxtrfullformat[2]{%
8411   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8412   \ifglxtrininsertinside\else##2\fi
8413 }%
8414 \renewcommand*\glxtrfullplformat[2]{%
8415   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8416   \ifglxtrininsertinside\else##2\fi
8417 }%
8418 \renewcommand*\Glsxtrfullformat[2]{%
8419   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8420   \ifglxtrininsertinside\else##2\fi
8421 }%
8422 \renewcommand*\Glsxtrfullplformat[2]{%
8423   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8424   \ifglxtrininsertinside\else##2\fi
8425 }%
8426 }
```


-sc-nolong-desc

```
8427 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
8428 \newabbreviationstyle{nolong-short-sc}%
8429 {%
8430   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8431 }%
8432 {%
8433   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8434 \renewcommand*{\glxtrinlinefullformat}[2]{%
8435   \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
8436     \ifglxtrininsertinside##2\fi}%
8437   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8438   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8439 }%
8440 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8441   \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
8442     \ifglxtrininsertinside##2\fi}%
8443   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8444   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8445 }%
8446 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8447   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8448     \ifglxtrininsertinside##2\fi}%
8449   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8450   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8451 }%
8452 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8453   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8454     \ifglxtrininsertinside##2\fi}%
8455   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8456   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8457 }%
8458 }
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```
8459 \newabbreviationstyle{long-noshort-sc}%
8460 {%
8461   \renewcommand*{\CustomAbbreviationFields}{%
8462     name={\glxtrlongnoshortname},
8463     sort={\the\glssshorttok},
8464     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8465     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8466     text={\protect\glslongdefaultfont{\the\glslongtok}},
8467     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
```

```

8468     description={\the\glslongtok}%
8469 }%
8470 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8471   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8472 }%
8473 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8474 \renewcommand*\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8475 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8476 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8477 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8478 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8479 \renewcommand*\glsxtrsubsequentfmt}[2]{%
8480   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8481   \ifglsxtrininsertinside \else##2\fi
8482 }%
8483 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
8484   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8485   \ifglsxtrininsertinside \else##2\fi
8486 }%
8487 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8488   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8489   \ifglsxtrininsertinside \else##2\fi
8490 }%
8491 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8492   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8493   \ifglsxtrininsertinside \else##2\fi
8494 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8495 \renewcommand*\glsxtrinlinefullformat}[2]{%
8496   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8497   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8498   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8499 }%
8500 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8501   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8502   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8503   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8504 }%
8505 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8506   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8507   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8508   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8509 }%
8510 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8511   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8512   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8513 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8514 }%

```

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.

```

8515 \renewcommand*{\glxtrfullformat}[2]{%
8516 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8517 \ifglxtrininsertinside\else##2\fi
8518 }%
8519 \renewcommand*{\glxtrfullplformat}[2]{%
8520 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8521 \ifglxtrininsertinside\else##2\fi
8522 }%
8523 \renewcommand*{\Glsxtrfullformat}[2]{%
8524 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8525 \ifglxtrininsertinside\else##2\fi
8526 }%
8527 \renewcommand*{\Glsxtrfullplformat}[2]{%
8528 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8529 \ifglxtrininsertinside\else##2\fi
8530 }%
8531 }

```

long-sc Backward compatibility:

```

8532 \@glxtr@deprecated@abbrstyle{long-sc}{long-noshort-sc}

```

noshort-sc-desc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8533 \newabbreviationstyle{long-noshort-sc-desc}%
8534 {%
8535 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8536 }%
8537 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8538 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8539 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8540 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8541 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8542 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8543 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8544 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8545 \ifglxtrininsertinside \else##2\fi
8546 }%
8547 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8548 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8549 \ifglxtrininsertinside \else##2\fi
8550 }%

```

```

8551 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8552   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8553   \ifglxtrinsertinside \else##2\fi
8554 }%
8555 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8556   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8557   \ifglxtrinsertinside \else##2\fi
8558 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8559 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8560   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8561   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8562   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8563 }%
8564 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8565   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8566   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8567   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8568 }%
8569 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8570   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8571   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8572   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8573 }%
8574 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8575   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8576   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8577   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8578 }%

```

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.

```

8579 \renewcommand*{\glxtrfullformat}[2]{%
8580   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8581   \ifglxtrinsertinside\else##2\fi
8582 }%
8583 \renewcommand*{\glxtrfullplformat}[2]{%
8584   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8585   \ifglxtrinsertinside\else##2\fi
8586 }%
8587 \renewcommand*{\Glsxtrfullformat}[2]{%
8588   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8589   \ifglxtrinsertinside\else##2\fi
8590 }%
8591 \renewcommand*{\Glsxtrfullplformat}[2]{%
8592   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8593   \ifglxtrinsertinside\else##2\fi
8594 }%
8595 }

```

long-desc-sc Backward compatibility:

```
8596 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
8597 \newabbreviationstyle{short-sc-footnote}%
8598 {%
8599   \renewcommand*{\CustomAbbreviationFields}{%
8600     name={\glxtrfootnotename},
8601     sort={\the\glsshorttok},
8602     description={\the\glslongtok},%
8603     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8604       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8605       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8606     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8607       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8608       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8609     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.

```
8610 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8611   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8612   \glshasattribute{\the\glslabeltok}{regular}%
8613   {%
8614     \glsssetattribute{\the\glslabeltok}{regular}{false}%
8615   }%
8616   {%
8617 }%
8618 }%
8619 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8620 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8621 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8622 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8623 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8624 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```
8625 \renewcommand*{\glxtrfullformat}[2]{%
8626   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8627   \ifglxtrininsertinside\else##2\fi
8628   \protect\glxtrabbrvfootnote{##1}%
8629   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8630 }%
8631 \renewcommand*{\glxtrfullplformat}[2]{%
8632   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8633   \ifglxtrininsertinside\else##2\fi
8634   \protect\glxtrabbrvfootnote{##1}%
8635   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

8636 }%
8637 \renewcommand*{\Glsxtrfullformat}[2]{%
8638   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8639   \ifglxtrinsertinside\else##2\fi
8640   \protect\glxtrabbrvfootnote{##1}%
8641   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8642 }%
8643 \renewcommand*{\Glsxtrfullplformat}[2]{%
8644   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8645   \ifglxtrinsertinside\else##2\fi
8646   \protect\glxtrabbrvfootnote{##1}%
8647   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8648 }%

```

The first use full form and the inline full form use the short (long) style.

```

8649 \renewcommand*{\glxtrinlinefullformat}[2]{%
8650   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8651   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8652   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8653 }%
8654 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8655   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8656   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8657   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8658 }%
8659 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8660   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8661   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8662   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8663 }%
8664 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8665   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8666   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8667   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8668 }%
8669 }

```

footnote-sc Backward compatibility:

```

8670 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8671 \newabbreviationstyle{short-sc-postfootnote}%
8672 {%
8673   \renewcommand*{\CustomAbbreviationFields}{%
8674     name={\glxtrfootnotename},
8675     sort={\the\glsshorttok},
8676     description={\the\glslongtok},%
8677     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8678     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8679     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
8680 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8681 \csdef{glxstrpostlink\glscategorylabel}{%
8682 \glxtrifwasfirstuse
8683 {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
8684 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8685 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
8686 }%
8687 {}%
8688 }%
8689 \glshasattribute{\the\glslabeltok}{regular}%
8690 {%
8691 \glissetattribute{\the\glslabeltok}{regular}{false}%
8692 }%
8693 {}%
8694 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8695 \renewcommand*{\glxtrsetupfulldefs}{%
8696 \let\glxtrifwasfirstuse\@secondoftwo
8697 }%
8698 }%
8699 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8700 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8701 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8702 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8703 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8704 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8705 \renewcommand*{\glxtrfullformat}[2]{%
8706 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8707 \ifglxtrininsertinside\else##2\fi
8708 }%
8709 \renewcommand*{\glxtrfullplformat}[2]{%
8710 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8711 \ifglxtrininsertinside\else##2\fi
8712 }%
8713 \renewcommand*{\Glsxtrfullformat}[2]{%
8714 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8715 \ifglxtrininsertinside\else##2\fi
8716 }%
8717 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

8718 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8719 \ifglxtrinsertinside\else##2\fi
8720 }%

```

The first use full form and the inline full form use the short (long) style.

```

8721 \renewcommand*{\glxtrinlinefullformat}[2]{%
8722 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8723 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8724 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8725 }%
8726 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8727 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8728 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8729 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8730 }%
8731 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8732 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8733 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8734 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8735 }%
8736 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8737 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8738 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8739 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8740 }%
8741 }

```

postfootnote-sc Backward compatibility:

```

8742 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

```

\glxtrsmfont Maintained for backward compatibility.
8743 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
8744 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8745 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
8746 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

```

and for the default short form suffix:

\glxtrsmsuffix

```
8747 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8748 \newabbreviationstyle{long-short-sm}%
8749 {%
8750   \renewcommand*{\CustomAbbreviationFields}{%
8751     name={\glxtrlongshortname},
8752     sort={\the\glsshorttok},
8753     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8754       \protect\glxtrfullsep{\the\glslabeltok}%
8755       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8756     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8757       \protect\glxtrfullsep{\the\glslabeltok}%
8758       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8759     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8760     description={\the\glslongtok}}%
8761   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8762     \glshasattribute{\the\glslabeltok}{regular}%
8763     {%
8764       \glissetattribute{\the\glslabeltok}{regular}{false}%
8765     }%
8766   }%
8767 }%
8768 }%
8769 {%
8770   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8771   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8772   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8773 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8774 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8775 \renewcommand*{\glxtrfullformat}[2]{%
8776   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8777   \ifglxtrininsertinside\else##2\fi
8778   \glxtrfullsep{##1}%
8779   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8780 }%
8781 \renewcommand*{\glxtrfullplformat}[2]{%
8782   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8783   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8784   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8785 }%
8786 \renewcommand*{\Glsxtrfullformat}[2]{%
8787   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8788   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8789   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%

```

```

8790 }%
8791 \renewcommand*{\Glsxtrfullplformat}[2]{%
8792   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8793   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8794   \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8795 }%
8796 }

```

g-short-sm-desc

```

8797 \newabbreviationstyle{long-short-sm-desc}%
8798 {%
8799   \renewcommand*{\CustomAbbreviationFields}{%
8800     name={\glxtrlongshortdescname},
8801     sort={\glxtrlongshortdescsort},%
8802     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8803       \protect\glxtrfullsep{\the\glslabeltok}%
8804       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8805     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8806       \protect\glxtrfullsep{\the\glslabeltok}%
8807       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8808     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8809     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8810   }%

```

Unset the regular attribute if it has been set.

```

8811 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8812   \glshasattribute{\the\glslabeltok}{regular}%
8813   {%
8814     \glissetattribute{\the\glslabeltok}{regular}{false}%
8815   }%
8816   {}%
8817 }%
8818 }%
8819 {%

```

As long-short-sm style:

```

8820 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8821 }

```

short-sm-long Now the short (long) version

```

8822 \newabbreviationstyle{short-sm-long}%
8823 {%
8824   \renewcommand*{\CustomAbbreviationFields}{%
8825     name={\glxtrshortlongname},
8826     sort={\the\glsshorttok},
8827     description={\the\glslongtok},%
8828     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8829       \protect\glxtrfullsep{\the\glslabeltok}%
8830       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8831     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%

```

```

8832 \protect\glxtrfullsep{\the\glslabeltok}%
8833 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8834 plural={\protect\glssabrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8835 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8836 \glshasattribute{\the\glslabeltok}{regular}%
8837 {%
8838 \glissetattribute{\the\glslabeltok}{regular}{false}%
8839 }%
8840 {}%
8841 }%
8842 }%
8843 {%
8844 \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
8845 \renewcommand*\glsfirstabrvfont[1]{\glsfirstabrvsmfont{##1}}%
8846 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8847 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8848 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8849 \renewcommand*{\glxtrfullformat}[2]{%
8850 \glsfirstabrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8851 \ifglxtrininsertinside\else##2\fi
8852 \glxtrfullsep{##1}%
8853 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8854 }%
8855 \renewcommand*{\glxtrfullplformat}[2]{%
8856 \glsfirstabrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8857 \ifglxtrininsertinside\else##2\fi
8858 \glxtrfullsep{##1}%
8859 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8860 }%
8861 \renewcommand*{\Glsxtrfullformat}[2]{%
8862 \glsfirstabrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8863 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8864 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8865 }%
8866 \renewcommand*{\Glsxtrfullplformat}[2]{%
8867 \glsfirstabrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8868 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8869 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8870 }%
8871 }

```

rt-sm-long-desc As before but user provides description

```

8872 \newabbreviationstyle{short-sm-long-desc}%
8873 {%
8874 \renewcommand*{\CustomAbbreviationFields}{%
8875 name={\glxtrshortlongdescname},

```

```

8876 sort={\glxtrshortlongdescsort},
8877 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8878 \protect\glxtrfullsep{\the\glslabeltok}%
8879 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8880 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8881 \protect\glxtrfullsep{\the\glslabeltok}%
8882 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8883 text={\protect\glssabbrvsmfont{\the\glsshorttok}},%
8884 plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%
8885 }%

```

Unset the regular attribute if it has been set.

```

8886 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8887 \glshasattribute{\the\glslabeltok}{regular}%
8888 {%
8889 \glissetattribute{\the\glslabeltok}{regular}{false}%
8890 }%
8891 {}%
8892 }%
8893 }%
8894 {%

```

As short-sm-long style:

```

8895 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8896 }

```

short-sm

```

8897 \newabbreviationstyle{short-sm}%
8898 {%
8899 \renewcommand*{\CustomAbbreviationFields}{%
8900 name={\glxtrshortnolongname},
8901 sort={\the\glsshorttok},
8902 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8903 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8904 text={\protect\glssabbrvsmfont{\the\glsshorttok}},
8905 plural={\protect\glssabbrvsmfont{\the\glsshortpltok}},
8906 description={\the\glslongtok}}%
8907 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8908 \glissetattribute{\the\glslabeltok}{regular}{true}}%
8909 }%
8910 {%
8911 \renewcommand*\glssabbrvfont[1]{\glssabbrvsmfont{##1}}%
8912 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8913 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8914 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8915 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8916 \renewcommand*{\glxtrinlinefullformat}[2]{%
8917 \protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}%

```

```

8918     \ifglxtrinsertinside##2\fi}%
8919     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8920     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8921 }%
8922 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8923     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
8924         \ifglxtrinsertinside##2\fi}%
8925     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8926     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8927 }%

8928 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8929     \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8930         \ifglxtrinsertinside##2\fi}%
8931     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8932     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8933 }%
8934 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8935     \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8936         \ifglxtrinsertinside##2\fi}%
8937     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8938     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8939 }%

```

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.

```

8940 \renewcommand*{\glxtrfullformat}[2]{%
8941     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8942     \ifglxtrinsertinside\else##2\fi
8943 }%
8944 \renewcommand*{\glxtrfullplformat}[2]{%
8945     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8946     \ifglxtrinsertinside\else##2\fi
8947 }%
8948 \renewcommand*{\Glsxtrfullformat}[2]{%
8949     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8950     \ifglxtrinsertinside\else##2\fi
8951 }%
8952 \renewcommand*{\Glsxtrfullplformat}[2]{%
8953     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8954     \ifglxtrinsertinside\else##2\fi
8955 }%
8956 }

```

short-sm-nolong

```
8957 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8958 \newabbreviationstyle{short-sm-desc}%

```

```

8959 {%
8960   \renewcommand*{\CustomAbbreviationFields}{%
8961     name={\glxtrshortdescname},
8962     sort={\the\glsshorttok},
8963     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8964     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8965     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8966     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8967     description={\the\glslongtok}}%
8968   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8969     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8970 }%
8971 {%
8972   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8973   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8974   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8975   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8976   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8977   \renewcommand*{\glxtrinlinefullformat}[2]{%
8978     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8979     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8980   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8981 }%
8982   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8983     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8984     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8985   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8986 }%
8987   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8988     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8989     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8990   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8991 }%
8992   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8993     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8994     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8995   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8996 }%

```

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.

```

8997   \renewcommand*{\glxtrfullformat}[2]{%
8998     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8999     \ifglxtrininsertinside\else##2\fi
9000 }%
9001   \renewcommand*{\glxtrfullplformat}[2]{%
9002     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9003     \ifglxtrininsertinside\else##2\fi

```

```

9004 }%
9005 \renewcommand*{\Glsxtrfullformat}[2]{%
9006   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9007   \ifglsxtrininsertinside\else##2\fi
9008 }%
9009 \renewcommand*{\Glsxtrfullplformat}[2]{%
9010   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9011   \ifglsxtrininsertinside\else##2\fi
9012 }%
9013 }

```

-sm-nolong-desc

```

9014 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

9015 \newabbreviationstyle{nolong-short-sm}%
9016 {%
9017   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9018 }%
9019 {%
9020   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9021 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9022   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9023     \ifglsxtrininsertinside##2\fi}%
9024   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9025   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9026 }%
9027 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9028   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9029     \ifglsxtrininsertinside##2\fi}%
9030   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9031   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9032 }%
9033 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9034   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9035     \ifglsxtrininsertinside##2\fi}%
9036   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9037   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9038 }%
9039 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9040   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9041     \ifglsxtrininsertinside##2\fi}%
9042   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9043   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9044 }%
9045 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9046 \newabbreviationstyle{long-noshort-sm}%
9047 {%
9048   \renewcommand*{\CustomAbbreviationFields}{%
9049     name={\glsxtrlongnoshortname},
9050     sort={\the\glsshorttok},
9051     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9052     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9053     text={\protect\glslongdefaultfont{\the\glslongtok}},
9054     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9055     description={\the\glslongtok}%
9056   }%
9057   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9058     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9059 }%
9060 {%
9061   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
9062   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9063   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
9064   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9065   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9066 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9067   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9068   \ifglxtrininsertinside \else##2\fi
9069 }%
9070 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9071   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9072   \ifglxtrininsertinside \else##2\fi
9073 }%
9074 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9075   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9076   \ifglxtrininsertinside \else##2\fi
9077 }%
9078 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9079   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9080   \ifglxtrininsertinside \else##2\fi
9081 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9082 \renewcommand*{\glxtrinlinefullformat}[2]{%
9083   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9084   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9085   \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
9086 }%
9087 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9088   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9089   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```



```

9090 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9091 }%
9092 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9093 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9094 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9095 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9096 }%
9097 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9098 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9099 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9100 \glxstrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9101 }%

```

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.

```

9102 \renewcommand*{\glxtrfullformat}[2]{%
9103 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9104 \ifglxtrinsertinside\else##2\fi
9105 }%
9106 \renewcommand*{\glxtrfullplformat}[2]{%
9107 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9108 \ifglxtrinsertinside\else##2\fi
9109 }%
9110 \renewcommand*{\Glsxtrfullformat}[2]{%
9111 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9112 \ifglxtrinsertinside\else##2\fi
9113 }%
9114 \renewcommand*{\Glsxtrfullplformat}[2]{%
9115 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9116 \ifglxtrinsertinside\else##2\fi
9117 }%
9118 }

```

long-sm Backward compatibility:

```

9119 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9120 \newabbreviationstyle{long-noshort-sm-desc}%
9121 {%
9122 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9123 }%
9124 {%
9125 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9126 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9127 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9128 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9129 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9130 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9131   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9132   \ifglxtrinsertinside \else##2\fi
9133 }%
9134 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9135   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9136   \ifglxtrinsertinside \else##2\fi
9137 }%
9138 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9139   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9140   \ifglxtrinsertinside \else##2\fi
9141 }%
9142 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9143   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9144   \ifglxtrinsertinside \else##2\fi
9145 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9146 \renewcommand*{\glxtrinlinefullformat}[2]{%
9147   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9148   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9149   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9150 }%
9151 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9152   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9153   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9154   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9155 }%
9156 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9157   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9158   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9159   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9160 }%
9161 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9162   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9163   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9164   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9165 }%

```

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.

```

9166 \renewcommand*{\glxtrfullformat}[2]{%
9167   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9168   \ifglxtrinsertinside\else##2\fi
9169 }%
9170 \renewcommand*{\glxtrfullplformat}[2]{%
9171   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9172   \ifglxtrinsertinside\else##2\fi
9173 }%
9174 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9175 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9176 \ifglxtrinsertinside\else##2\fi
9177 }%
9178 \renewcommand*{\Glsxtrfullplformat}[2]{%
9179 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9180 \ifglxtrinsertinside\else##2\fi
9181 }%
9182 }

```

long-desc-sm Backward compatibility:

```

9183 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9184 \newabbreviationstyle{short-sm-footnote}%
9185 {%
9186 \renewcommand*{\CustomAbbreviationFields}{%
9187 name={\glxtrfootnotename},
9188 sort={\the\glsshorttok},
9189 description={\the\glslongtok},%
9190 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9191 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9192 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9193 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9194 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9195 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9196 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.

```

9197 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9198 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9199 \glsattribute{\the\glslabeltok}{regular}%
9200 {%
9201 \glssetattribute{\the\glslabeltok}{regular}{false}%
9202 }%
9203 {}%
9204 }%
9205 }%
9206 {%
9207 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9208 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9209 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
9210 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9211 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9212 \renewcommand*{\glxtrfullformat}[2]{%
9213 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9214 \ifglxtrinsertinside\else##2\fi
9215 \protect\glxtrabbrvfootnote{##1}%

```

```

9216     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9217 }%
9218 \renewcommand*{\glxtrfullplformat}[2]{%
9219     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9220     \ifglxtrininsertinside\else##2\fi
9221     \protect\glxtrabbrvfootnote{##1}%
9222     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9223 }%
9224 \renewcommand*{\Glsxtrfullformat}[2]{%
9225     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9226     \ifglxtrininsertinside\else##2\fi
9227     \protect\glxtrabbrvfootnote{##1}%
9228     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9229 }%
9230 \renewcommand*{\Glsxtrfullplformat}[2]{%
9231     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9232     \ifglxtrininsertinside\else##2\fi
9233     \protect\glxtrabbrvfootnote{##1}%
9234     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9235 }%

```

The first use full form and the inline full form use the short (long) style.

```

9236 \renewcommand*{\glxtrininlinefullformat}[2]{%
9237     \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9238     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9239     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9240 }%
9241 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9242     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9243     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9244     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9245 }%
9246 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9247     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9248     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9249     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9250 }%
9251 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9252     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9253     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9254     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9255 }%
9256 }

```

footnote-sm Backward compatibility:

```

9257 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

9258 \newabbreviationstyle{short-sm-postfootnote}%
9259 {}

```

```

9260 \renewcommand*{\CustomAbbreviationFields}{%
9261   name={\glxtrfootnotename},
9262   sort={\the\glsshorttok},
9263   description={\the\glslongtok},%
9264   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9265   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9266   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.

```

9267 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9268   \csdef{glxtrpostlink\glscategorylabel}{%
9269     \glxtrifwasfirstuse
9270     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9271       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9272         {\glsfirstlongfootnotefont{\glsenentrylong{\glslabel}}}}%
9273     }%
9274   }%
9275 }%
9276 \glshasattribute{\the\glslabeltok}{regular}%
9277 {%
9278   \glissetattribute{\the\glslabeltok}{regular}{false}%
9279 }%
9280 }%
9281 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9282 \renewcommand*{\glxtrsetupfulldefs}{%
9283   \let\glxtrifwasfirstuse\@secondoftwo
9284 }%
9285 }%
9286 {%
9287 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9288 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9289 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9290 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9291 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9292 \renewcommand*{\glxtrfullformat}[2]{%
9293   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9294   \ifglxtrininsertinside\else##2\fi
9295 }%
9296 \renewcommand*{\glxtrfullplformat}[2]{%
9297   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9298   \ifglxtrininsertinside\else##2\fi
9299 }%

```

```

9300 \renewcommand*{\Glsxtrfullformat}[2]{%
9301   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9302   \ifglxtrinsertinside\else##2\fi
9303 }%
9304 \renewcommand*{\Glsxtrfullplformat}[2]{%
9305   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9306   \ifglxtrinsertinside\else##2\fi
9307 }%

```

The first use full form and the inline full form use the short (long) style.

```

9308 \renewcommand*{\glxtrinlinefullformat}[2]{%
9309   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9310   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9311   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9312 }%
9313 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9314   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9315   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9316   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9317 }%
9318 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9319   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9320   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9321   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9322 }%
9323 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9324   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9325   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9326   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9327 }%
9328 }

```

postfootnote-sm Backward compatibility:

```

9329 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9330 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

9331 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

9332 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`firstlongemfont` Only used by the “long-em” styles.

```
9333 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
9334 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9335 \newabbreviationstyle{long-short-em}%
9336 {%
9337   \renewcommand*{\CustomAbbreviationFields}{%
9338     name={\glsxtrlongshortname},
9339     sort={\the\glsshorttok},
9340     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
9341       \protect\glsxtrfullsep{\the\glslabeltok}%
9342       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9343     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
9344       \protect\glsxtrfullsep{\the\glslabeltok}%
9345       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9346     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9347     description={\the\glslongtok}}%
9348   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9349     \glshasattribute{\the\glslabeltok}{regular}}%
9350   {%
9351     \glissetattribute{\the\glslabeltok}{regular}{false}%
9352   }%
9353   {}%
9354 }%
9355 }%
9356 {%
9357   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9358   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9359   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%

```

Use the default long fonts.

```
9360 \renewcommand*{\glfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9361 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9362 \renewcommand*{\glsxtrfullformat}[2]{%
9363   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9364   \ifglsxtrininsertinside\else##2\fi
9365   \glsxtrfullsep{##1}%
9366   \glsxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9367 }%
9368 \renewcommand*{\glsxtrfullplformat}[2]{%
9369   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9370   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9371   \glsxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9372 }%
9373 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9374 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9375 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9376 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9377 }%
9378 \renewcommand*{\Glsxtrfullplformat}[2]{%
9379 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9380 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9381 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9382 }%
9383 }

```

g-short-em-desc

```

9384 \newabbreviationstyle{long-short-em-desc}%
9385 {%
9386 \renewcommand*{\CustomAbbreviationFields}{%
9387 name={\glxtrlongshortdescname},
9388 sort={\glxtrlongshortdescsort},%
9389 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9390 \protect\glxtrfullsep{\the\glslabeltok}%
9391 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9392 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9393 \protect\glxtrfullsep{\the\glslabeltok}%
9394 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9395 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9396 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9397 }%

```

Unset the regular attribute if it has been set.

```

9398 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9399 \glshasattribute{\the\glslabeltok}{regular}%
9400 {%
9401 \glissetattribute{\the\glslabeltok}{regular}{false}%
9402 }%
9403 {}%
9404 }%
9405 }%
9406 {%

```

As long-short-em style:

```

9407 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9408 }

```

ong-em-short-em

```

9409 \newabbreviationstyle{long-em-short-em}%
9410 {%
9411 \glslongemfont is used in the description since \glsdesc doesn't set the style.
9412 \renewcommand*{\CustomAbbreviationFields}{%
9413 name={\glxtrlongshortname},
9414 sort={\the\glsshorttok},

```



```

9414 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9415 \protect\glsxtrfullsep{\the\glslabeltok}%
9416 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9417 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9418 \protect\glsxtrfullsep{\the\glslabeltok}%
9419 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9420 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9421 description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9422 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9423 \glsattribute{\the\glslabeltok}{regular}%
9424 {%
9425 \glssetattribute{\the\glslabeltok}{regular}{false}%
9426 }%
9427 {}%
9428 }%
9429 }%
9430 {%
9431 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9432 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9433 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9434 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9435 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9436 \renewcommand*{\glsxtrfullformat}[2]{%
9437 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9438 \ifglsxtrininsertinside\else##2\fi
9439 \glsxtrfullsep{##1}%
9440 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9441 }%
9442 \renewcommand*{\glsxtrfullplformat}[2]{%
9443 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9444 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9445 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9446 }%
9447 \renewcommand*{\Glsxtrfullformat}[2]{%
9448 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9449 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9450 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9451 }%
9452 \renewcommand*{\Glsxtrfullplformat}[2]{%
9453 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9454 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9455 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9456 }%
9457 }

```

m-short-em-desc

```
9458 \newabbreviationstyle{long-em-short-em-desc}%
9459 {%
9460   \renewcommand*{\CustomAbbreviationFields}{%
9461     name={\glxtrlongshortdescname},
9462     sort={\glxtrlongshortdescsort},%
9463     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9464       \protect\glxtrfullsep{\the\glslabeltok}%
9465       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9466     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9467       \protect\glxtrfullsep{\the\glslabeltok}%
9468       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9469     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9470     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}
9471   }%
```

Unset the regular attribute if it has been set.

```
9472   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9473     \glshasattribute{\the\glslabeltok}{regular}%
9474     {%
9475       \glissetattribute{\the\glslabeltok}{regular}{false}%
9476     }%
9477   }%
9478 }%
9479 }%
9480 {%
9481   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9482 }
```

short-em-long Now the short (long) version

```
9483 \newabbreviationstyle{short-em-long}%
9484 {%
9485   \renewcommand*{\CustomAbbreviationFields}{%
9486     name={\glxtrshortlongname},
9487     sort={\the\glsshorttok},
9488     description={\the\glslongtok},%
9489     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9490       \protect\glxtrfullsep{\the\glslabeltok}%
9491       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9492     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9493       \protect\glxtrfullsep{\the\glslabeltok}%
9494       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9495     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}
9496   }%
```

Unset the regular attribute if it has been set.

```
9496   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9497     \glshasattribute{\the\glslabeltok}{regular}%
9498     {%
9499       \glissetattribute{\the\glslabeltok}{regular}{false}%
9500     }%
```

```

9501    {}%
9502  }%
9503 }%
9504 {%

```

Mostly as short-long style:

```

9505 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9506 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9507 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9508 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9509 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9510 \renewcommand*{\glssxtrfullformat}[2]{%
9511   \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9512   \ifglssxtrininsertinside\else##2\fi
9513   \glssxtrfullsep{##1}%
9514   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9515 }%
9516 \renewcommand*{\glssxtrfullplformat}[2]{%
9517   \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9518   \ifglssxtrininsertinside\else##2\fi
9519   \glssxtrfullsep{##1}%
9520   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9521 }%
9522 \renewcommand*{\Glsxtrfullformat}[2]{%
9523   \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9524   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9525   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9526 }%
9527 \renewcommand*{\Glsxtrfullplformat}[2]{%
9528   \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9529   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9530   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9531 }%
9532 }

```

rt-em-long-desc As before but user provides description

```

9533 \newabbreviationstyle{short-em-long-desc}%
9534 {%
9535   \renewcommand*{\CustomAbbreviationFields}{%
9536     name={\glssxtrshortlongdescname},
9537     sort={\glssxtrshortlongdescsort},
9538     first={\protect\glssfirstabbrvemfont{\the\glssshorttok}}%
9539     \protect\glssxtrfullsep{\the\glsslabeltok}}%
9540     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongtok}}},%
9541     firstplural={\protect\glssfirstabbrvemfont{\the\glssshortpltok}}%
9542     \protect\glssxtrfullsep{\the\glsslabeltok}}%
9543     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongpltok}}},%
9544     text={\protect\glssabbrvemfont{\the\glssshorttok}},%

```

```

9545     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9546 }%

```

Unset the regular attribute if it has been set.

```

9547 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9548   \glshasattribute{\the\glslabeltok}{regular}%
9549   {%
9550     \glssetattribute{\the\glslabeltok}{regular}{false}%
9551   }%
9552   {}%
9553 }%
9554 }%
9555 {%
9556   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9557 }

```

hort-em-long-em

```

9558 \newabbreviationstyle{short-em-long-em}%
9559 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9560 \renewcommand*{\CustomAbbreviationFields}{%
9561   name={\glsxtrshortlongname},
9562   sort={\the\glsshorttok},
9563   description={\protect\glslongemfont{\the\glslongtok}},%
9564   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9565   \protect\glsxtrfullsep{\the\glslabeltok}%
9566   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9567   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9568   \protect\glsxtrfullsep{\the\glslabeltok}%
9569   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9570   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9571 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9572   \glshasattribute{\the\glslabeltok}{regular}%
9573   {%
9574     \glssetattribute{\the\glslabeltok}{regular}{false}%
9575   }%
9576   {}%
9577 }%
9578 }%
9579 {%
9580 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9581 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9582 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9583 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9584 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

9585 \renewcommand*{\glxtrfullformat}[2]{%
9586   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9587   \ifglxtrinsertinside\else##2\fi
9588   \glxtrfullsep{##1}%
9589   \glxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9590 }%
9591 \renewcommand*{\glxtrfullplformat}[2]{%
9592   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9593   \ifglxtrinsertinside\else##2\fi
9594   \glxtrfullsep{##1}%
9595   \glxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9596 }%
9597 \renewcommand*{\Glsxtrfullformat}[2]{%
9598   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9599   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9600   \glxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9601 }%
9602 \renewcommand*{\Glsxtrfullplformat}[2]{%
9603   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9604   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9605   \glxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9606 }%
9607 }

```

em-long-em-desc

```

9608 \newabbreviationstyle{short-em-long-em-desc}%
9609 {%
9610   \renewcommand*{\CustomAbbreviationFields}{%
9611     name={\glxtrshortlongdescname},%
9612     sort={\glxtrshortlongdescsort},%
9613     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9614     \protect\glxtrfullsep{\the\glslabeltok}}%
9615     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9616     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9617     \protect\glxtrfullsep{\the\glslabeltok}}%
9618     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9619     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9620     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9621 }%

```

Unset the regular attribute if it has been set.

```

9622 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9623   \glshasattribute{\the\glslabeltok}{regular}%
9624   {%
9625     \glissetattribute{\the\glslabeltok}{regular}{false}%
9626   }%
9627   {}%
9628 }%

```

```

9629 }%
9630 {%
9631   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9632 }

```

short-em

```

9633 \newabbreviationstyle{short-em}%
9634 {%
9635   \renewcommand*{\CustomAbbreviationFields}{%
9636     name={\glstrshortnolongname},
9637     sort={\the\glsshorttok},
9638     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9639     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9640     text={\protect\glabbrvemfont{\the\glsshorttok}},
9641     plural={\protect\glabbrvemfont{\the\glsshortpltok}},
9642     description={\the\glslongtok}}%
9643   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9644     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9645 }%
9646 {%
9647   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9648   \renewcommand*{\glabbrvfnt}[1]{\glabbrvemfont{##1}}%
9649   \renewcommand*{\glfirstabbrvfnt}[1]{\glfirstabbrvemfont{##1}}%
9650   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9651   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9652   \renewcommand*{\glxtrinlinefullformat}[2]{%
9653     \protect\glfirstabbrvemfont{\glaccessshort{##1}}%
9654     \ifglxtrininsertinside##2\fi}%
9655     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9656     \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9657   }%
9658   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9659     \protect\glfirstabbrvemfont{\glaccessshortpl{##1}}%
9660     \ifglxtrininsertinside##2\fi}%
9661     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9662     \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9663   }%

9664   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9665     \protect\glfirstabbrvemfont{\Glsaccessshort{##1}}%
9666     \ifglxtrininsertinside##2\fi}%
9667     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9668     \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9669   }%
9670   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9671     \protect\glfirstabbrvemfont{\Glsaccessshortpl{##1}}%
9672     \ifglxtrininsertinside##2\fi}%
9673     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9674 \glstrparen{\glstrfirstlongdefaultfont{\glstraccesslongpl{##1}}}%
9675 }%

```

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.

```

9676 \renewcommand*{\glstrfullformat}[2]{%
9677   \glstrfirstabbrvemfont{\glstraccessshort{##1}\ifglstrinsertinside##2\fi}%
9678   \ifglstrinsertinside\else##2\fi
9679 }%
9680 \renewcommand*{\glstrfullplformat}[2]{%
9681   \glstrfirstabbrvemfont{\glstraccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9682   \ifglstrinsertinside\else##2\fi
9683 }%
9684 \renewcommand*{\Glsxtrfullformat}[2]{%
9685   \glstrfirstabbrvemfont{\glstraccessshort{##1}\ifglstrinsertinside##2\fi}%
9686   \ifglstrinsertinside\else##2\fi
9687 }%
9688 \renewcommand*{\Glsxtrfullplformat}[2]{%
9689   \glstrfirstabbrvemfont{\glstraccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9690   \ifglstrinsertinside\else##2\fi
9691 }%
9692 }

```

short-em-nolong

```

9693 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9694 \newabbreviationstyle{short-em-desc}%
9695 {%
9696   \renewcommand*{\CustomAbbreviationFields}{%
9697     name={\glstrshortdescname},
9698     sort={\the\glsshorttok},
9699     first={\protect\glstrfirstabbrvemfont{\the\glsshorttok}},
9700     firstplural={\protect\glstrfirstabbrvemfont{\the\glsshortpltok}},
9701     text={\protect\glstrabbrvemfont{\the\glsshorttok}},
9702     plural={\protect\glstrabbrvemfont{\the\glsshortpltok}},
9703     description={\the\glslongtok}}%
9704   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9705     \glstrsetattribute{\the\glslabeltok}{regular}{true}}%
9706 }%
9707 {%
9708   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrxtremsuffix}%
9709   \renewcommand*{\glstrabbrvfont}[1]{\glstrabbrvemfont{##1}}%
9710   \renewcommand*{\glstrfirstabbrvfont}[1]{\glstrfirstabbrvemfont{##1}}%
9711   \renewcommand*{\glstrfirstlongfont}[1]{\glstrfirstlongdefaultfont{##1}}%
9712   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9713 \renewcommand*{\glstrinlinefullformat}[2]{%
9714   \glstrfirstabbrvemfont{\glstraccessshort{##1}\ifglstrinsertinside##2\fi}%

```

```

9715     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9716     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9717 }%
9718 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9719     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9720     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9721     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9722 }%
9723 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9724     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9725     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9726     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9727 }%
9728 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9729     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9730     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9731     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9732 }%

```

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.

```

9733 \renewcommand*{\glxtrfullformat}[2]{%
9734     \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9735     \ifglxtrinsertinside\else##2\fi
9736 }%
9737 \renewcommand*{\glxtrfullplformat}[2]{%
9738     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9739     \ifglxtrinsertinside\else##2\fi
9740 }%
9741 \renewcommand*{\Glsxtrfullformat}[2]{%
9742     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9743     \ifglxtrinsertinside\else##2\fi
9744 }%
9745 \renewcommand*{\Glsxtrfullplformat}[2]{%
9746     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9747     \ifglxtrinsertinside\else##2\fi
9748 }%
9749 }

```

-em-nolong-desc

```

9750 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9751 \newabbreviationstyle{nolong-short-em}%
9752 {%
9753     \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9754 }%
9755 {%
9756     \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```


The inline full form displays the long form followed by the short form in parentheses.

```

9757 \renewcommand*{\glxtrinlinefullformat}[2]{%
9758   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9759     \ifglxtrinsertinside##2\fi}%
9760   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9761   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9762 }%
9763 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9764   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9765     \ifglxtrinsertinside##2\fi}%
9766   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9767   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9768 }%
9769 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9770   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9771     \ifglxtrinsertinside##2\fi}%
9772   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9773   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9774 }%
9775 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9776   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9777     \ifglxtrinsertinside##2\fi}%
9778   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9779   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9780 }%
9781 }

```

`long-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```

9782 \newabbreviationstyle{long-noshort-em}%
9783 {%
9784   \renewcommand*{\CustomAbbreviationFields}{%
9785     name={\glxtrlongnoshortname},
9786     sort={\the\glsshorttok},
9787     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9788     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9789     text={\protect\glslongdefaultfont{\the\glslongtok}},
9790     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9791     description={\the\glslongtok}%
9792   }%
9793   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9794     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9795 }%
9796 {%
9797   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9798   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9799   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9800   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9801   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9802 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9803   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9804   \ifglxtrininsertinside \else##2\fi
9805 }%
9806 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9807   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9808   \ifglxtrininsertinside \else##2\fi
9809 }%
9810 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9811   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9812   \ifglxtrininsertinside \else##2\fi
9813 }%
9814 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9815   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9816   \ifglxtrininsertinside \else##2\fi
9817 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9818 \renewcommand*{\glxtrinlinefullformat}[2]{%
9819   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9820   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9821   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9822 }%
9823 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9824   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9825   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9826   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9827 }%
9828 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9829   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9830   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9831   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9832 }%
9833 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9834   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9835   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9836   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9837 }%

```

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.

```

9838 \renewcommand*{\glxtrfullformat}[2]{%
9839   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9840   \ifglxtrininsertinside\else##2\fi
9841 }%
9842 \renewcommand*{\glxtrfullplformat}[2]{%
9843   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9844   \ifglxtrininsertinside\else##2\fi
9845 }%
9846 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9847 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9848 \ifglxtrinsertinside\else##2\fi
9849 }%
9850 \renewcommand*{\Glsxtrfullplformat}[2]{%
9851 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9852 \ifglxtrinsertinside\else##2\fi
9853 }%
9854 }

```

long-em Backward compatibility:

```

9855 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9856 \newabbreviationstyle{long-em-noshort-em}%
9857 {%
9858 \renewcommand*{\CustomAbbreviationFields}{%
9859 name={\glxtrlongnoshortname},
9860 sort={\the\glsshorttok},
9861 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9862 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9863 text={\protect\glslongemfont{\the\glslongtok}},
9864 plural={\protect\glslongemfont{\the\glslongpltok}},%
9865 description={\protect\glslongemfont{\the\glslongtok}}%
9866 }%
9867 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9868 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9869 }%
9870 {%
9871 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9872 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9873 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9874 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9875 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9876 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9877 \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9878 \ifglxtrinsertinside \else##2\fi
9879 }%
9880 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9881 \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9882 \ifglxtrinsertinside \else##2\fi
9883 }%
9884 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9885 \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9886 \ifglxtrinsertinside \else##2\fi
9887 }%
9888 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9889 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9890 \ifglxtrinsertinside \else##2\fi

```

9891 }%

The inline full form displays the long format followed by the short form in parentheses.

```
9892 \renewcommand*{\glxstrinlinefullformat}[2]{%
9893   \glsfirslongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9894   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9895   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9896 }%
9897 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9898   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9899   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9900   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9901 }%
9902 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9903   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9904   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9905   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9906 }%
9907 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9908   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9909   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9910   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9911 }%
```

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.

```
9912 \renewcommand*{\glxstrfullformat}[2]{%
9913   \glsfirslongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9914   \ifglxstrinsertinside\else##2\fi
9915 }%
9916 \renewcommand*{\glxstrfullplformat}[2]{%
9917   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9918   \ifglxstrinsertinside\else##2\fi
9919 }%
9920 \renewcommand*{\Glsxtrfullformat}[2]{%
9921   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9922   \ifglxstrinsertinside\else##2\fi
9923 }%
9924 \renewcommand*{\Glsxtrfullplformat}[2]{%
9925   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9926   \ifglxstrinsertinside\else##2\fi
9927 }%
9928 }
```

oshort-em-noreg Like long-em-noshort-em but doesn't set the regular attribute.

```
9929 \newabbreviationstyle{long-em-noshort-em-noreg}%
9930 {%
9931   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```

9932 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9933   \glshasattribute{\the\glslabeltok}{regular}}%
9934   {%
9935     \glissetattribute{\the\glslabeltok}{regular}{false}}%
9936   }%
9937   {}%
9938 }%
9939 }%
9940 {%
9941   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}}%
9942 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9943 \newabbreviationstyle{long-noshort-em-desc}%
9944 {%
9945   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}}%
9946 }%
9947 {%
9948   \renewcommand*{\abbrvpluralsuffix}{\protect\glstremsuffix}%
9949   \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9950   \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
9951   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9952   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9953 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9954   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9955   \ifglsxtrininsertinside \else##2\fi
9956 }%
9957 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9958   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9959   \ifglsxtrininsertinside \else##2\fi
9960 }%
9961 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9962   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9963   \ifglsxtrininsertinside \else##2\fi
9964 }%
9965 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9966   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9967   \ifglsxtrininsertinside \else##2\fi
9968 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9969 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9970   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9971   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9972   \glsxtrparen{\protect\glsfirstabrvemfont{\glssaccessshort{##1}}}%
9973 }%
9974 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

```

```

9975 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9976 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9977 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9978 }%
9979 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9980 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9981 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9982 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9983 }%
9984 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9985 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9986 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9987 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9988 }%

```

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.

```

9989 \renewcommand*{\glxtrfullformat}[2]{%
9990 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9991 \ifglxtrinsertinside\else##2\fi
9992 }%
9993 \renewcommand*{\glxtrfullplformat}[2]{%
9994 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9995 \ifglxtrinsertinside\else##2\fi
9996 }%
9997 \renewcommand*{\Glsxtrfullformat}[2]{%
9998 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9999 \ifglxtrinsertinside\else##2\fi
10000 }%
10001 \renewcommand*{\Glsxtrfullplformat}[2]{%
10002 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10003 \ifglxtrinsertinside\else##2\fi
10004 }%
10005 }

```

long-desc-em Backward compatibility:

```
10006 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```

10007 \newabbreviationstyle{long-em-noshort-em-desc}%
10008 {%
10009 \renewcommand*{\CustomAbbreviationFields}{%
10010 name={\glxtrlongnoshortdescname},
10011 sort={\the\glslongtok},
10012 first={\protect\glsfirstlongemfont{\the\glslongtok}},
10013 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10014 text={\glslongemfont{\the\glslongtok}},
10015 plural={\glslongemfont{\the\glslongpltok}}}%

```

```

10016 }%
10017 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10018   \glssetattribute{\the\glslabeltok}{regular}{true}}%
10019 }%
10020 {%
10021   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
10022   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10023   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10024   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10025   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10026 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10027   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10028   \ifglsxtrininsertinside \else##2\fi
10029 }%
10030 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10031   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10032   \ifglsxtrininsertinside \else##2\fi
10033 }%
10034 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10035   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10036   \ifglsxtrininsertinside \else##2\fi
10037 }%
10038 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10039   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10040   \ifglsxtrininsertinside \else##2\fi
10041 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10042 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10043   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10044   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10045   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10046 }%
10047 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10048   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10049   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10050   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10051 }%
10052 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10053   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10054   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10055   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10056 }%
10057 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10058   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10059   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10060   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10061 }%

```

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.

```

10062 \renewcommand*{\glxtrfullformat}[2]{%
10063   \glsfirslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10064   \ifglxtrinsertinside\else##2\fi
10065 }%
10066 \renewcommand*{\glxtrfullplformat}[2]{%
10067   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10068   \ifglxtrinsertinside\else##2\fi
10069 }%
10070 \renewcommand*{\Glsxtrfullformat}[2]{%
10071   \glsfirslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10072   \ifglxtrinsertinside\else##2\fi
10073 }%
10074 \renewcommand*{\Glsxtrfullplformat}[2]{%
10075   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10076   \ifglxtrinsertinside\else##2\fi
10077 }%
10078 }

```

`t-em-desc-noreg` Like `long-em-noshort-em-desc` but doesn't set the regular attribute.

```

10079 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10080 {%
10081   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

10082 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10083   \glshasattribute{\the\glslabeltok}{regular}%
10084   {%
10085     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10086   }%
10087   {}%
10088 }%
10089 }%
10090 {%
10091   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10092 }

```

`ort-em-footnote`

```

10093 \newabbreviationstyle{short-em-footnote}%
10094 {%
10095   \renewcommand*{\CustomAbbreviationFields}{%
10096     name={\glxtrfootnotename},
10097     sort={\the\glsshorttok},
10098     description={\the\glslongtok},%
10099     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10100       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10101       {\protect\glsfirslongfootnotefont{\the\glslongtok}}},%
10102     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%

```



```

10103 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10104 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10105 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.

```

10106 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10107 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
10108 \glsattribute{\the\glslabeltok}{regular}%
10109 {%
10110 \glssetattribute{\the\glslabeltok}{regular}{false}%
10111 }%
10112 {}%
10113 }%
10114 }%
10115 {%
10116 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
10117 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10118 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10119 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
10120 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10121 \renewcommand*{\glxtrfullformat}[2]{%
10122 \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
10123 \ifglxtrininsertinside\else##2\fi
10124 \protect\glxtrabbrvfootnote{##1}%
10125 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10126 }%
10127 \renewcommand*{\glxtrfullplformat}[2]{%
10128 \glsfirstabbrvemfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
10129 \ifglxtrininsertinside\else##2\fi
10130 \protect\glxtrabbrvfootnote{##1}%
10131 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10132 }%
10133 \renewcommand*{\Glsxtrfullformat}[2]{%
10134 \glsfirstabbrvemfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
10135 \ifglxtrininsertinside\else##2\fi
10136 \protect\glxtrabbrvfootnote{##1}%
10137 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10138 }%
10139 \renewcommand*{\Glsxtrfullplformat}[2]{%
10140 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
10141 \ifglxtrininsertinside\else##2\fi
10142 \protect\glxtrabbrvfootnote{##1}%
10143 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10144 }%

```

The first use full form and the inline full form use the short (long) style.

```

10145 \renewcommand*{\glxtrininlinefullformat}[2]{%
10146 \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%

```

```

10147     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10148     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10149 }%
10150 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10151     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10152     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10153     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10154 }%
10155 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10156     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10157     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10158     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10159 }%
10160 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10161     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10162     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10163     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10164 }%
10165 }

```

footnote-em Backward compatibility:

```

10166 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

10167 \newabbreviationstyle{short-em-postfootnote}%
10168 {%
10169     \renewcommand*{\CustomAbbreviationFields}{%
10170         name={\glxtrfootnotename},
10171         sort={\the\glssshorttok},
10172         description={\the\glslongtok},%
10173         first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},%
10174         firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},%
10175         plural={\protect\glssabbrvemfont{\the\glssshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

10176 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10177     \csdef{glxtrpostlink\glscategorylabel}{%
10178         \glxtrifwasfirstuse
10179         {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

10180         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10181             {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
10182         }%
10183     }%
10184 }%
10185 \glshasattribute{\the\glslabeltok}{regular}%
10186 {%

```

```

10187 \glsetattribute{\the\glslabeltok}{regular}{false}%
10188 }%
10189 {}%
10190 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

10191 \renewcommand*\glxtrsetupfulldefs{%
10192 \let\glxtrifwasfirstuse\@secondoftwo
10193 }%
10194 }%
10195 {%
10196 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
10197 \renewcommand*\glabbrvfont[1]{\glabbrvemfont{##1}}%
10198 \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvemfont{##1}}%
10199 \renewcommand*\glfirstlongfont[1]{\glfirstlongfootnotefont{##1}}%
10200 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

10201 \renewcommand*\glxtrfullformat[2]{%
10202 \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10203 \ifglxtrininsertinside\else##2\fi
10204 }%
10205 \renewcommand*\glxtrfullplformat[2]{%
10206 \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10207 \ifglxtrininsertinside\else##2\fi
10208 }%
10209 \renewcommand*\Glsxtrfullformat[2]{%
10210 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10211 \ifglxtrininsertinside\else##2\fi
10212 }%
10213 \renewcommand*\Glsxtrfullplformat[2]{%
10214 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10215 \ifglxtrininsertinside\else##2\fi
10216 }%

```

The first use full form and the inline full form use the short (long) style.

```

10217 \renewcommand*\glxtrininlinefullformat[2]{%
10218 \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10219 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10220 \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
10221 }%
10222 \renewcommand*\glxtrininlinefullplformat[2]{%
10223 \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10224 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10225 \glxtrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
10226 }%
10227 \renewcommand*\Glsxtrininlinefullformat[2]{%
10228 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10229 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10230 \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%

```

```

10231 }%
10232 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10233   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10234   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10235   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10236 }%
10237 }

```

postfootnote-em Backward compatibility:

```

10238 \@glxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}

```

1.7.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glxtruserfield Default is the useri field.

```

10239 \newcommand*{\glxtruserfield}{useri}

```

glxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If \glscurrentfieldvalue has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```

10240 \ifdef\glscurrentfieldvalue
10241 {
10242   \newcommand*{\glxtruserparen}[2]{%
10243     \glsxtrfullsep{#2}%
10244     \glsxtrparen
10245       {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}}}%
10246   }
10247 }
10248 {
10249   \newcommand*{\glxtruserparen}[2]{%
10250     \glsxtrfullsep{#2}%
10251     \glsxtrparen
10252       {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}}}%
10253   }
10254 }

```

Font used for short form:

lsabbrvuserfont

```

10255 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

10256 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
10257 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10258 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10259 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10260 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10261 \newabbreviationstyle{long-short-user}%
10262 {%
10263   \renewcommand*{\CustomAbbreviationFields}{%
10264     name={\glsxtrlongshortname},
10265     sort={\the\glsshorttok},
10266     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10267       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10268     {\the\glslabeltok}},%
10269     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10270       \protect\glsxtruserparen
10271       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10272     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10273     description={\protect\glsuserdescription{\the\glslongtok}%
10274       {\the\glslabeltok}}}%
10274 }
```

Unset the regular attribute if it has been set.

```
10275 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10276   \glshasattribute{\the\glslabeltok}{regular}%
10277   {%
10278     \glissetattribute{\the\glslabeltok}{regular}{false}%
10279   }%
10280   {}%
10281 }%
10282 }%
10283 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10284 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10285 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10286 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10287 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10288 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline form are the same for this style.

```

10289 \renewcommand*{\glxtrfullformat}[2]{%
10290   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10291   \ifglxtrinsertinside\else##2\fi
10292   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10293 }%
10294 \renewcommand*{\glxtrfullplformat}[2]{%
10295   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10296   \ifglxtrinsertinside\else##2\fi
10297   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10298 }%
10299 \renewcommand*{\Glsxtrfullformat}[2]{%
10300   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10301   \ifglxtrinsertinside\else##2\fi
10302   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10303 }%
10304 \renewcommand*{\Glsxtrfullplformat}[2]{%
10305   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10306   \ifglxtrinsertinside\else##2\fi
10307   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10308 }%
10309 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10310 \newabbreviationstyle{long-postshort-user}%
10311 {%
10312   \renewcommand*{\CustomAbbreviationFields}{%
10313     name={\glxtrlongshortname},
10314     sort={\the\glsshorttok},
10315     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10316     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10317     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10318     description={\protect\glsuserdescription{\the\glslongtok}%
10319       {\the\glslabeltok}}}%
10320 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10321   \csdef{glxtrpostlink\glscategorylabel}{%
10322     \glxtrifwasfirstuse
10323     {%
10324       \glxtruserparen
10325       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10326       {\glslabel}%
10327     }%
10328     {}%
10329   }%
10330   \glsasattribute{\the\glslabeltok}{regular}%
10331   {%
10332     \glssetattribute{\the\glslabeltok}{regular}{false}%
10333   }%

```

```

10334    {}%
10335  }%
10336 }%
10337 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10338 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10339 \renewcommand*{\glsabrvfont}[1]{\glsabrvuserfont{##1}}%
10340 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10341 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10342 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10343 \renewcommand*{\glxtrfullformat}[2]{%
10344   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10345   \ifglxtrininsertinside\else##2\fi
10346 }%
10347 \renewcommand*{\glxtrfullplformat}[2]{%
10348   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10349   \ifglxtrininsertinside\else##2\fi
10350 }%
10351 \renewcommand*{\Glsxtrfullformat}[2]{%
10352   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10353   \ifglxtrininsertinside\else##2\fi
10354 }%
10355 \renewcommand*{\Glsxtrfullplformat}[2]{%
10356   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10357   \ifglxtrininsertinside\else##2\fi
10358 }%

```

In-line format:

```

10359 \renewcommand*{\glxtrinlinefullformat}[2]{%
10360   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10361   \ifglxtrininsertinside\else##2\fi
10362   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10363 }%
10364 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10365   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10366   \ifglxtrininsertinside\else##2\fi
10367   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10368 }%
10369 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10370   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10371   \ifglxtrininsertinside\else##2\fi
10372   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10373 }%
10374 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10375   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10376   \ifglxtrininsertinside\else##2\fi
10377   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%

```

```

10378 }%
10379 }

```

ortuserdescname

```

10380 \newcommand*{\glxtrlongshortuserdescname}{%
10381   \protect\glslonguserfont{\the\glslongtok}%
10382   \protect\glxtruserparen
10383   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10384 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10385 \newabbreviationstyle{long-postshort-user-desc}%
10386 {%
10387   \renewcommand*{\CustomAbbreviationFields}{%
10388     name={\glxtrlongshortuserdescname},
10389     sort={\the\glslongtok},
10390     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10391     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10392     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10393     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10394 }%
10395 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10396   \csdef{glxtrpostlink\glscategorylabel}{%
10397     \glxtrifwasfirstuse
10398     {%
10399       \glxtruserparen
10400       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10401       {\glslabel}%
10402     }%
10403   }%
10404 }%
10405 \glshasattribute{\the\glslabeltok}{regular}%
10406 {%
10407   \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10408 }%
10409 {}%
10410 }%
10411 }%
10412 {%
10413   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10414 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10415 \newabbreviationstyle{short-postlong-user}%
10416 {%
10417   \renewcommand*{\CustomAbbreviationFields}{%
10418     name={\glxtrshortlongname},
10419     sort={\the\glsshorttok},

```



```

10420 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10421 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10422 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10423 description={\protect\glsuserdescription{\the\glslongtok}%
10424 {\the\glslabeltok}}}%
10425 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10426 \csdef{glsxtrpostlink\glscategorylabel}{%
10427 \glsxtrifwasfirstuse
10428 {%
10429 \glsxtruserparen
10430 {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10431 {\glslabel}}%
10432 }%
10433 {}%
10434 }%
10435 \glsattribute{\the\glslabeltok}{regular}%
10436 {%
10437 \glssetattribute{\the\glslabeltok}{regular}{false}%
10438 }%
10439 {}%
10440 }%
10441 }%
10442 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10443 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10444 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10445 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10446 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10447 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10448 \renewcommand*{\glsxtrfullformat}[2]{%
10449 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10450 \ifglsxtrininsertinside\else##2\fi
10451 }%
10452 \renewcommand*{\glsxtrfullplformat}[2]{%
10453 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10454 \ifglsxtrininsertinside\else##2\fi
10455 }%
10456 \renewcommand*{\Glsxtrfullformat}[2]{%
10457 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10458 \ifglsxtrininsertinside\else##2\fi
10459 }%
10460 \renewcommand*{\Glsxtrfullplformat}[2]{%
10461 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10462 \ifglsxtrininsertinside\else##2\fi
10463 }%

```

In-line format:

```

10464 \renewcommand*{\glxtrinlinefullformat}[2]{%
10465   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10466   \ifglxtrininsertinside\else##2\fi
10467   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10468 }%
10469 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10470   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10471   \ifglxtrininsertinside\else##2\fi
10472   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10473 }%
10474 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10475   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10476   \ifglxtrininsertinside\else##2\fi
10477   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10478 }%
10479 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10480   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10481   \ifglxtrininsertinside\else##2\fi
10482   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10483 }%
10484 }

```

onguserdesname

```

10485 \newcommand*{\glxtrshortlonguserdesname}{%
10486   \protect\glssabbrvuserfont{\the\glssshorttok}%
10487   \protect\glxtruserparen
10488     {\protect\glslonguserfont{\the\glslongpltok}}}%
10489     {\the\glslabelltok}%
10490 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10491 \newabbreviationstyle{short-postlong-user-desc}%
10492 {%
10493   \renewcommand*{\CustomAbbreviationFields}{%
10494     name={\glxtrshortlonguserdesname},
10495     sort={\the\glssshorttok},
10496     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10497     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10498     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10499     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10500 }%
10501 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10502   \csdef{glxtrpostlink\glscategorylabel}{%
10503     \glxtrifwasfirstuse
10504     {%
10505       \glxtruserparen
10506         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10507         {\glslabel}%

```

```

10508     }%
10509     {}%
10510     }%
10511     \glshasattribute{\the\glslabeltok}{regular}%
10512     {%
10513         \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10514     }%
10515     {}%
10516 }%
10517 }%
10518 {%
10519     \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10520 }

```

short-user-desc

```

10521 \newabbreviationstyle{long-short-user-desc}%
10522 {%
10523     \renewcommand*{\CustomAbbreviationFields}{%
10524         name={\glsxtrlongshortuserdescname},
10525         sort={\glsxtrlongshortdescsort},%

10526         first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10527             \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10528             {\the\glslabeltok}},%
10529         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10530             \protect\glsxtruserparen
10531             {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10532         text={\protect\glsabbrvfont{\the\glsshorttok}},%
10533         plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10534     }%

```

Unset the regular attribute if it has been set.

```

10535     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10536         \glshasattribute{\the\glslabeltok}{regular}%
10537         {%
10538             \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10539         }%
10540         {}%
10541     }%
10542 }%
10543 {%
10544     \GlsXtrUseAbbrStyleFmts{long-short-user}%
10545 }

```

short-long-user

```

10546 \newabbreviationstyle{short-long-user}%
10547 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10548 \renewcommand*{\CustomAbbreviationFields}{%
10549     name={\glxtrshortlongname},
10550     sort={\the\glsshorttok},
10551     description={\protect\gluserdescription{\the\glslongtok}%
10552         {\the\glslabeltok}},%
10553     first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10554         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}%
10555         {\the\glslabeltok}},%
10556     firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10557         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}%
10558         {\the\glslabeltok}},%
10559     plural={\protect\glabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10560 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10561     \glshasattribute{\the\glslabeltok}{regular}%
10562     {%
10563         \glissetattribute{\the\glslabeltok}{regular}{false}%
10564     }%
10565     {}%
10566 }%
10567 }%
10568 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10569 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10570 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
10571 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvuserfont{##1}}%
10572 \renewcommand*{\glfirstlongfont}[1]{\glfirstlonguserfont{##1}}%
10573 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10574 \renewcommand*{\glxtrfullformat}[2]{%
10575     \glfirstabbrvuserfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10576     \ifglxtrininsertinside\else##2\fi
10577     \glxtruserparen{\glfirstlonguserfont{\glaccesslong{##1}}}{##1}%
10578 }%
10579 \renewcommand*{\glxtrfullplformat}[2]{%
10580     \glfirstabbrvuserfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10581     \ifglxtrininsertinside\else##2\fi
10582     \glxtruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10583 }%
10584 \renewcommand*{\Glsxtrfullformat}[2]{%
10585     \glfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10586     \ifglxtrininsertinside\else##2\fi
10587     \glxtruserparen{\glfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10588 }%
10589 \renewcommand*{\Glsxtrfullplformat}[2]{%
10590     \glfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10591     \ifglxtrininsertinside\else##2\fi

```

```

10592 \glstruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10593 }%
10594 }

```

-long-user-desc

```

10595 \newabbreviationstyle{short-long-user-desc}%
10596 {%
10597 \renewcommand*{\CustomAbbreviationFields}{%
10598   name={\glstrshortlonguserdescname},
10599   sort={\glstrshortlongdescsort},%

10600   first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10601     \protect\glstruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}}%
10602     {\the\glslabeltok}},%
10603   firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10604     \protect\glstruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}}%
10605     {\the\glslabeltok}},%
10606   text={\protect\glabbrvfont{\the\glsshorttok}},%
10607   plural={\protect\glabbrvfont{\the\glsshortpltok}}}%
10608 }%

```

Unset the regular attribute if it has been set.

```

10609 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10610 \glshasattribute{\the\glslabeltok}{regular}%
10611 {%
10612 \glissetattribute{\the\glslabeltok}{regular}{false}%
10613 }%
10614 {}%
10615 }%
10616 }%
10617 {%
10618 \GlsXtrUseAbbrStyleFmts{short-long-user}%
10619 }

```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glstrlong` set `\glinsert` to empty with the entire link-text stored in `\glscustomtext`.

`\trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glinsert` so check for that and expand.

```

10620 \newrobustcmd*{\glstrifhyphenstart}[3]{%
10621 \ifx\glinsert#1\relax
10622 \expandafter@\glstrifhyphenstart#1\relax\relax
10623 \@end@\glstrifhyphenstart{#2}{#3}%
10624 \else
10625 \@glstrifhyphenstart#1\relax\relax@\end@\glstrifhyphenstart{#2}{#3}%

```

```
10626 \fi
10627 }
```

trifhyphenstart

```
10628 \def\@glxtrifhyphenstart#1#2\end@glxtrifhyphenstart#3#4{%
10629 \ifx-#1\relax#3\else #4\fi
10630 }
```

rlonghyphenshort

```
\glxtrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10631 \newcommand*\glxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10632 {%
```

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.

```
10633 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
10634 \glsfirslonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10635 \ifglxtrininsertinside\else{#4}\fi
10636 \glxtrfullsep{#1}%
10637 \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10638 \ifglxtrininsertinside\else{#4}\fi}%
10639 }%
10640 }
```

abbrvhyphenfont

```
10641 \newcommand*\glsabbrvhyphenfont{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10642 \newcommand*\glsfirstabbrvhyphenfont{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10643 \newcommand*\glslonghyphenfont{\glslongdefaultfont}%
```

tlonghyphenfont

```
10644 \newcommand*\glsfirslonghyphenfont{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10645 \newcommand*\glxtrhyphensuffix{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the keywords attribute.

```
10646 \newabbreviationstyle{long-hyphen-short-hyphen}%
10647 {%
10648   \renewcommand*{\CustomAbbreviationFields}{%
10649     name={\glxtrlongshortname},
10650     sort={\the\glsshorttok},
10651     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10652       \protect\glxtrfullsep{\the\glslabeltok}%
10653       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10654     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10655       \protect\glxtrfullsep{\the\glslabeltok}%
10656       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10657     plural={\protect\glsshortvhyphenfont{\the\glsshortpltok}}},%
10658     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10659 }
```

Unset the regular attribute if it has been set.

```
10659 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10660   \glshasattribute{\the\glslabeltok}{regular}%
10661   {%
10662     \glissetattribute{\the\glslabeltok}{regular}{false}%
10663   }%
10664 }%
10665 }%
10666 }%
10667 {%
10668   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10669   \renewcommand*{\glsshortfont}[1]{\glsshortvhyphenfont{##1}}%
10670   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10671   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10672   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
10673 }
```

The first use full form and the inline full form are the same for this style.

```
10673 \renewcommand*{\glxtrfullformat}[2]{%
10674   \glxtrlonghyphenshort{##1}{\glssuccesslong{##1}}{\glssuccessshort{##1}}{##2}%
10675 }%
10676 \renewcommand*{\glxtrfullplformat}[2]{%
10677   \glxtrlonghyphenshort{##1}{\glssuccesslongpl{##1}}%
10678   {\glssuccessshortpl{##1}}{##2}%
10679 }%
10680 \renewcommand*{\Glsxtrfullformat}[2]{%
10681   \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\Glsaccessshort{##1}}{##2}%
10682 }%
10683 \renewcommand*{\Glsxtrfullplformat}[2]{%
10684   \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10685   {\Glsaccessshortpl{##1}}{##2}%
10686 }%
10687 }
```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```
10688 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10689 }
```

```

10689 {%
10690   \renewcommand*{\CustomAbbreviationFields}{%
10691     name={\glxtrlongshortdescname},
10692     sort={\glxtrlongshortdescsort},
10693     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10694       \protect\glxtrfullsep{\the\glslabeltok}%
10695       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10696     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10697       \protect\glxtrfullsep{\the\glslabeltok}%
10698       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10699     text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
10700     plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
10701   }%

```

Unset the regular attribute if it has been set.

```

10702   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10703     \glshasattribute{\the\glslabeltok}{regular}%
10704     {%
10705       \glissetattribute{\the\glslabeltok}{regular}{false}%
10706     }%
10707   }%
10708 }%
10709 }%
10710 {%
10711   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10712 }

```

onghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```

10713 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10714 {%

```

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.

```

10715   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10716     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10717     \ifglxtrininsertinsideelse{#3}\fi
10718   }%
10719 }

```

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.


```

10720 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10721 {%
10722   \renewcommand*{\CustomAbbreviationFields}{%
10723     name={\glxtrlongnoshortdescname},
10724     sort={\expandonce\glxtrorglong},
10725     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10726     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10727     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10728   }%

```

Unset the regular attribute if it has been set.

```

10729 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10730   \glshasattribute{\the\glslabeltok}{regular}%
10731   {%
10732     \glissetattribute{\the\glslabeltok}{regular}{false}%
10733   }%
10734   {}}%
10735 }%
10736 }%
10737 {%
10738   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10739 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10740 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
10741 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10742 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10743 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10744 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10745   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10746 }%
10747 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10748   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10749 }%
10750 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10751   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10752 }%
10753 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10754   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10755 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10756 \renewcommand*{\glxtrininlinefullformat}[2]{%
10757   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10758   \glxtrfullsep{##1}%
10759   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10760 }%
10761 \renewcommand*{\glxtrininlinefullplformat}[2]{%
10762   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%

```

```

10763 \glstrfullsep{##1}%
10764 \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10765 }%
10766 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10767 \glstrlonghyphennohshort{##1}{\glsaccesslong{##1}}{##2}%
10768 \glstrfullsep{##1}%
10769 \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10770 }%
10771 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10772 \glstrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10773 \glstrfullsep{##1}%
10774 \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10775 }%

```

The first use full form only displays the long form.

```

10776 \renewcommand*{\glstrfullformat}[2]{%
10777 \glstrlonghyphennohshort{##1}{\glsaccesslong{##1}}{##2}%
10778 }%
10779 \renewcommand*{\glstrfullplformat}[2]{%
10780 \glstrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10781 }%
10782 \renewcommand*{\Glsxtrfullformat}[2]{%
10783 \glstrlonghyphennohshort{##1}{\glsaccesslong{##1}}{##2}%
10784 }%
10785 \renewcommand*{\Glsxtrfullplformat}[2]{%
10786 \glstrlonghyphennohshort{##1}{\glsaccesslongpl{##1}}{##2}%
10787 }%
10788 }

```

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.

```

10789 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10790 {%
10791 \renewcommand*{\CustomAbbreviationFields}{%
10792 name={\glstrlongnoshortname},
10793 sort={\the\glsshorttok},
10794 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10795 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10796 text={\protect\glslonghyphenfont{\the\glslongtok}},%
10797 plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10798 description={\the\glslongtok}%
10799 }%

```

Unset the regular attribute if it has been set.

```

10800 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10801 \glshasattribute{\the\glslabeltok}{regular}%
10802 {%
10803 \glissetattribute{\the\glslabeltok}{regular}{false}%
10804 }%

```

```

10805     {}%
10806   }%
10807 }%
10808 {%
10809   \GlsXtrUseAbbrStyleFmts{long-desc}%
10810 }

```

glsxtrlonghyphen `\glsxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The `<insert>` is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10811 \newcommand*{\glsxtrlonghyphen}[3]{%
    Grouping is needed to localise the redefinitions.
10812  {%
10813    \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10814    \glsfirstlonghyphenfont{#1}%
10815  }%
10816 }

```

rposthyphenshort `\glsxtrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glsxtrlonghyphenshort` but omits the `<long>` part. This always uses the singular short form.

```

10817 \newcommand*{\glsxtrposthyphenshort}[2]{%
10818  {%
10819    \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10820    \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10821    \glsxtrfullsep{#1}%
10822    \glsxtrparen
10823    {\glsfirstabbrvhyphenfont{\glsentryshort{#1}\ifglsxtrininsertinside{#2}\fi}%
10824     \ifglsxtrininsertinside\else{#2}\fi
10825  }%
10826 }%
10827 }

```

hyphensequent `\glsxtrposthyphensequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10828 \newcommand*{\glsxtrposthyphensequent}[2]{%
10829   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%

```

```

10830 \ifglxtrinsertinside \else{#2}\fi
10831 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10832 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10833 {%
10834   \renewcommand*{\CustomAbbreviationFields}{%
10835     name={\glxtrlongshortname},
10836     sort={\the\glsshorttok},
10837     first={\protect\glxfirstlonghyphenfont{\the\glslongtok}},%
10838     firstplural={\protect\glxfirstlonghyphenfont{\the\glslongpltok}},%
10839     plural={\protect\glxabbrvhyphenfont{\the\glsshortpltok}},%
10840     description={\protect\glxlonghyphenfont{\the\glslongtok}}}%
10841   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10842     \csdef{glxtrpostlink\glscategorylabel}{%
10843       \glxtrifwasfirstuse
10844       {%
10845         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10846       }%
10847     }%

```

Put the insertion into the post-link:

```

10848       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10849     }%
10850   }%
10851   \glshasattribute{\the\glslabeltok}{regular}%
10852   {%
10853     \glissetattribute{\the\glslabeltok}{regular}{false}%
10854   }%
10855   {}%
10856 }%
10857 }%
10858 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10859 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10860 \renewcommand*{\glxabbrvfont}[1]{\glxabbrvhyphenfont{##1}}%
10861 \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvhyphenfont{##1}}%
10862 \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlonghyphenfont{##1}}%
10863 \renewcommand*{\glxlongfont}[1]{\glxlonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10864 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10865   \glxabbrvfont{\glxaccessshort{##1}}%
10866 }%
10867 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10868   \glxabbrvfont{\glxaccessshortpl{##1}}%
10869 }%
10870 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

10871 \glsabbrvfont{\Glsaccessshort{##1}}%
10872 }%
10873 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10874 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10875 }%

First use full form:

10876 \renewcommand*{\glsxtrfullformat}[2]{%
10877 \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10878 }%
10879 \renewcommand*{\glsxtrfullplformat}[2]{%
10880 \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10881 }%
10882 \renewcommand*{\Glsxtrfullformat}[2]{%
10883 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10884 }%
10885 \renewcommand*{\Glsxtrfullplformat}[2]{%
10886 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10887 }%

```

In-line format.

```

10888 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10889 \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10890 \ifglsxtrininsertinside{##2}\fi}%
10891 \ifglsxtrininsertinside \else{##2}\fi
10892 }%
10893 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10894 \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10895 \ifglsxtrininsertinside{##2}\fi}%
10896 \ifglsxtrininsertinside \else{##2}\fi
10897 }%
10898 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10899 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10900 \ifglsxtrininsertinside{##2}\fi}%
10901 \ifglsxtrininsertinside \else{##2}\fi
10902 }%
10903 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10904 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10905 \ifglsxtrininsertinside{##2}\fi}%
10906 \ifglsxtrininsertinside \else{##2}\fi
10907 }%
10908 }

```

`ort-hyphen-desc` Like `long-hyphen-postshort-hyphen` but the description must be supplied by the user.

```

10909 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10910 {%
10911 \renewcommand*{\CustomAbbreviationFields}{%
10912 name={\glsxtrlongshortdescname},
10913 sort={\glsxtrlongshortdescsort},%
10914 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%

```

```

10915 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10916 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10917 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10918 }%
10919 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10920 \csdef{glsxtrpostlink\glscategorylabel}{%
10921 \glsxtrifwasfirstuse
10922 {%
10923 \glsxtrposthyphenshort{\glslabel}{\glsinsert}}%
10924 }%
10925 {%

```

Put the insertion into the post-link:

```

10926 \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
10927 }%
10928 }%
10929 \glsattribute{\the\glslabeltok}{regular}%
10930 {%
10931 \glssetattribute{\the\glslabeltok}{regular}{false}%
10932 }%
10933 {}%
10934 }%
10935 }%
10936 {%
10937 \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10938 }

```

rshorthyphenlong

```
\glsxtrshorthyphenlong{<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.

```
10939 \newcommand*{\glsxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10940 {%
```

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.)

```

10941 \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}}%
10942 \glsfirstabbrvhyphenfont{#2\ifglsxtrininsertinside{#4}\fi}%
10943 \ifglsxtrininsertinside\else{#4}\fi
10944 \glsxtrfullsep{#1}%
10945 \glsxtrparen{\glsfirstlonghyphenfont{#3\ifglsxtrininsertinside{#4}\fi}%
10946 \ifglsxtrininsertinside\else{#4}\fi}%
10947 }%
10948 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```

10949 \newabbreviationstyle{short-hyphen-long-hyphen}%
10950 {%
10951   \renewcommand*{\CustomAbbreviationFields}{%
10952     name={\glxtrshortlongname},
10953     sort={\the\glsshorttok},
10954     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10955       \protect\glxtrfullsep{\the\glslabeltok}%
10956       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10957     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10958       \protect\glxtrfullsep{\the\glslabeltok}%
10959       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10960     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10961     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10962   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10963     \glshasattribute{\the\glslabeltok}{regular}%
10964     {%
10965       \glissetattribute{\the\glslabeltok}{regular}{false}%
10966     }%
10967   }%
10968 }%
10969 }%
10970 {%
10971   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10972   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10973   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10974   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10975   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10976   \renewcommand*{\glxtrfullformat}[2]{%
10977     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10978   }%
10979   \renewcommand*{\glxtrfullplformat}[2]{%
10980     \glxtrshorthyphenlong{##1}%
10981     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10982   }%
10983   \renewcommand*{\Glsxtrfullformat}[2]{%
10984     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10985   }%
10986   \renewcommand*{\Glsxtrfullplformat}[2]{%
10987     \glxtrshorthyphenlong{##1}%
10988     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10989   }%
10990 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10991 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%

```

```

10992 {%
10993   \renewcommand*{\CustomAbbreviationFields}{%
10994     name={\glxtrshortlongdescname},
10995     sort={\glxtrshortlongdescsort},
10996     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10997       \protect\glxtrfullsep{\the\glslabeltok}%
10998       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10999     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
11000       \protect\glxtrfullsep{\the\glslabeltok}%
11001       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11002     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11003     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
11004   }%

```

Unset the regular attribute if it has been set.

```

11005   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11006     \glshasattribute{\the\glslabeltok}{regular}%
11007     {%
11008       \glissetattribute{\the\glslabeltok}{regular}{false}%
11009     }%
11010   }%
11011 }%
11012 }%
11013 {%
11014   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11015 }

```

`\glxtrshorthyphen`

```
\glxtrshorthyphen{<short>}{<label>}{<insert>}
```

Used by short-hyphen-postlong-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
11016 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

11017 {%
11018   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}}%
11019   \glsfirstabbrvhyphenfont{#1}%
11020 }%
11021 }

```

`\glxtrposthyphenlong`

```
\glxtrposthyphenlong{<label>}{<insert>}
```

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the *<short>* part. This always uses the singular long form.


```

11022 \newcommand*{\glxtrposthyphenlong}[2]{%
11023 {%
11024   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11025   \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11026   \glxtrfullsep{#1}%
11027   \glxtrparen
11028   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
11029   \ifglxtrininsertinside\else{#2}\fi
11030 }%
11031 }%
11032 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

11033 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11034 {%
11035   \renewcommand*{\CustomAbbreviationFields}{%
11036     name={\glxtrshortlongname},
11037     sort={\the\glsshorttok},
11038     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11039     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11040     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
11041     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11042   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11043     \csdef{glxtrpostlink\glscategorylabel}{%
11044       \glxtrifwasfirstuse
11045       {%
11046         \glxtrposthyphenlong{\glslabel}{\glinsert}%
11047       }%
11048     }%

```

Put the insertion into the post-link:

```

11049       \glxtrposthyphensubsequent{\glslabel}{\glinsert}%
11050     }%
11051   }%
11052   \glshasattribute{\the\glslabeltok}{regular}%
11053   {%
11054     \glissetattribute{\the\glslabeltok}{regular}{false}%
11055   }%
11056   {}%
11057 }%
11058 }%
11059 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11060 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11061 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
11062 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11063 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11064 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

11065 \renewcommand*{\glxtrsubsequentfmt}[2]{%
11066   \glsabbrvfont{\glsaccessshort{##1}}%
11067 }%
11068 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11069   \glsabbrvfont{\glsaccessshortpl{##1}}%
11070 }%
11071 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11072   \glsabbrvfont{\Glsaccessshort{##1}}%
11073 }%
11074 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11075   \glsabbrvfont{\Glsaccessshortpl{##1}}%
11076 }%

```

First use full form:

```

11077 \renewcommand*{\glxtrfullformat}[2]{%
11078   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
11079 }%
11080 \renewcommand*{\glxtrfullplformat}[2]{%
11081   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
11082 }%
11083 \renewcommand*{\Glsxtrfullformat}[2]{%
11084   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
11085 }%
11086 \renewcommand*{\Glsxtrfullplformat}[2]{%
11087   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
11088 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

11089 \renewcommand*{\glxtrinlinefullformat}[2]{%
11090   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
11091   \ifglxtrininsertinside{##2}\fi}%
11092 \ifglxtrininsertinside \else{##2}\fi
11093 }%
11094 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11095   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
11096   \ifglxtrininsertinside{##2}\fi}%
11097 \ifglxtrininsertinside \else{##2}\fi
11098 }%
11099 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11100   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
11101   \ifglxtrininsertinside{##2}\fi}%
11102 \ifglxtrininsertinside \else{##2}\fi
11103 }%
11104 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11105   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
11106   \ifglxtrininsertinside{##2}\fi}%
11107 \ifglxtrininsertinside \else{##2}\fi
11108 }%

```

```
11109 }
```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
11110 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
11111 {%
11112   \renewcommand*{\CustomAbbreviationFields}{%
11113     name={\glstrshortlongdescname},
11114     sort={\glstrshortlongdescsort},%
11115     first={\protect\glfirstabbrvhyphenfont{\the\glsshorttok}},%
11116     firstplural={\protect\glfirstabbrvhyphenfont{\the\glsshortpltok}},%
11117     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11118     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11119   }%
11120   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11121     \csdef{glstrpostlink\glscategorylabel}{%
11122       \glstrifwasfirstuse
11123       {%
11124         \glstrposthyphenlong{\glslabel}{\glsinsert}%
11125       }%
11126       {%
```

Put the insertion into the post-link:

```
11127       \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
11128     }%
11129   }%
11130   \glshasattribute{\the\glslabeltok}{regular}%
11131   {%
11132     \glissetattribute{\the\glslabeltok}{regular}{false}%
11133   }%
11134   {}%
11135 }%
11136 }%
11137 {%
11138   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
11139 }
```

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

```
11140 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
11141 \newcommand*{\glfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
11142 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
11143 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
11144 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
11145 \newcommand*{\glsxtronlyname}{%
```

```
11146 \protect\glsabbrvonlyfont{\the\glsshorttok}}%
```

```
11147 }
```

only-short-only

```
11148 \newabbreviationstyle{long-only-short-only}%
```

```
11149 {%
```

```
11150 \renewcommand*{\CustomAbbreviationFields}{%
```

```
11151 name={\glsxtronlyname},
```

```
11152 sort={\the\glsshorttok},
```

```
11153 first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
```

```
11154 firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
```

```
11155 plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
```

```
11156 description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
11157 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
11158 \glsattribute{\the\glslabeltok}{regular}}%
```

```
11159 {%
```

```
11160 \glssetattribute{\the\glslabeltok}{regular}{false}}%
```

```
11161 }%
```

```
11162 {}%
```

```
11163 }%
```

```
11164 }%
```

```
11165 {%
```

```
11166 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
```

```
11167 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
```

```
11168 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
```

```
11169 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
```

```
11170 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
11171 \renewcommand*{\glsxtrfullformat}[2]{%
```

```
11172 \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}}%
```

```
11173 \ifglsxtrinsertinside\else##2\fi
```

```
11174 }%
```

```
11175 \renewcommand*{\glsxtrfullplformat}[2]{%
```

```
11176 \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}}%
```

```
11177 \ifglsxtrinsertinside\else##2\fi
```

```
11178 }%
```

```
11179 \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

11180 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11181 \ifglxtrinsertinside\else##2\fi
11182 }%
11183 \renewcommand*{\Glsxtrfullplformat}[2]{%
11184 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11185 \ifglxtrinsertinside\else##2\fi
11186 }%

```

The inline full form does show the short form.

```

11187 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11188 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11189 \ifglxtrinsertinside\else##2\fi
11190 \glsxtrfullsep{##1}%
11191 \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshort{##1}}}%
11192 }%
11193 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11194 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11195 \ifglxtrinsertinside\else##2\fi
11196 \glsxtrfullsep{##1}%
11197 \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11198 }%
11199 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11200 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11201 \ifglxtrinsertinside\else##2\fi
11202 \glsxtrfullsep{##1}%
11203 \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11204 }%
11205 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11206 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11207 \ifglxtrinsertinside\else##2\fi
11208 \glsxtrfullsep{##1}%
11209 \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11210 }%
11211 }

```

xtronlydescsort

```

11212 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

11213 \newcommand*{\glxtronlydescname}{%
11214 \protect\glslongfont{\the\glslongtok}%
11215 }

```

short-only-desc

```

11216 \newabbreviationstyle{long-only-short-only-desc}%
11217 {%
11218 \renewcommand*{\CustomAbbreviationFields}{%
11219 name={\glxtronlydescname},
11220 sort={\glxtronlydescsort},%

```

```

11221   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11222   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11223   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
11224   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
11225 }%

```

Unset the regular attribute if it has been set.

```

11226 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11227   \glshasattribute{\the\glslabeltok}{regular}}%
11228   {%
11229     \glssetattribute{\the\glslabeltok}{regular}{false}}%
11230   }%
11231   {}%
11232 }%
11233 }%
11234 {%
11235   \GlsXtrUseAbbrStyleFmts{long-only-short-only}}%
11236 }

```

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:

```

11237 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```
11238 \renewcommand*{\markright}[1]{%
11239   \glxtrmarkhook
11240   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
11241   \glxtrrestoremarkhook
11242 }
```

\markboth Save original definition:

```
11243 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
11244 \renewcommand*{\markboth}[2]{%
11245   \glxtrmarkhook
11246   \@glxtr@org@markboth
11247   {\@glxtrinmark#1\@glxtrnotinmark}%
11248   {\@glxtrinmark#2\@glxtrnotinmark}%
11249   \glxtrrestoremarkhook
11250 }
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
11251 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
11252 \renewcommand*{\@starttoc}[1]{%
11253   \glxtrmarkhook
11254   \@glxtrinmark
11255   \@glxtr@org@@starttoc{#1}%
11256   \@glxtrnotinmark
11257   \glxtrrestoremarkhook
11258 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

glxtrRevertMarks

```
11259 \newcommand*{\glxtrRevertMarks}{%
11260   \let\markright\@glxtr@org@markright
11261   \let\markboth\@glxtr@org@markboth
11262   \let\@starttoc\@glxtr@org@@starttoc
11263 }
```

glxtrRevertTocMarks Just restores \@starttoc.

```
11264 \newcommand*{\glxtrRevertTocMarks}{%
11265   \let\@starttoc\@glxtr@org@@starttoc
11266 }
```

\glxtrifinmark

```
11267 \newcommand*{\glxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
11268 \newrobustcmd*{\@glsxtrinmark}{%
11269   \let\glsxtrifinmark\@firstoftwo
11270 }
```

glsxtrnotinmark

```
11271 \newrobustcmd*{\@glsxtrnotinmark}{%
11272   \let\glsxtrifinmark\@secondoftwo
11273 }
```

eorpdforheading

```
11274 \ifdef\texorpdfstring
11275 {
11276   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11277 }
11278 {
11279   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
11280 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11281 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
11282   \let\@glsxtr@org@MakeUppercase\MakeUppercase
11283   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11284   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11285   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11286   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11287   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11288   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11289   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11290   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11291   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11292   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11293   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11294   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11295   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11296   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11297   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11298   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11299   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11300   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11301   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11302   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11303   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11304   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11305   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```


New definitions

```

11306 \let\glxstrifinmark\@firstoftwo
11307 \let\MakeUppercase\MakeTextUppercase
11308 \let\glxstrtitleorpdforheading\@thirdofthree
11309 \let\glxstrtitleshort\glxstrheadshort
11310 \let\glxstrtitleshortpl\glxstrheadshortpl
11311 \let\Glsxstrtitleshort\Glsxtrheadshort
11312 \let\Glsxstrtitleshortpl\Glsxtrheadshortpl
11313 \let\glxstrtitlename\glxstrheadname
11314 \let\Glsxstrtitlename\Glsxtrheadname
11315 \let\glxstrtitletext\glxstrheadtext
11316 \let\Glsxstrtitletext\Glsxtrheadtext
11317 \let\glxstrtitleplural\glxstrheadplural
11318 \let\Glsxstrtitleplural\Glsxtrheadplural
11319 \let\glxstrtitlefirst\glxstrheadfirst
11320 \let\Glsxstrtitlefirst\Glsxtrheadfirst
11321 \let\glxstrtitlefirstplural\glxstrheadfirstplural
11322 \let\Glsxstrtitlefirstplural\Glsxtrheadfirstplural
11323 \let\glxstrtitlelong\glxstrheadlong
11324 \let\glxstrtitlelongpl\glxstrheadlongpl
11325 \let\Glsxstrtitlelong\Glsxtrheadlong
11326 \let\Glsxstrtitlelongpl\Glsxtrheadlongpl
11327 \let\glxstrtitlefull\glxstrheadfull
11328 \let\glxstrtitlefullpl\glxstrheadfullpl
11329 \let\Glsxstrtitlefull\Glsxtrheadfull
11330 \let\Glsxstrtitlefullpl\Glsxtrheadfullpl
11331 }

```

`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.)

```

11332 \newcommand*{\glxstrrestoremarkhook}{%
11333 \let\glxstrifinmark\@secondoftwo
11334 \let\MakeUppercase\@glxstr@org@MakeUppercase
11335 \let\glxstrtitleorpdforheading\@glxstr@org@glxstrtitleorpdforheading
11336 \let\glxstrtitleshort\@glxstr@org@glxstrtitleshort
11337 \let\glxstrtitleshortpl\@glxstr@org@glxstrtitleshortpl
11338 \let\Glsxstrtitleshort\@glxstr@org@Glsxtrtitleshort
11339 \let\Glsxstrtitleshortpl\@glxstr@org@Glsxtrtitleshortpl
11340 \let\glxstrtitlename\@glxstr@org@glxstrtitlename
11341 \let\Glsxstrtitlename\@glxstr@org@Glsxtrtitlename
11342 \let\glxstrtitletext\@glxstr@org@glxstrtitletext
11343 \let\Glsxstrtitletext\@glxstr@org@Glsxtrtitletext
11344 \let\glxstrtitleplural\@glxstr@org@glxstrtitleplural
11345 \let\Glsxstrtitleplural\@glxstr@org@Glsxtrtitleplural
11346 \let\glxstrtitlefirst\@glxstr@org@glxstrtitlefirst
11347 \let\Glsxstrtitlefirst\@glxstr@org@Glsxtrtitlefirst
11348 \let\glxstrtitlefirstplural\@glxstr@org@glxstrtitlefirstplural

```

```

11349 \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
11350 \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
11351 \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
11352 \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
11353 \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
11354 \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
11355 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
11356 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
11357 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
11358 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glsxtrheadshort Command used to display short form in the page header.

```

11359 \newcommand*{\glsxtrheadshort}[1]{%
11360   \protect\NoCaseChange
11361   {%
11362     \glsifattribute{#1}{headuc}{true}%
11363     {%
11364       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11365     }%
11366     {%
11367       \glsxtrshort[noindex,hyper=false]{#1}[]%
11368     }%
11369   }%
11370 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11371 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11372   \glsxtrshort[noindex,hyper=false]{#1}[]%
11373 }

```

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.

```

11374 \newcommand*{\glsxtrheadshortpl}[1]{%
11375   \protect\NoCaseChange
11376   {%
11377     \glsifattribute{#1}{headuc}{true}%
11378     {%
11379       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11380     }%
11381     {%
11382       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11383     }%
11384   }%
11385 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11386 \newrobustcmd*{\glxtrtitleshortpl}[1]{%
11387   \glxtrshortpl[noindex,hyper=false]{#1}[]%
11388 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11389 \newcommand*{\Glsxtrheadshort}[1]{%
11390   \protect\NoCaseChange
11391   {%
11392     \glsifattribute{#1}{headuc}{true}%
11393     {%
11394       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11395     }%
11396     {%
11397       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11398     }%
11399   }%
11400 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11401 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11402   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11403 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11404 \newcommand*{\Glsxtrheadshortpl}[1]{%
11405   \protect\NoCaseChange
11406   {%
11407     \glsifattribute{#1}{headuc}{true}%
11408     {%
11409       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11410     }%
11411     {%
11412       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11413     }%
11414   }%
11415 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11416 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11417   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11418 }
```

`\glxtrheadname` As above but for the name value.

```

11419 \newcommand*{\glxstrheadname}[1]{%
11420   \protect\NoCaseChange
11421   {%
11422     \gl@ifattribute{#1}{headuc}{true}%
11423     {%
11424       \GLSname[noindex,hyper=false]{#1}[]%
11425     }%
11426     {%
11427       \glSname[noindex,hyper=false]{#1}[]%
11428     }%
11429   }%
11430 }

```

glxstrtitlename Command to display name value in section title and table of contents.

```

11431 \newrobustcmd*{\glxstrtitlename}[1]{%
11432   \glSname[noindex,hyper=false]{#1}[]%
11433 }

```

\Glsxtrheadname First letter converted to upper case

```

11434 \newcommand*{\Glsxtrheadname}[1]{%
11435   \protect\NoCaseChange
11436   {%
11437     \gl@ifattribute{#1}{headuc}{true}%
11438     {%
11439       \GLSname[noindex,hyper=false]{#1}[]%
11440     }%
11441     {%
11442       \glSname[noindex,hyper=false]{#1}[]%
11443     }%
11444   }%
11445 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11446 %\changes{1.21}{2017-11-03}{new}
11447 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11448   \glSname[noindex,hyper=false]{#1}[]%
11449 }

```

\glxstrheadtext As above but for the text value.

```

11450 \newcommand*{\glxstrheadtext}[1]{%
11451   \protect\NoCaseChange
11452   {%
11453     \gl@ifattribute{#1}{headuc}{true}%
11454     {%
11455       \GLStext[noindex,hyper=false]{#1}[]%
11456     }%
11457     {%
11458       \glstext[noindex,hyper=false]{#1}[]%

```

```

11459 }%
11460 }%
11461 }

```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```

11462 \newrobustcmd*{\glsxtrtitletext}[1]{%
11463   \glstext[noindex,hyper=false]{#1}[]%
11464 }

```

`\Glsxtrheadtext` First letter converted to upper case

```

11465 \newcommand*{\Glsxtrheadtext}[1]{%
11466   \protect\NoCaseChange
11467   {%
11468     \glsifattribute{#1}{headuc}{true}%
11469     {%
11470       \GLStext[noindex,hyper=false]{#1}[]%
11471     }%
11472     {%
11473       \Glstext[noindex,hyper=false]{#1}[]%
11474     }%
11475   }%
11476 }

```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```

11477 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11478   \Glstext[noindex,hyper=false]{#1}[]%
11479 }

```

`lsxtrheadplural` As above but for the plural value.

```

11480 \newcommand*{\lsxtrheadplural}[1]{%
11481   \protect\NoCaseChange
11482   {%
11483     \glsifattribute{#1}{headuc}{true}%
11484     {%
11485       \GLSplural[noindex,hyper=false]{#1}[]%
11486     }%
11487     {%
11488       \glsplural[noindex,hyper=false]{#1}[]%
11489     }%
11490   }%
11491 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```

11492 \newrobustcmd*{\sxtrtitleplural}[1]{%
11493   \glsplural[noindex,hyper=false]{#1}[]%
11494 }

```

lsxtrheadplural Convert first letter to upper case.

```
11495 \newcommand*{\Glsxtrheadplural}[1]{%
11496   \protect\NoCaseChange
11497   {%
11498     \glsifattribute{#1}{headuc}{true}%
11499     {%
11500       \GLSplural[noindex,hyper=false]{#1}[]%
11501     }%
11502   }%
11503   \Glsplural[noindex,hyper=false]{#1}[]%
11504 }%
11505 }%
11506 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11507 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11508   \Glsplural[noindex,hyper=false]{#1}[]%
11509 }
```

glxtrheadfirst As above but for the first value.

```
11510 \newcommand*{\glxtrheadfirst}[1]{%
11511   \protect\NoCaseChange
11512   {%
11513     \glsifattribute{#1}{headuc}{true}%
11514     {%
11515       \GLSfirst[noindex,hyper=false]{#1}[]%
11516     }%
11517   }%
11518   \glsfirst[noindex,hyper=false]{#1}[]%
11519 }%
11520 }%
11521 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
11522 \newrobustcmd*{\glxtrtitlefirst}[1]{%
11523   \glsfirst[noindex,hyper=false]{#1}[]%
11524 }
```

Glsxtrheadfirst First letter converted to upper case

```
11525 \newcommand*{\Glsxtrheadfirst}[1]{%
11526   \protect\NoCaseChange
11527   {%
11528     \glsifattribute{#1}{headuc}{true}%
11529     {%
11530       \GLSfirst[noindex,hyper=false]{#1}[]%
11531     }%
11532   }%
```

```

11533     \Glsfirst[noindex,hyper=false]{#1}[]%
11534 }%
11535 }%
11536 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11537 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11538   \Glsfirst[noindex,hyper=false]{#1}[]%
11539 }

```

headfirstplural As above but for the firstplural value.

```

11540 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11541   \protect\NoCaseChange
11542   {%
11543     \glsifattribute{#1}{headuc}{true}%
11544     {%
11545       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11546     }%
11547     {%
11548       \glsfirstplural[noindex,hyper=false]{#1}[]%
11549     }%
11550   }%
11551 }

```

itlefirstplural Command to display firstplural value in section title and table of contents.

```

11552 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11553   \glsfirstplural[noindex,hyper=false]{#1}[]%
11554 }

```

headfirstplural First letter converted to upper case

```

11555 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11556   \protect\NoCaseChange
11557   {%
11558     \glsifattribute{#1}{headuc}{true}%
11559     {%
11560       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11561     }%
11562     {%
11563       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11564     }%
11565   }%
11566 }

```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11567 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11568   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11569 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```
11570 \newcommand*{\glxstrheadlong}[1]{%
11571   \protect\NoCaseChange
11572   {%
11573     \gl@ifattribute{#1}{headuc}{true}%
11574     {%
11575       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11576     }%
11577     {%
11578       \glxstrlong[noindex,hyper=false]{#1}[]%
11579     }%
11580   }%
11581 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
11582 \newrobustcmd*{\glxstrtitlelong}[1]{%
11583   \glxstrlong[noindex,hyper=false]{#1}[]%
11584 }
```

`lsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
11585 \newcommand*{\glxtrheadlongpl}[1]{%
11586   \protect\NoCaseChange
11587   {%
11588     \gl@ifattribute{#1}{headuc}{true}%
11589     {%
11590       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11591     }%
11592     {%
11593       \glxstrlongpl[noindex,hyper=false]{#1}[]%
11594     }%
11595   }%
11596 }
```

`sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
11597 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
11598   \glxstrlongpl[noindex,hyper=false]{#1}[]%
11599 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
11600 \newcommand*{\Glsxtrheadlong}[1]{%
11601   \protect\NoCaseChange
11602   {%
11603     \gl@ifattribute{#1}{headuc}{true}%
11604     {%
11605       \GLSxtrlong[noindex,hyper=false]{#1}[]%
```



```

11606 }%
11607 {%
11608     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11609 }%
11610 }%
11611 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11612 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11613     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11614 }

```

lgsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11615 \newcommand*{\Glsxtrheadlongpl}[1]{%
11616     \protect\NoCaseChange
11617     {%
11618         \glsifattribute{#1}{headuc}{true}%
11619         {%
11620             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11621         }%
11622     }%
11623     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11624 }%
11625 }%
11626 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11627 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11628     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11629 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11630 \newcommand*{\glsxtrheadfull}[1]{%
11631     \protect\NoCaseChange
11632     {%
11633         \glsifattribute{#1}{headuc}{true}%
11634         {%
11635             \GLSxtrfull[noindex,hyper=false]{#1}[]%
11636         }%
11637     }%
11638     \glsxtrfull[noindex,hyper=false]{#1}[]%
11639 }%
11640 }%
11641 }

```

`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
11642 \newrobustcmd*{\glsxtrtitlefull}[1]{%  
11643   \glsxtrfull[noindex,hyper=false]{#1}[]%  
11644 }
```

`lsxtrheadfullpl` Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrfullpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
11645 \newcommand*{\glsxtrheadfullpl}[1]{%  
11646   \protect\NoCaseChange  
11647   {%  
11648     \glsifattribute{#1}{headuc}{true}%  
11649     {%  
11650       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%  
11651     }%  
11652     {%  
11653       \glsxtrfullpl[noindex,hyper=false]{#1}[]%  
11654     }%  
11655   }%  
11656 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
11657 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%  
11658   \glsxtrfullpl[noindex,hyper=false]{#1}[]%  
11659 }
```

`\GLsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
11660 \newcommand*{\GLsxtrheadfull}[1]{%  
11661   \protect\NoCaseChange  
11662   {%  
11663     \glsifattribute{#1}{headuc}{true}%  
11664     {%  
11665       \GLSxtrfull[noindex,hyper=false]{#1}[]%  
11666     }%  
11667     {%  
11668       \GLsxtrfull[noindex,hyper=false]{#1}[]%  
11669     }%  
11670   }%  
11671 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11672 \newrobustcmd*{\Glsxtrtitlefull}[1]{%  
11673   \GLsxtrfull[noindex,hyper=false]{#1}[]%  
11674 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11675 \newcommand*{\Glsxtrheadfullpl}[1]{%
11676   \protect\NoCaseChange
11677   {%
11678     \glsifattribute{#1}{headuc}{true}%
11679     {%
11680       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11681     }%
11682     {%
11683       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11684     }%
11685   }%
11686 }

```

`\sxttrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11687 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11688   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11689 }

```

`\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.

```

11690 \ifdef\texorpdfstring
11691 {
11692   \newcommand*{\glsfmtshort}[1]{%
11693     \texorpdfstring
11694       {\glsxtrtitleshort{#1}}%
11695       {\glsentryshort{#1}}%
11696   }
11697 }
11698 {
11699   \newcommand*{\glsfmtshort}[1]{%
11700     \glsxtrtitleshort{#1}}
11701 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11702 \ifdef\texorpdfstring
11703 {
11704   \newcommand*{\glsfmtshortpl}[1]{%
11705     \texorpdfstring
11706       {\glsxtrtitleshortpl{#1}}%
11707       {\glsentryshortpl{#1}}%
11708   }
11709 }
11710 {
11711   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11712 \glstrtitleshortpl{#1}}
11713 }

```

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).

```

11714 \ifdef\texorpdfstring
11715 {
11716   \newcommand*{\Glsfmtshort}[1]{%
11717     \texorpdfstring
11718       {\glstrtitleshort{#1}}%
11719       {\glentryshort{#1}}%
11720   }
11721 }
11722 {
11723   \newcommand*{\Glsfmtshort}[1]{%
11724     \glstrtitleshort{#1}}
11725 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11726 \ifdef\texorpdfstring
11727 {
11728   \newcommand*{\Glsfmtshortpl}[1]{%
11729     \texorpdfstring
11730       {\glstrtitleshortpl{#1}}%
11731       {\glentryshortpl{#1}}%
11732   }
11733 }
11734 {
11735   \newcommand*{\Glsfmtshortpl}[1]{%
11736     \glstrtitleshortpl{#1}}
11737 }

```

`\glsfmtname` As above but for the name value.

```

11738 \ifdef\texorpdfstring
11739 {
11740   \newcommand*{\glsfmtname}[1]{%
11741     \texorpdfstring
11742       {\glstrtitlename{#1}}%
11743       {\glentryname{#1}}%
11744   }
11745 }
11746 {
11747   \newcommand*{\glsfmtname}[1]{%
11748     \glstrtitlename{#1}}
11749 }

```

`\Glsfmtname` First letter converted to upper case.

```

11750 \ifdef\texorpdfstring
11751 {
11752   \newcommand*{\Glsfmtname}[1]{%
11753     \texorpdfstring
11754     {\Glsxtrtitlename{#1}}%
11755     {\glsentryname{#1}}%
11756   }
11757 }
11758 {
11759   \newcommand*{\Glsfmtname}[1]{%
11760     \Glsxtrtitlename{#1}}
11761 }

```

`\glsfmttext` As above but for the text value.

```

11762 \ifdef\texorpdfstring
11763 {
11764   \newcommand*{\glsfmttext}[1]{%
11765     \texorpdfstring
11766     {\glsxtrtitletext{#1}}%
11767     {\glsentrytext{#1}}%
11768   }
11769 }
11770 {
11771   \newcommand*{\glsfmttext}[1]{%
11772     \glsxtrtitletext{#1}}
11773 }

```

`\Glsfmttext` First letter converted to upper case.

```

11774 \ifdef\texorpdfstring
11775 {
11776   \newcommand*{\Glsfmttext}[1]{%
11777     \texorpdfstring
11778     {\Glsxtrtitletext{#1}}%
11779     {\glsentrytext{#1}}%
11780   }
11781 }
11782 {
11783   \newcommand*{\Glsfmttext}[1]{%
11784     \Glsxtrtitletext{#1}}
11785 }

```

`\glsfmtplural` As above but for the plural value.

```

11786 \ifdef\texorpdfstring
11787 {
11788   \newcommand*{\glsfmtplural}[1]{%
11789     \texorpdfstring
11790     {\glsxtrtitleplural{#1}}%
11791     {\glsentryplural{#1}}%
11792   }

```

```

11793 }
11794 {
11795   \newcommand*{\glsfmtplural}[1]{%
11796     \glsxtrtitleplural{#1}}
11797 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11798 \ifdef\texorpdfstring
11799 {
11800   \newcommand*{\Glsfmtplural}[1]{%
11801     \texorpdfstring
11802       {\Glsxtrtitleplural{#1}}%
11803       {\glsentryplural{#1}}%
11804   }
11805 }
11806 {
11807   \newcommand*{\Glsfmtplural}[1]{%
11808     \Glsxtrtitleplural{#1}}
11809 }

```

`\glsfmtfirst` As above but for the first value.

```

11810 \ifdef\texorpdfstring
11811 {
11812   \newcommand*{\glsfmtfirst}[1]{%
11813     \texorpdfstring
11814       {\glsxtrtitlefirst{#1}}%
11815       {\glsentryfirst{#1}}%
11816   }
11817 }
11818 {
11819   \newcommand*{\glsfmtfirst}[1]{%
11820     \glsxtrtitlefirst{#1}}
11821 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11822 \ifdef\texorpdfstring
11823 {
11824   \newcommand*{\Glsfmtfirst}[1]{%
11825     \texorpdfstring
11826       {\Glsxtrtitlefirst{#1}}%
11827       {\glsentryfirst{#1}}%
11828   }
11829 }
11830 {
11831   \newcommand*{\Glsfmtfirst}[1]{%
11832     \Glsxtrtitlefirst{#1}}
11833 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11834 \ifdef\teorpdfstring
11835 {
11836   \newcommand*\glsfmtfirstpl}[1]{%
11837     \teorpdfstring
11838     {\glsxtrtitlefirstplural{#1}}%
11839     {\glsentryfirstplural{#1}}%
11840   }
11841 }
11842 {
11843   \newcommand*\glsfmtfirstpl}[1]{%
11844     \glsxtrtitlefirstplural{#1}}
11845 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11846 \ifdef\teorpdfstring
11847 {
11848   \newcommand*\Glsfmtfirstpl}[1]{%
11849     \teorpdfstring
11850     {\Glsxtrtitlefirstplural{#1}}%
11851     {\glsentryfirstplural{#1}}%
11852   }
11853 }
11854 {
11855   \newcommand*\Glsfmtfirstpl}[1]{%
11856     \Glsxtrtitlefirstplural{#1}}
11857 }

```

`\glsfmtlong` As above but for the long value.

```

11858 \ifdef\teorpdfstring
11859 {
11860   \newcommand*\glsfmtlong}[1]{%
11861     \teorpdfstring
11862     {\glsxtrtitlelong{#1}}%
11863     {\glsentrylong{#1}}%
11864   }
11865 }
11866 {
11867   \newcommand*\glsfmtlong}[1]{%
11868     \glsxtrtitlelong{#1}}
11869 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11870 \ifdef\teorpdfstring
11871 {
11872   \newcommand*\Glsfmtlong}[1]{%
11873     \teorpdfstring
11874     {\Glsxtrtitlelong{#1}}%
11875     {\glsentrylong{#1}}%
11876   }

```

```

11877 }
11878 {
11879   \newcommand*{\Glsfmtlong}[1]{%
11880     \Glsxtrtitlelong{#1}}
11881 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11882 \ifdef\texorpdfstring
11883 {
11884   \newcommand*{\glsfmtlongpl}[1]{%
11885     \texorpdfstring
11886     {\Glsxtrtitlelongpl{#1}}%
11887     {\Glsentrylongpl{#1}}%
11888   }
11889 }
11890 {
11891   \newcommand*{\glsfmtlongpl}[1]{%
11892     \Glsxtrtitlelongpl{#1}}
11893 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11894 \ifdef\texorpdfstring
11895 {
11896   \newcommand*{\Glsfmtlongpl}[1]{%
11897     \texorpdfstring
11898     {\Glsxtrtitlelongpl{#1}}%
11899     {\Glsentrylongpl{#1}}%
11900   }
11901 }
11902 {
11903   \newcommand*{\Glsfmtlongpl}[1]{%
11904     \Glsxtrtitlelongpl{#1}}
11905 }

```

`\glsfmtfull` In-line full format.

```

11906 \ifdef\texorpdfstring
11907 {
11908   \newcommand*{\glsfmtfull}[1]{%
11909     \texorpdfstring
11910     {\Glsxtrtitlefull{#1}}%
11911     {\Glsxtrinlinefullformat{#1}{}}%
11912   }
11913 }
11914 {
11915   \newcommand*{\glsfmtfull}[1]{%
11916     \Glsxtrtitlefull{#1}}
11917 }

```

`\Glsfmtfull` First letter converted to upper case.


```

11918 \ifdef\teorpdfstring
11919 {
11920   \newcommand*{\Glsfmtfull}[1]{%
11921     \teorpdfstring
11922     {\Glsxtrtitlefull{#1}}%
11923     {\Glsxtrinlinefullformat{#1}{}}%
11924   }
11925 }
11926 {
11927   \newcommand*{\Glsfmtfull}[1]{%
11928     \Glsxtrtitlefull{#1}}
11929 }

```

`\glsfmtfullpl` In-line full plural format.

```

11930 \ifdef\teorpdfstring
11931 {
11932   \newcommand*{\glsfmtfullpl}[1]{%
11933     \teorpdfstring
11934     {\glsxtrtitlefullpl{#1}}%
11935     {\glsxtrinlinefullplformat{#1}{}}%
11936   }
11937 }
11938 {
11939   \newcommand*{\glsfmtfullpl}[1]{%
11940     \glsxtrtitlefullpl{#1}}
11941 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11942 \ifdef\teorpdfstring
11943 {
11944   \newcommand*{\Glsfmtfullpl}[1]{%
11945     \teorpdfstring
11946     {\Glsxtrtitlefullpl{#1}}%
11947     {\Glsxtrinlinefullplformat{#1}{}}%
11948   }
11949 }
11950 {
11951   \newcommand*{\Glsfmtfullpl}[1]{%
11952     \Glsxtrtitlefullpl{#1}}
11953 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11954 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```

```

11955 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11956 }

```

sariesExtraLang

```

11957 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11958 \ProvidesFile{glossariesxtr-#1.ldf}%
11959 }

```

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.

```

11960 \newcommand{\glxtr@loaddialect}{%
11961 \IfTrackedLanguageFileExists{\this@dialect}%
11962 {glossariesxtr-}% prefix
11963 {.ldf}%
11964 {%
11965 \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11966 }%
11967 {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11968 \@glxtrdialecthook
11969 }

```

```

11970 \ifpackageloaded{tracklang}
11971 {%
11972 \AnyTrackedLanguages
11973 {%
11974 \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11975 }%
11976 {}%
11977 }
11978 {}

```

Load `glossaries-extra-stylemods` if required.

```

11979 \@glxtr@redefstyles

```

and set the style:

```

11980 \@glxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11981 \NeedsTeXFormat{LaTeX2e}
11982 \ProvidesPackage{glossaries-extra-bib2gls}[2018/12/01 1.38 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11983 \newcommand*{\glshex}{\string\u}

```

`\glscapturedgroup`

```

11984 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```

11985 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11986   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11987 }

```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11988 \newcommand*{\GlsXtrProvideCommand}{\providecommand}

```

`\GlsXtrRenewCommand` Like `\renewcommand` but only generates a warning rather than an error if the command isn't defined.

```

11989 \newcommand*{\GlsXtrRenewCommand}{\@star@or@long\GlsXtr@renewcommand}

```

`\GlsXtr@renewcommand`

```

11990 \newcommand*{\GlsXtr@renewcommand}[1]{%
11991   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
11992   \expandafter\@ifundefined\@gtempa
11993   {%
11994     \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
11995   }%
11996   \relax
11997   \relax
11998   \let\@ifdefinable\@rc@ifdefinable
11999   \new@command#1%
12000 }

```

`\GlsXtr@GlossaryLocation` For use with indexcounter and bib2gls.

```

12001 \newcommand*{\GlsXtr@GlossaryLocation}[2]{#1}

```

`\GlsXtrIndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with indexcounter and bib2gls.

```

12002 \ifdef\hyperref
12003 {%
12004   \newcommand*{\GlsXtrIndexCounterLink}[2]{%

```

```

12005 \glstrifhasfield{indexcounter}{#2}%
12006 {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
12007 {#1}%
12008 }
12009 }
12010 {
12011 \newcommand*\GlsXtrIndexCounterLink}[2]{#1}
12012 }

```

`\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.

```

12013 \newcommand*\GlsXtrDualField{dual}

```

`\GlsXtrDualBackLink` `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```

12014 \newcommand*\GlsXtrDualBackLink}[2]{%
12015 \glstrifhasfield{\GlsXtrDualField}{#2}%
12016 {\glshyperlink[#1]{\glscurrentfieldvalue}}%
12017 {#2}%
12018 }

```

`\TeXEntryAliases` Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```

12019 \newcommand*\GlsXtrBibTeXEntryAliases{%
12020 article=bibtexentry,
12021 book=bibtexentry,
12022 booklet=bibtexentry,
12023 conference=bibtexentry,
12024 inbook=bibtexentry,
12025 incollection=bibtexentry,
12026 inproceedings=bibtexentry,
12027 manual=bibtexentry,
12028 mastersthesis=bibtexentry,
12029 misc=bibtexentry,
12030 phdthesis=bibtexentry,
12031 proceedings=bibtexentry,
12032 techreport=bibtexentry,
12033 unpublished=bibtexentry
12034 }

```

`\provideBibTeXFields` Convenient shortcut to define the standard BibTeX fields.

```
12035 \newcommand*{\GlsXtrProvideBibTeXFields}{%
12036   \glsaddstoragekey{address}{}{\glstrbibaddress}%
12037   \glsaddstoragekey{author}{}{\glstrbibauthor}%
12038   \glsaddstoragekey{booktitle}{}{\glstrbibbooktitle}%
12039   \glsaddstoragekey{chapter}{}{\glstrbibchapter}%
12040   \glsaddstoragekey{edition}{}{\glstrbibedition}%
12041   \glsaddstoragekey{howpublished}{}{\glstrbibhowpublished}%
12042   \glsaddstoragekey{institution}{}{\glstrbibinstitution}%
12043   \glsaddstoragekey{journal}{}{\glstrbibjournal}%
12044   \glsaddstoragekey{month}{}{\glstrbibmonth}%
12045   \glsaddstoragekey{note}{}{\glstrbibnote}%
12046   \glsaddstoragekey{number}{}{\glstrbibnumber}%
12047   \glsaddstoragekey{organization}{}{\glstrbiborganization}%
12048   \glsaddstoragekey{pages}{}{\glstrbibpages}%
12049   \glsaddstoragekey{publisher}{}{\glstrbibpublisher}%
12050   \glsaddstoragekey{school}{}{\glstrbibschooll}%
12051   \glsaddstoragekey{series}{}{\glstrbibseries}%
12052   \glsaddstoragekey{title}{}{\glstrbibtitle}%
12053   \glsaddstoragekey{bibtextype}{}{\glstrbibtype}%
12054   \glsaddstoragekey{volume}{}{\glstrbibvolume}%
12055 }
```

Multiple supplementary references are only supported with `bib2gls`.

`\ltisupplocation` This is like `\glstrsupphypernumber` 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.

```
12056 \newcommand*{\glstrmultisupplocation}[3]{%
12057   {%
12058     \def\glstrsupplocationurl{#2}%
12059     \glshypernumber{#1}%
12060   }%
12061 }
```

`\trdisplaysupplloc` `\glstrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```
12062 \newcommand*{\glstrdisplaysupplloc}[5]{%
12063   \setentrycounter{#1}{#2}%
12064   \glstrmultisupplocation{#5}{#4}{#3}%
12065 }
```

`splaylocnameref` `\glstrdisplaylocnameref{<prefix>}{<counter>}{<format>}{<location>}{<name>}{<href>}{<hcounter>}{<external file>}` Used with the `[nameref]` record package option. The `<href>` argument was obtained from `\@currentHref` and the `<hcounter>` argument was obtained from `\theHentrycounter`, which is more reliable. If `hyperref` hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```
12066 \ifundef\hyperlink
12067 {
12068   \newcommand*{\glstrdisplaylocnameref}[8]{%
12069     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
12070   }
12071 }
12072 {
```

Default action uses `<hcounter>`. Equations and pages typically don't have a title, so check the counter name.

```
12073   \newcommand*{\glstrdisplaylocnameref}[8]{%
12074     \ifstrequal{#2}{equation}%
12075     {\glsxtrnameref link{#3}{( #4 )}{#2.#7}{#8}}%
12076     {%
12077       \ifstreempty{#5}%
12078       {%
```

No title, so just use the location as the link text.

```
12079       \glsxtrnameref link{#3}{#4}{#2.#7}{#8}%
12080     }%
12081     {%
12082       \ifstrequal{#2}{page}%
12083       {\glsxtrnameref link{#3}{#4}{#2.#7}{#8}}%
12084       {\glsxtrnameref link{#3}{#5}{#2.#7}{#8}}%
12085     }%
12086   }%
12087 }
12088 }
```

`lsxtrnameref link` `\glstrfmt nameref link{<format>}{<title>}{<href>}{<external file>}`

```
12089 \newcommand*{\glsxtrnameref link}[4]{%
```

Locally change `\glshypernumber` to `\@firstofone` to remove the normal location hyperlink.

```
12090   \begingroup
12091   \let\glshypernumber\@firstofone
```

If the `<external file>` argument is empty, an internal link is used, otherwise an external one is needed.

```
12092   \ifstreempty{#4}%
12093   {\glstrfmt internal nameref{#3}{#1}{#2}}%
```

```

12094     {\glstrfmtexternalnameref{#3}{#1}{#2}{#4}}%
12095 \endgroup
12096 }

```

```

lsxtrnameloclink \glstrnamerefloclink{<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.

```

12097 \newcommand{\glstrnameloclink}[6]{%
12098 \begingroup
12099 \setentrycounter[#1]{#2}%
12100 \def\glstr@locationhypertext{#5}%
12101 \let\glshypernumber\@firstofone
12102 \def\@glsnumberformat{#3}%
12103 \def\glstrsupplocationurl{#6}%
12104 \toks@={}%
12105 \@glstr@bibgls@removespaces#4 \@nil
12106 \endgroup
12107 }

```

`ls@removespaces`

```

12108 \def\@glstr@bibgls@removespaces#1 #2\@nil{%
12109 \toks@=\expandafter{\the\toks@#1}%
12110 \ifx\#2\%
12111 \edef\x{\the\toks@}%
12112 \ifx\x\empty
12113 \else
12114 \protected@edef\x{\glstrycounter\@glo@counterprefix\the\toks@}%
12115 \ifvoid\glstrsupplocationurl
12116 {%
12117 \expandafter\glstrfmtinternalnameref\expandafter{\x}%
12118 {\@glsnumberformat}{\glstr@locationhypertext}%
12119 }%
12120 {%
12121 \expandafter\glstrfmtexternalnameref\expandafter{\x}%
12122 {\@glsnumberformat}{\glstr@locationhypertext}{\glstrsupplocationurl}%
12123 }%
12124 \fi
12125 \else
12126 \@gls@ReturnAfterFi{%
12127 \glstr@bibgls@removespaces#2\@nil
12128 }%
12129 \fi
12130 }

```

tinternalnameref

`\glstrfmtinternalnameloc{<target>}{<format>}{<title>}`

```
12131 \newcommand*{\glstrfmtinternalnameref}[3]{%
12132   \csuse{#2}{\glsdohyperlink{#1}{#3}}%
12133 }
```

texternalnameref

`\glstrfmtexternalnameloc{<target>}{<format>}{<title>}{<file>}`

```
12134 \newcommand*{\glstrfmtexternalnameref}[4]{%
12135   \csuse{#2}{\hyperref{#4}{#1}{#3}}%
12136 }
```

\glstrSetWidest

`\glstrSetWidest{<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.

```
12137 \newcommand*{\glstrSetWidest}[3]{%
```

Check which style options have been provided. (The style packages may not have been loaded.)

```
12138   \ifdef\glupdatewidest
12139   {%
12140     \ifdef\glslongextraUpdateWidest
12141     {%
```

Relevant style packages all loaded. If the *<type>* has been given, append to glossary preamble.

```
12142     \ifstrempy{#1}
12143     {%
12144       \glupdatewidest[#2]{#3}%
12145       \ifnum#2=0\relax
12146         \glslongextraUpdateWidest{#3}%
12147       \else
12148         \glslongextraUpdateWidestChild{#2}{#3}%
12149       \fi
12150     }%
12151   {%
12152     \apptoglossarypreamble[#1]{\glupdatewidest[#2]{#3}}%
12153     \ifnum#2=0\relax
12154       \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12155     \else
12156       \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12157     \fi
```



```

12158     }%
12159     }%
12160     {%

```

Only alttree.

```

12161     \ifstrempy{#1}
12162     {%
12163         \glsupdatewidest[#2]{#3}%
12164     }%
12165     {%
12166         \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
12167     }%
12168     }%
12169 }%
12170 {%

```

`\glsupdatewidest` hasn't been defined. This could just mean that the `glossaries-extra-stylemods` package hasn't been loaded.

```

12171     \ifdef\glssetwidest
12172     {%
12173         \ifdef\glslongextraUpdateWidest
12174         {%

```

Relevant glossary-tree and glossary-longextra have been loaded. If the `<type>` has been given, append to glossary preamble.

```

12175     \ifstrempy{#1}
12176     {%
12177         \glssetwidest[#2]{#3}%
12178         \ifnum#2=0\relax
12179             \glslongextraUpdateWidest{#3}%
12180         \else
12181             \glslongextraUpdateWidestChild{#2}{#3}%
12182         \fi
12183     }%
12184     {%
12185         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
12186         \ifnum#2=0\relax
12187             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12188         \else
12189             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12190         \fi
12191     }%
12192     }%
12193     {%

```

Only alttree.

```

12194     \ifstrempy{#1}
12195     {%
12196         \glssetwidest[#2]{#3}%
12197     }%
12198     {%

```

```

12199         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
12200     }%
12201 }%
12202 }%
12203 {%
12204     \ifdef\glslongextraUpdateWidest
12205     {%
        glossary-longextra has been loaded.
12206         \ifstrempy{#1}
12207         {%
12208             \ifnum#2=0\relax
12209                 \glslongextraUpdateWidest{#3}%
12210             \else
12211                 \glslongextraUpdateWidestChild{#2}{#3}%
12212             \fi
12213         }%
12214     {%
12215         \ifnum#2=0\relax
12216             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
12217         \else
12218             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
12219         \fi
12220     }%
12221 }%
        Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.
12222 }%
12223 }%
12224 }%
12225 }

```

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).

```

12226 \newcommand*\glsxtrSetWidestFallback}[2]{%
12227     \ifnum#1=0\relax
12228         \ifdef\glsFindWidestTopLevelName
12229         {%
12230             \glsFindWidestTopLevelName[#2]%
12231         }%
12232     {%
12233         \GlossariesExtraWarning{You need stylemods={tree} to
12234             provide a fallback for set-widest}%
12235     }%

```

```

12236 \else
12237 \ifdef\glsFindWidestLevelTwo
12238 {%
12239 \glsFindWidestLevelTwo[#2]%
12240 \ifdef\glslongextraUpdateWidestChild
12241 {%
12242 \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
12243 \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
12244 }%
12245 {}%
12246 }%
12247 {%
12248 \GlossariesExtraWarning{You need stylemods={tree} to
12249 provide a fallback for set-widest}%
12250 }%
12251 \fi
12252 }

```

`r@labelprefixes` List of label prefixes.

```

12253 \newcommand*{\@glxtr@labelprefixes}{}

```

`arlabelprefixes` List of label prefixes.

```

12254 \newcommand*{\glxtrclearlabelprefixes}{%
12255 \renewcommand*{\@glxtr@labelprefixes}{}%
12256 }

```

`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.

```

12257 \newcommand*{\glxtraddlabelprefix}[1]{%
12258 \ifstrepty{#1}%
12259 {\glxtraddlabelprefix{\empty}}%
12260 {%
12261 \ifdefempty\@glxtr@labelprefixes
12262 {\def\@glxtr@labelprefixes{#1}}%
12263 {\appto\@glxtr@labelprefixes{,#1}}%
12264 }%
12265 }

```

`pendlabelprefix` Inserts at the start of the list.

```

12266 \newcommand*{\glxtrprependlabelprefix}[1]{%
12267 \ifstrepty{#1}%
12268 {\glxtrprependlabelprefix{\empty}}%
12269 {%
12270 \ifdefempty\@glxtr@labelprefixes
12271 {\def\@glxtr@labelprefixes{#1}}%
12272 {\preto\@glxtr@labelprefixes{#1,}}%
12273 }%
12274 }

```

nlabelfixlist `\glstrifinlabelfixlist{<prefix>}{<true>}{<false>}`

Test if the given prefix is in the list.

```
12275 \newcommand*{\glstrifinlabelfixlist}[3]{%
12276   \ifstrempy{#1}%
12277   {\glstrifinlabelfixlist{\empty}{#2}{#3}}%
12278   {%
12279     \DTLifinlist{#1}{\@glstr@labelfixes}{#2}{#3}%
12280   }%
12281 }
```

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**.

```
12282 \AtBeginDocument{%
12283   \protected@write\@auxout{}{\string\providecommand{\string\@glstr@prefixlabellist}[1]{}}%
12284   \protected@write\@auxout{}{\string\@glstr@prefixlabellist{\@glstr@labelfixes}}%
12285 }
```

t@prefixedlabel Iterate through all the prefixes and find the first prefix and label combination that exists. If none found, this could mean that it's the first \LaTeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```
12286 \newcommand*{\@glstr@get@prefixedlabel}[1]{%
12287   \begingroup
```

Initialise to the unprefix label in the event that the list is empty.

```
12288   \edef\@glstr@thislabel{#1}%
12289   \@for\@glstr@prefix:=\@glstr@labelfixes\do
12290   {%
12291     \edef\@glstr@thislabel{\@glstr@prefix#1}%
12292     \ifglstryexists{\@glstr@thislabel}{\@endfortrue}{}%
12293   }%
12294   \edef\x{\endgroup\noexpand\def\noexpand\@glstr@thislabel{\@glstr@thislabel}}\x
12295 }
```

\dgl Like \gls but tries the prefixes. (Can't use \pgls as that's provided by glossaries-prefix.) Since this command is designed for bib2gls's dual entry system, the "d" stands for "dual".

```
12296 \newrobustcmd*{\dgl}{\@glstr@hyp@opt\dgl}
```

\@dgl

```
12297 \newcommand*{\@dgl}[2][{}]{%
12298   \@glstr@get@prefixedlabel{#2}%
12299   \new@ifnextchar[{\@glstr@{#1}{\@glstr@thislabel}}{\@glstr@{#1}{\@glstr@thislabel}}[{}]{%
12300 }
```

\dglspl

```
12301 \newrobustcmd*{\dglspl}{\@glstr@hyp@opt\dglspl}
```

```

\@dglsp1
12302 \newcommand*{\@dglsp1}[2] [] {%
12303   \@glsxtr@get@prefixedlabel{#2}%
12304   \new@ifnextchar[{\@glspl@{#1}{\@gls@thislabel}}{\@glspl@{#1}{\@gls@thislabel}}[]}%
12305 }

\dGls
12306 \newrobustcmd*{\dGls}{\@gls@hyp@opt\dGls}

\@dGls
12307 \newcommand*{\@dGls}[2] [] {%
12308   \@glsxtr@get@prefixedlabel{#2}%
12309   \new@ifnextchar[{\@Gls@{#1}{\@gls@thislabel}}{\@Gls@{#1}{\@gls@thislabel}}[]}%
12310 }

\dGlspl
12311 \newrobustcmd*{\dGlspl}{\@gls@hyp@opt\dGlspl}

\@dGlspl
12312 \newcommand*{\@dGlspl}[2] [] {%
12313   \@glsxtr@get@prefixedlabel{#2}%
12314   \new@ifnextchar[{\@Glspl@{#1}{\@gls@thislabel}}{\@Glspl@{#1}{\@gls@thislabel}}[]}%
12315 }

\dGLS
12316 \newrobustcmd*{\dGLS}{\@gls@hyp@opt\dGLS}

\@dGLS
12317 \newcommand*{\@dGLS}[2] [] {%
12318   \@glsxtr@get@prefixedlabel{#2}%
12319   \new@ifnextchar[{\@GLS@{#1}{\@gls@thislabel}}{\@GLS@{#1}{\@gls@thislabel}}[]}%
12320 }

\dGLSpl
12321 \newrobustcmd*{\dGLSpl}{\@gls@hyp@opt\dGLSpl}

\@dGLSpl
12322 \newcommand*{\@dGLSpl}[2] [] {%
12323   \@glsxtr@get@prefixedlabel{#2}%
12324   \new@ifnextchar[{\@GLSpl@{#1}{\@gls@thislabel}}{\@GLSpl@{#1}{\@gls@thislabel}}[]}%
12325 }

\dglslink Like \glslink but tries the prefixes.
12326 \newrobustcmd*{\dglslink}[3] [] {%
12327   \@glsxtr@get@prefixedlabel{#2}%
12328   \glslink[#1]{\@gls@thislabel}{#3}%
12329 }

```

`\dglldisp` Like `\glldisp` but tries the prefixes.

```
12330 \newrobustcmd*{\dglldisp}[3][]{%  
12331   \@glxtr@get@prefixedlabel{#2}%  
12332   \glldisp[#1]{\@gl@thislabel}{#3}%  
12333 }
```

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`

```
12334 \providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
12335 \providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
12336 \providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
12337 \providecommand*{\Zeta}{\mathrm{Z}}
```

`\Eta`

```
12338 \providecommand*{\Eta}{\mathrm{H}}
```

`\Iota`

```
12339 \providecommand*{\Iota}{\mathrm{I}}
```

`\Kappa`

```
12340 \providecommand*{\Kappa}{\mathrm{K}}
```

`\Mu`

```
12341 \providecommand*{\Mu}{\mathrm{M}}
```

`\Nu`

```
12342 \providecommand*{\Nu}{\mathrm{N}}
```

`\Omicron`

```
12343 \providecommand*{\Omicron}{\mathrm{O}}
```

`\Rho`

```
12344 \providecommand*{\Rho}{\mathrm{P}}
```

```

\Tau
12345 \providecommand*\Tau{\mathrm{T}}

\Chi
12346 \providecommand*\Chi{\mathrm{X}}

\Digamma
12347 \providecommand*\Digamma{\mathrm{F}}

\omicron
12348 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
12349 \@ifpackageloaded{upgreek}%
12350 {

\Upalpha
12351   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
12352   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
12353   \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
12354   \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
12355   \providecommand*\Upeta{\mathrm{H}}

\Upiota
12356   \providecommand*\Upiota{\mathrm{I}}

\Upkappa
12357   \providecommand*\Upkappa{\mathrm{K}}

\Upmu
12358   \providecommand*\Upmu{\mathrm{M}}

\Upnu
12359   \providecommand*\Upnu{\mathrm{N}}

\Upomicron
12360   \providecommand*\Upomicron{\mathrm{O}}

```

```

\Uprho
12361 \providecommand*\Uprho{\mathrm{P}}

\Uptau
12362 \providecommand*\Uptau{\mathrm{T}}

\Upchi
12363 \providecommand*\Upchi{\mathrm{X}}

\upomicron
12364 \providecommand*\upomicron{\mathrm{o}}

12365 }%
12366 {}% upgreek.sty not loaded

```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-<tag>.ldf` (where *<tag>* identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of *<tag>*. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```

sort-rule={\glxtrcontrolrules
; \glxtrspacerules
; \glxtrnonprintablerules
; \glxtrcombiningdiacriticrules
, \glxtrhyphenrules
< \glxtrgeneralpuncrules
< \glxtrdigitrules
< \glxtrfractionrules
< \glxtrGeneralLatinIVrules
< \glxtrMathItalicGreekIrules
}

```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the 'ignored characters' section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they've been made active.

```

12367 \newcommand*\glxtrcontrolrules{%
12368 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
12369 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001

```



```

12370 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
12371 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
12372 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
12373 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
12374 0010\string'\string=\glshex 0011
12375 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
12376 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
12377 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
12378 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
12379 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
12380 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
12381 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
12382 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
12383 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
12384 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
12385 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
12386 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
12387 }

```

lsxtrspacerules These are space characters.

```

12388 \newcommand*{\glsxtrspacerules}{%
12389 \string' \string'\string;
12390 \string'\glshex 00A0\string'\string;
12391 \string'\glshex 2000\string'\string;
12392 \string'\glshex 2001\string'\string;
12393 \string'\glshex 2002\string'\string;
12394 \string'\glshex 2003\string'\string;
12395 \string'\glshex 2004\string'\string;
12396 \string'\glshex 2005\string'\string;
12397 \string'\glshex 2006\string'\string;
12398 \string'\glshex 2007\string'\string;
12399 \string'\glshex 2008\string'\string;
12400 \string'\glshex 2009\string'\string;
12401 \string'\glshex 200A\string'\string;
12402 \string'\glshex 3000\string'
12403 }

```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

12404 \newcommand*{\glsxtrnonprintablerules}{%
12405 \string'\glshex FEFF\string'\string;
12406 \string'\glshex 000A\string'\string;
12407 \string'\glshex 0009\string'\string;
12408 \string'\glshex 000C\string'\string;
12409 \string'\glshex 000B\string'
12410 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

12411 \newcommand*{\glsxtrcombiningdiacriticrules}{%
12412 \glsxtrcombiningdiacriticrules\string;

```

```

12413 \glxtrcombingdiacriticIIrules\string;
12414 \glxtrcombingdiacriticIIIrules\string;
12415 \glxtrcombingdiacriticIVrules
12416 }

```

diacriticIrules First set of combining diacritic marks.

```

12417 \newcommand*{\glxtrcombingdiacriticIrules}{%
12418 \glshex 0301\string;% combining acute
12419 \glshex 0300\string;% combining grave
12420 \glshex 0306\string;% combining breve
12421 \glshex 0302\string;% combining circumflex
12422 \glshex 030C\string;% combining caron
12423 \glshex 030A\string;% combining ring
12424 \glshex 030D\string;% combining vertical line above
12425 \glshex 0308\string;% combining diaeresis
12426 \glshex 030B\string;% combining double acute
12427 \glshex 0303\string;% combining tilde
12428 \glshex 0307\string;% combining dot above
12429 \glshex 0304% combining macron
12430 }

```

diacriticIIrules Second set of combining diacritic marks.

```

12431 \newcommand*{\glxtrcombingdiacriticIIrules}{%
12432 \glshex 0337\string;% combining short solidus overlay
12433 \glshex 0327\string;% combining cedilla
12434 \glshex 0328\string;% combining ogonek
12435 \glshex 0323\string;% combining dot below
12436 \glshex 0332\string;% combining low line
12437 \glshex 0305\string;% combining overline
12438 \glshex 0309\string;% combining hook above
12439 \glshex 030E\string;% combining double vertical line above
12440 \glshex 030F\string;% combining double grave accent
12441 \glshex 0310\string;% combining candrabindu
12442 \glshex 0311\string;% combining inverted breve
12443 \glshex 0312\string;% combining turned comma above
12444 \glshex 0313\string;% combining comma above
12445 \glshex 0314\string;% combining reversed comma above
12446 \glshex 0315\string;% combining comma above right
12447 \glshex 0316\string;% combining grave accent below
12448 \glshex 0317% combining acute accent below
12449 }

```

diacriticIIIrules Third set of combining diacritic marks.

```

12450 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
12451 \glshex 0318\string;% combining left tack below
12452 \glshex 0319\string;% combining right tack below
12453 \glshex 031A\string;% combining left angle above
12454 \glshex 031B\string;% combining horn
12455 \glshex 031C\string;% combining left half ring below

```

12456 \glshex 031D\string;% combining up tack below
 12457 \glshex 031E\string;% combining down tack below
 12458 \glshex 031F\string;% combining plus sign below
 12459 \glshex 0320\string;% combining minus sign below
 12460 \glshex 0321\string;% combining palatalized hook below
 12461 \glshex 0322\string;% combining retroflex hook below
 12462 \glshex 0324\string;% combining diaeresis below
 12463 \glshex 0325\string;% combining ring below
 12464 \glshex 0326\string;% combining comma below
 12465 \glshex 0329\string;% combining vertical line below
 12466 \glshex 032A\string;% combining bridge below
 12467 \glshex 032B\string;% combining inverted double arch below
 12468 \glshex 032C\string;% combining caron below
 12469 \glshex 032D\string;% combining circumflex accent below
 12470 \glshex 032E\string;% combining breve below
 12471 \glshex 032F\string;% combining inverted breve below
 12472 \glshex 0330\string;% combining tilde below
 12473 \glshex 0331\string;% combining macron below
 12474 \glshex 0333\string;% combining double low line
 12475 \glshex 0334\string;% combining tilde overlay
 12476 \glshex 0335\string;% combining short stroke overlay
 12477 \glshex 0336\string;% combining long stroke overlay
 12478 \glshex 0338\string;% combining long solidus overlay
 12479 \glshex 0339\string;% combining combining right half ring below
 12480 \glshex 033A\string;% combining inverted bridge below
 12481 \glshex 033B\string;% combining square below
 12482 \glshex 033C\string;% combining seagull below
 12483 \glshex 033D\string;% combining x above
 12484 \glshex 033E\string;% combining vertical tilde
 12485 \glshex 033F\string;% combining double overline
 12486 \glshex 0342\string;% combining Greek perispomeni
 12487 \glshex 0344\string;% combining Greek dialytika tonos
 12488 \glshex 0345\string;% combining Greek ypogegrammeni
 12489 \glshex 0360\string;% combining double tilde
 12490 \glshex 0361\string;% combining double inverted breve
 12491 \glshex 0483\string;% combining Cyrillic titlo
 12492 \glshex 0484\string;% combining Cyrillic palatalization
 12493 \glshex 0485\string;% combining Cyrillic dasia pneumata
 12494 \glshex 0486% combining Cyrillic psili pneumata
 12495 }

iacriticIVrules Fourth set of combining diacritic marks.

12496 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
 12497 \glshex 20D0\string;% combining left harpoon above
 12498 \glshex 20D1\string;% combining right harpoon above
 12499 \glshex 20D2\string;% combining long vertical line overlay
 12500 \glshex 20D3\string;% combining short vertical line overlay
 12501 \glshex 20D4\string;% combining anticlockwise arrow above
 12502 \glshex 20D5\string;% combining clockwise arrow above

```

12503 \glshex 20D6\string;% combining left arrow above
12504 \glshex 20D7\string;% combining right arrow above
12505 \glshex 20D8\string;% combining ring overlay
12506 \glshex 20D9\string;% combining clockwise ring overlay
12507 \glshex 20DA\string;% combining anticlockwise ring overlay
12508 \glshex 20DB\string;% combining three dots above
12509 \glshex 20DC\string;% combining four dots above
12510 \glshex 20DD\string;% combining enclosing circle
12511 \glshex 20DE\string;% combining enclosing square
12512 \glshex 20DF\string;% combining enclosing diamond
12513 \glshex 20E0\string;% combining enclosing circle backslash
12514 \glshex 20E1% combining left right arrow above
12515 }

```

sxtrhyphenrules Hyphens.

```

12516 \newcommand*{\glsxtrhyphenrules}{%
12517 \string'\string-\string'\string;% ASCII hyphen
12518 \glshex 00AD\string;% soft hyphen
12519 \glshex 2010\string;% hyphen
12520 \glshex 2011\string;% non-breaking hyphen
12521 \glshex 2012\string;% figure dash
12522 \glshex 2013\string;% en dash
12523 \glshex 2014\string;% em dash
12524 \glshex 2015\string;% horizontal bar
12525 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
12526 }

```

eneralpuncrules General punctuation.

```

12527 \newcommand*{\glsxtrgeneralpuncrules}{%
12528 \glsxtrgeneralpuncIrules
12529 \string<\glsxtrcurrencyrules
12530 \string<\glsxtrgeneralpuncIIrules
12531 }

```

eneralpuncIrules First set of general punctuation.

```

12532 \newcommand*{\glsxtrgeneralpuncIrules}{%
12533 \string'\glshex 005F\string'% underscore
12534 \string<\glshex 00AF% macron
12535 \string<\string'\glshex 002C\string'% comma
12536 \string<\string'\glshex 003B\string'% semi-colon
12537 \string<\string'\glshex 003A\string'% colon
12538 \string<\string'\glshex 0021\string'% exclamation mark
12539 \string<\glshex 00A1% inverted exclamation mark
12540 \string<\string'\glshex 003F\string'% question mark
12541 \string<\glshex 00BF% inverted question mark
12542 \string<\string'\glshex 002F\string'% solidus
12543 \string<\string'\glshex 002E\string'% full stop
12544 \string<\glshex 00B4% acute accent
12545 \string<\string'\glshex 0060\string'% grave accent

```

```

12546 \string<\string'\glshex 005E\string'% circumflex accent
12547 \string<\glshex 00A8% diaersis
12548 \string<\string'\glshex 007E\string'% tilde
12549 \string<\glshex 00B7% middle dot
12550 \string<\glshex 00B8% cedilla
12551 \string<\string'\glshex 0027\string'% straight apostrophe
12552 \string<\string'\glshex 0022\string'% straight double quote
12553 \string<\glshex 00AB% left guillemet
12554 \string<\glshex 00BB% right guillemet
12555 \string<\string'\glshex 0028\string'% left parenthesis
12556 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
12557 \string<\string'\glshex 0029\string'% right parenthesis
12558 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
12559 \string<\string'\glshex 005B\string'% left square bracket
12560 \string<\string'\glshex 005D\string'% right square bracket
12561 \string<\string'\glshex 007B\string'% left curly bracket
12562 \string<\string'\glshex 007D\string'% right curly bracket
12563 \string<\glshex 00A7% section sign
12564 \string<\glshex 00B6% pilcrow sign
12565 \string<\glshex 00A9% copyright sign
12566 \string<\glshex 00AE% registered sign
12567 \string<\string'\glshex 0040\string'% at sign
12568 }

```

trcurrencyrules General punctuation.

```

12569 \newcommand*{\glxtrcurrencyrules}{%
12570 \glshex 00A4% currency sign
12571 \string<\glshex 0E3F% Thai currency symbol baht
12572 \string<\glshex 00A2% cent sign
12573 \string<\glshex 20A1% colon sign
12574 \string<\glshex 20A2% cruzeiro sign
12575 \string<\string'\glshex 0024\string'% dollar sign
12576 \string<\glshex 20AB% dong sign
12577 \string<\glshex 20AC% euro sign
12578 \string<\glshex 20A3% French franc sign
12579 \string<\glshex 20A4% lira sign
12580 \string<\glshex 20A5% mill sign
12581 \string<\glshex 20A6% naira sign
12582 \string<\glshex 20A7% peseta sign
12583 \string<\glshex 00A3% pound sign
12584 \string<\glshex 20A8% rupee sign
12585 \string<\glshex 20AA% new sheqel sign
12586 \string<\glshex 20A9% won sign
12587 \string<\glshex 00A5% yen sign
12588 }

```

eralpuncIIrules Second set of general punctuation.

```

12589 \newcommand*{\glxtrgeneralpuncIIrules}{%
12590 \string'\glshex 002A\string'% asterisk

```

```

12591 \string<\string'\glshex 005C\string'% backslash
12592 \string<\string'\glshex 0026\string'% ampersand
12593 \string<\string'\glshex 0023\string'% hash sign
12594 \string<\string'\glshex 0025\string'% percent sign
12595 \string<\string'\glshex 002B\string'% plus sign
12596 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12597 \string<\glshex 00B1% plus-minus sign
12598 \string<\glshex 00F7% division sign
12599 \string<\glshex 00D7% multiplication sign
12600 \string<\string'\glshex 003C\string'% less-than sign
12601 \string<\string'\glshex 003D\string'% equals sign
12602 \string<\string'\glshex 003E\string'% greater-than sign
12603 \string<\glshex 00AC% not sign
12604 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12605 \string<\glshex 00A6% broken bar
12606 \string<\glshex 00B0% degree sign
12607 \string<\glshex 00B5% micron sign
12608 }

```

eralLatinIrules Basic Latin alphabet.

```

12609 \newcommand*{\glxtrGeneralLatinIrules}{%
12610 \glxtrLatinA
12611 \string<b,B%
12612 \string<c,C%
12613 \string<d,D%
12614 \string<\glxtrLatinE
12615 \string<f,F%
12616 \string<g,G%
12617 \string<\glxtrLatinH
12618 \string<\glxtrLatinI
12619 \string<j,J%
12620 \string<\glxtrLatinK
12621 \string<\glxtrLatinL
12622 \string<\glxtrLatinM
12623 \string<\glxtrLatinN
12624 \string<\glxtrLatinO
12625 \string<\glxtrLatinP
12626 \string<q,Q%
12627 \string<r,R%
12628 \string<\glxtrLatinS
12629 \string<\glxtrLatinT
12630 \string<u,U%
12631 \string<v,V%
12632 \string<w,W%
12633 \string<\glxtrLatinX
12634 \string<y,Y%
12635 \string<z,Z
12636 }

```

alLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```
12637 \newcommand*{\glxtrGeneralLatinIIrules}{%
12638   \glxtrLatinA
12639   \string<b,B%
12640   \string<c,C%
12641   \string<d,D%
12642   \string<\glxtrLatinEth
12643   \string<\glxtrLatinE
12644   \string<f,F%
12645   \string<g,G%
12646   \string<\glxtrLatinH
12647   \string<\glxtrLatinI
12648   \string<j,J%
12649   \string<\glxtrLatinK
12650   \string<\glxtrLatinL
12651   \string<\glxtrLatinM
12652   \string<\glxtrLatinN
12653   \string<\glxtrLatinO
12654   \string<\glxtrLatinP
12655   \string<q,Q%
12656   \string<r,R%
12657   \string<\glxtrLatinS
12658   \string& SS \string, \glxtrLatinEszettSs
12659   \string<\glxtrLatinT
12660   \string<u,U%
12661   \string<v,V%
12662   \string<w,W%
12663   \string<\glxtrLatinX
12664   \string<y,Y%
12665   \string<z,Z%
12666 }
```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```
12667 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12668   \glxtrLatinA
12669   \string<b,B%
12670   \string<c,C%
12671   \string<d,D%
12672   \string<\glxtrLatinEth
12673   \string<\glxtrLatinE
12674   \string<f,F%
12675   \string<g,G%
12676   \string<\glxtrLatinH
12677   \string<\glxtrLatinI
12678   \string<j,J%
12679   \string<\glxtrLatinK
12680   \string<\glxtrLatinL
12681   \string<\glxtrLatinM
12682   \string<\glxtrLatinN
```

```

12683 \string<\glxtrLatinO
12684 \string<\glxtrLatinP
12685 \string<q,Q%
12686 \string<r,R%
12687 \string<\glxtrLatinS
12688 \string& SZ, \glxtrLatinEszettSz
12689 \string<\glxtrLatinT
12690 \string<u,U%
12691 \string<v,V%
12692 \string<w,W%
12693 \string<\glxtrLatinX
12694 \string<y,Y%
12695 \string<z,Z%
12696 }

```

ralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12697 \newcommand*{\glxtrGeneralLatinIVrules}{%
12698 \glxtrLatinA
12699 \string& AE , \glxtrLatinAELigature
12700 \string<b,B%
12701 \string<c,C%
12702 \string<d,D%
12703 \string<\glxtrLatinEth
12704 \string<\glxtrLatinE
12705 \string<f,F%
12706 \string<g,G%
12707 \string<\glxtrLatinH
12708 \string<\glxtrLatinI
12709 \string<j,J%
12710 \string<\glxtrLatinK
12711 \string<\glxtrLatinL
12712 \string<\glxtrLatinM
12713 \string<\glxtrLatinN
12714 \string<\glxtrLatinO
12715 \string& OE , \glxtrLatinOELigature
12716 \string<\glxtrLatinP
12717 \string<q,Q%
12718 \string<r,R%
12719 \string<\glxtrLatinS
12720 \string& SS , \glxtrLatinEszettSs
12721 \string<\glxtrLatinT
12722 \string& th =\glshex 00DE
12723 \string& TH =\glshex 00FE
12724 \string<u,U%
12725 \string<v,V%
12726 \string<w,W%
12727 \string<\glxtrLatinX
12728 \string<y,Y%

```



```

12729 \string<z,Z%
12730 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12731 \newcommand*{\glxtrGeneralLatinVrules}{%
12732 \glxtrLatinA
12733 \string<b,B%
12734 \string<c,C%
12735 \string<d,D%
12736 \string<\glxtrLatinEth
12737 \string<\glxtrLatinE
12738 \string<f,F%
12739 \string<g,G%
12740 \string<\glxtrLatinH
12741 \string<\glxtrLatinI
12742 \string<j,J%
12743 \string<\glxtrLatinK
12744 \string<\glxtrLatinL
12745 \string<\glxtrLatinM
12746 \string<\glxtrLatinN
12747 \string<\glxtrLatinO
12748 \string<\glxtrLatinP
12749 \string<q,Q%
12750 \string<r,R%
12751 \string<\glxtrLatinS
12752 \string& SS , \glxtrLatinEszettSs
12753 \string<\glxtrLatinT
12754 \string& th =\glshex 00DE
12755 \string& TH =\glshex 00FE
12756 \string<u,U%
12757 \string<v,V%
12758 \string<w,W%
12759 \string<\glxtrLatinX
12760 \string<y,Y%
12761 \string<z,Z%
12762 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12763 \newcommand*{\glxtrGeneralLatinVrules}{%
12764 \glxtrLatinA
12765 \string<b,B%
12766 \string<c,C%
12767 \string<d,D%
12768 \string<\glxtrLatinEth
12769 \string<\glxtrLatinE
12770 \string<f,F%
12771 \string<g,G%
12772 \string<\glxtrLatinH
12773 \string<\glxtrLatinI

```

```

12774 \string<j,J%
12775 \string<\glxtrLatinK
12776 \string<\glxtrLatinL
12777 \string<\glxtrLatinM
12778 \string<\glxtrLatinN
12779 \string<\glxtrLatinO
12780 \string<\glxtrLatinP
12781 \string<q,Q%
12782 \string<r,R%
12783 \string<\glxtrLatinS
12784 \string& SZ , \glxtrLatinEszettSz
12785 \string<\glxtrLatinT
12786 \string& th =\glshex 00DE
12787 \string& TH =\glshex 00FE
12788 \string<u,U%
12789 \string<v,V%
12790 \string<w,W%
12791 \string<\glxtrLatinX
12792 \string<y,Y%
12793 \string<z,Z%
12794 }

```

allLatinVIIrules 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).

```

12795 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12796 \glxtrLatinA
12797 \string<\glxtrLatinAELigature
12798 \string<b,B%
12799 \string<c,C%
12800 \string<d,D%
12801 \string<\glxtrLatinEth
12802 \string<\glxtrLatinE
12803 \string<f,F%
12804 \string<\glxtrLatinInsularG
12805 \string<\glxtrLatinH
12806 \string<\glxtrLatinI
12807 \string<j,J%
12808 \string<\glxtrLatinK
12809 \string<\glxtrLatinL
12810 \string<\glxtrLatinM
12811 \string<\glxtrLatinN
12812 \string<\glxtrLatinO
12813 \string<\glxtrLatinOELigature
12814 \string<\glxtrLatinP
12815 \string<q,Q%
12816 \string<r,R%
12817 \string<\glshex 017F=\glxtrLatinS % s and long s
12818 \string<\glxtrLatinT
12819 \string<\glxtrLatinThorn

```

```

12820 \string<u,U%
12821 \string<v,V%
12822 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12823 \string<\glsxtrLatinX
12824 \string<y,Y%
12825 \string<z,Z%
12826 }

```

LatinVIIIrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

12827 \newcommand*{\glsxtrGeneralLatinVIIIrules}{%
12828 \glsxtrLatinA
12829 \string& AE , \glsxtrLatinAELigature
12830 \string<b,B%
12831 \string<c,C%
12832 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12833 \string<\glsxtrLatinE
12834 \string<f,F%
12835 \string<g,G%
12836 \string<\glsxtrLatinH
12837 \string<\glsxtrLatinI
12838 \string<j,J%
12839 \string<\glsxtrLatinK
12840 \string<\glshex 0142\string=\glsxtrLatinL\string=\glshex 0141% L and \L
12841 \string<\glsxtrLatinM
12842 \string<\glsxtrLatinN
12843 \string<\glshex 00F8\string=\glsxtrLatinO\string=\glshex 00D8% O and \O
12844 \string& OE , \glsxtrLatinOELigature
12845 \string<\glsxtrLatinP
12846 \string<q,Q%
12847 \string<r,R%
12848 \string<\glsxtrLatinS
12849 \string& SS , \glsxtrLatinEszettSs
12850 \string<\glsxtrLatinT
12851 \string& th =\glshex 00DE
12852 \string& TH =\glshex 00FE
12853 \string<u,U%
12854 \string<v,V%
12855 \string<w,W%
12856 \string<\glsxtrLatinX
12857 \string<y,Y%
12858 \string<z,Z%
12859 }

```

\glsxtrLatinA

```

12860 \newcommand*{\glsxtrLatinA}{%
12861 a\string=\glshex 00AA\string=\glshex 2090,A
12862 }

```

\glsxtrLatinE

```
12863 \newcommand*{\glsxtrLatinE}{%
12864   e\string=\glshex 2091,E
12865 }
```

\glsxtrLatinH

```
12866 \newcommand*{\glsxtrLatinH}{%
12867   h\string=\glshex 2095,H
12868 }
```

\glsxtrLatinI

```
12869 \newcommand*{\glsxtrLatinI}{%
12870   i\string=\glshex 2071,I
12871 }
```

\glsxtrLatinK

```
12872 \newcommand*{\glsxtrLatinK}{%
12873   k\string=\glshex 2096,K
12874 }
```

\glsxtrLatinL

```
12875 \newcommand*{\glsxtrLatinL}{%
12876   l\string=\glshex 2097,L
12877 }
```

\glsxtrLatinM

```
12878 \newcommand*{\glsxtrLatinM}{%
12879   m\string=\glshex 2098,M
12880 }
```

\glsxtrLatinN

```
12881 \newcommand*{\glsxtrLatinN}{%
12882   n\string=\glshex 207F\string=\glshex 2099,N
12883 }
```

\glsxtrLatinO

```
12884 \newcommand*{\glsxtrLatinO}{%
12885   o\string=\glshex 00BA\string=\glshex 2092,O
12886 }
```

\glsxtrLatinP

```
12887 \newcommand*{\glsxtrLatinP}{%
12888   p\string=\glshex 209A,P
12889 }
```

\glsxtrLatinS

```
12890 \newcommand*{\glsxtrLatinS}{%
12891   s\string=\glshex 209B,S
12892 }
```

```

\glxtrLatinT
12893 \newcommand*{\glxtrLatinT}{%
12894   t\string=\glshex 209C,T
12895 }

\glxtrLatinX
12896 \newcommand*{\glxtrLatinX}{%
12897   x\string=\glshex 2093,X
12898 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
12899 \newcommand*{\glxtrLatinSchwa}{%
12900   \glshex 0259\string=\glshex 2094,\glshex 018F
12901 }

trLatinEszettSs
12902 \newcommand*{\glxtrLatinEszettSs}{%
12903   \glshex 00DF% eszett
12904   \string=\glshex 017Fs % long S s
12905 }

trLatinEszettSz
12906 \newcommand*{\glxtrLatinEszettSz}{%
12907   \glshex 00DF% eszett
12908   \string= \glshex 017Fz % long S z
12909 }

\glxtrLatinEth
12910 \newcommand*{\glxtrLatinEth}{%
12911   \glshex 00F0,\glshex 00D0% eth
12912 }

lsxtrLatinThorn
12913 \newcommand*{\glxtrLatinThorn}{%
12914   \glshex 00FE,\glshex 00DE% thorn
12915 }

LatinAELigature
12916 \newcommand*{\glxtrLatinAELigature}{%
12917   \glshex 00E6,\glshex 00C6% AE-ligature
12918 }

LatinOELigature
12919 \newcommand*{\glxtrLatinOELigature}{%
12920   \glshex 0153,\glshex 0152% OE-ligature
12921 }

```

\glxtrLatinAA

```
12922 \newcommand*{\glxtrLatinAA}{%
12923   \glshex 00E5=a\glshex 030A,% \aa
12924   \glshex 00C5=A\glshex 030A% \AA
12925 }
```

glxtrLatinWynn

```
12926 \newcommand*{\glxtrLatinWynn}{%
12927   \glshex 01BF,\glshex 01F7% wynn
12928 }
```

trLatinInsularG

```
12929 \newcommand*{\glxtrLatinInsularG}{%
12930   \glshex 1D79,\glshex A77D% insular G
12931   \string; g, G
12932 }
```

sxtrLatinOslash

```
12933 \newcommand*{\glxtrLatinOslash}{%
12934   \glshex 00F8,\glshex 00D8% \o, \O
12935 }
```

sxtrLatinLslash

```
12936 \newcommand*{\glxtrLatinLslash}{%
12937   \glshex 0142,\glshex 0141% \l, \L
12938 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12939 \newcommand*{\glxtrMathUpGreekIrules}{%
12940   \glxtrUpAlpha
12941   \string<\glxtrUpBeta
12942   \string<\glxtrUpGamma
12943   \string<\glxtrUpDelta
12944   \string<\glxtrUpEpsilon
12945   \string<\glxtrUpDigamma
12946   \string<\glxtrUpZeta
12947   \string<\glxtrUpEta
12948   \string<\glxtrUpTheta
12949   \string<\glxtrUpIota
12950   \string<\glxtrUpKappa
12951   \string<\glxtrUpLambda
12952   \string<\glxtrUpMu
12953   \string<\glxtrUpNu
12954   \string<\glxtrUpXi
12955   \string<\glxtrUpOmicron
12956   \string<\glxtrUpPi
12957   \string<\glxtrUpRho
12958   \string<\glxtrUpSigma
```

```

12959 \string<\glxtrUpTau
12960 \string<\glxtrUpUpsilon
12961 \string<\glxtrUpPhi
12962 \string<\glxtrUpChi
12963 \string<\glxtrUpPsi
12964 \string<\glxtrUpOmega
12965 }

```

hUpGreekIIrules Doesn't include digamma.

```

12966 \newcommand*{\glxtrMathUpGreekIIrules}{%
12967 \glxtrUpAlpha
12968 \string<\glxtrUpBeta
12969 \string<\glxtrUpGamma
12970 \string<\glxtrUpDelta
12971 \string<\glxtrUpEpsilon
12972 \string<\glxtrUpZeta
12973 \string<\glxtrUpEta
12974 \string<\glxtrUpTheta
12975 \string<\glxtrUpIota
12976 \string<\glxtrUpKappa
12977 \string<\glxtrUpLambda
12978 \string<\glxtrUpMu
12979 \string<\glxtrUpNu
12980 \string<\glxtrUpXi
12981 \string<\glxtrUpOmicron
12982 \string<\glxtrUpPi
12983 \string<\glxtrUpRho
12984 \string<\glxtrUpSigma
12985 \string<\glxtrUpTau
12986 \string<\glxtrUpUpsilon
12987 \string<\glxtrUpPhi
12988 \string<\glxtrUpChi
12989 \string<\glxtrUpPsi
12990 \string<\glxtrUpOmega
12991 }

```

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.

```

12992 \newcommand*{\glxtrMathItalicGreekIrules}{%
12993 \glxtrMathItalicAlpha
12994 \string<\glxtrMathItalicBeta
12995 \string<\glxtrMathItalicGamma
12996 \string<\glxtrMathItalicDelta
12997 \string<\glxtrMathItalicEpsilon
12998 \string<\glxtrUpDigamma
12999 \string<\glxtrMathItalicZeta
13000 \string<\glxtrMathItalicEta
13001 \string<\glxtrMathItalicTheta
13002 \string<\glxtrMathItalicIota

```

```

13003 \string<\glxtrMathItalicKappa
13004 \string<\glxtrMathItalicLambda
13005 \string<\glxtrMathItalicMu
13006 \string<\glxtrMathItalicNu
13007 \string<\glxtrMathItalicXi
13008 \string<\glxtrMathItalicOmicron
13009 \string<\glxtrMathItalicPi
13010 \string<\glxtrMathItalicRho
13011 \string<\glxtrMathItalicSigma
13012 \string<\glxtrMathItalicTau
13013 \string<\glxtrMathItalicUpsilon
13014 \string<\glxtrMathItalicPhi
13015 \string<\glxtrMathItalicChi
13016 \string<\glxtrMathItalicPsi
13017 \string<\glxtrMathItalicOmega
13018 }

```

LowerGreekIIrules Doesn't include digamma.

```

13019 \newcommand*{\glxtrMathItalicGreekIIrules}{%
13020 \glxtrMathItalicAlpha
13021 \string<\glxtrMathItalicBeta
13022 \string<\glxtrMathItalicGamma
13023 \string<\glxtrMathItalicDelta
13024 \string<\glxtrMathItalicEpsilon
13025 \string<\glxtrMathItalicZeta
13026 \string<\glxtrMathItalicEta
13027 \string<\glxtrMathItalicTheta
13028 \string<\glxtrMathItalicIota
13029 \string<\glxtrMathItalicKappa
13030 \string<\glxtrMathItalicLambda
13031 \string<\glxtrMathItalicMu
13032 \string<\glxtrMathItalicNu
13033 \string<\glxtrMathItalicXi
13034 \string<\glxtrMathItalicOmicron
13035 \string<\glxtrMathItalicPi
13036 \string<\glxtrMathItalicRho
13037 \string<\glxtrMathItalicSigma
13038 \string<\glxtrMathItalicTau
13039 \string<\glxtrMathItalicUpsilon
13040 \string<\glxtrMathItalicPhi
13041 \string<\glxtrMathItalicChi
13042 \string<\glxtrMathItalicPsi
13043 \string<\glxtrMathItalicOmega
13044 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

13045 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
13046 \glshex 1D6E2% upper case alpha (maths italic)
13047 \string<\glshex 1D6E3% upper case beta (maths italic)

```



```

13048 \string<\glshex 1D6E4% upper case gamma (maths italic)
13049 \string<\glshex 1D6E5% upper case delta (maths italic)
13050 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13051 \string<\glshex 03DC% upper case digamma
13052 \string<\glshex 1D6E7% upper case zeta (maths italic)
13053 \string<\glshex 1D6E8% upper case eta (maths italic)
13054 \string<\glshex 1D6E9% upper case theta (maths italic)
13055 \string=<\glshex 1D6F3% upper case theta variant (maths italic)
13056 \string<\glshex 1D6EA% upper case iota (maths italic)
13057 \string<\glshex 1D6EB% upper case kappa (maths italic)
13058 \string<\glshex 1D6EC% upper case lambda (maths italic)
13059 \string<\glshex 1D6ED% upper case mu (maths italic)
13060 \string<\glshex 1D6EE% upper case nu (maths italic)
13061 \string<\glshex 1D6EF% upper case xi (maths italic)
13062 \string<\glshex 1D6F0% upper case omicron (maths italic)
13063 \string<\glshex 1D6F1% upper case pi (maths italic)
13064 \string<\glshex 1D6F2% upper case rho (maths italic)
13065 \string<\glshex 1D6F4% upper case sigma (maths italic)
13066 \string<\glshex 1D6F5% upper case tau (maths italic)
13067 \string<\glshex 1D6F6% upper case upsilon (maths italic)
13068 \string<\glshex 1D6F7% upper case phi (maths italic)
13069 \string<\glshex 1D6F8% upper case chi (maths italic)
13070 \string<\glshex 1D6F9% upper case psi (maths italic)
13071 \string<\glshex 1D6FA% upper case omega (maths italic)
13072 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

13073 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
13074 \glshex 1D6E2% upper case alpha (maths italic)
13075 \string<\glshex 1D6E3% upper case beta (maths italic)
13076 \string<\glshex 1D6E4% upper case gamma (maths italic)
13077 \string<\glshex 1D6E5% upper case delta (maths italic)
13078 \string<\glshex 1D6E6% upper case epsilon (maths italic)
13079 \string<\glshex 1D6E7% upper case zeta (maths italic)
13080 \string<\glshex 1D6E8% upper case eta (maths italic)
13081 \string<\glshex 1D6E9% upper case theta (maths italic)
13082 \string=<\glshex 1D6F3% upper case theta variant (maths italic)
13083 \string<\glshex 1D6EA% upper case iota (maths italic)
13084 \string<\glshex 1D6EB% upper case kappa (maths italic)
13085 \string<\glshex 1D6EC% upper case lambda (maths italic)
13086 \string<\glshex 1D6ED% upper case mu (maths italic)
13087 \string<\glshex 1D6EE% upper case nu (maths italic)
13088 \string<\glshex 1D6EF% upper case xi (maths italic)
13089 \string<\glshex 1D6F0% upper case omicron (maths italic)
13090 \string<\glshex 1D6F1% upper case pi (maths italic)
13091 \string<\glshex 1D6F2% upper case rho (maths italic)
13092 \string<\glshex 1D6F4% upper case sigma (maths italic)
13093 \string<\glshex 1D6F5% upper case tau (maths italic)
13094 \string<\glshex 1D6F6% upper case upsilon (maths italic)

```

```

13095 \string<\glshex 1D6F7% upper case phi (maths italic)
13096 \string<\glshex 1D6F8% upper case chi (maths italic)
13097 \string<\glshex 1D6F9% upper case psi (maths italic)
13098 \string<\glshex 1D6FA% upper case omega (maths italic)
13099 }

```

LowerGreekiRules Lower case only (includes upright digamma).

```

13100 \newcommand*{\glxtrMathItalicLowerGreekiRules}{%
13101 \glshex 1D6FC% lower case alpha (maths italic)
13102 \string<\glshex 1D6FD% lower case beta (maths italic)
13103 \string<\glshex 1D6FE% lower case gamma (maths italic)
13104 \string<\glshex 1D6FF% lower case delta (maths italic)
13105 \string<\glshex 1D700% lower case epsilon (maths italic)
13106 \string<\glshex 1D716% lower case epsilon variant (maths italic)
13107 \string<\glshex 03DD% lower case digamma
13108 \string<\glshex 1D701% lower case zeta (maths italic)
13109 \string<\glshex 1D702% lower case eta (maths italic)
13110 \string<\glshex 1D703% lower case theta (maths italic)
13111 \string<\glshex 1D717% lower case theta variant (maths italic)
13112 \string<\glshex 1D704% lower case iota (maths italic)
13113 \string<\glshex 1D705% lower case kappa (maths italic)
13114 \string<\glshex 1D718% lower case kappa variant (maths italic)
13115 \string<\glshex 1D706% lower case lambda (maths italic)
13116 \string<\glshex 1D707% lower case mu (maths italic)
13117 \string<\glshex 1D708% lower case nu (maths italic)
13118 \string<\glshex 1D709% lower case xi (maths italic)
13119 \string<\glshex 1D70A% lower case omicron (maths italic)
13120 \string<\glshex 1D70B% lower case pi (maths italic)
13121 \string<\glshex 1D71B% lower case pi variant (maths italic)
13122 \string<\glshex 1D70C% lower case rho (maths italic)
13123 \string<\glshex 1D71A% lower case rho variant (maths italic)
13124 \string<\glshex 1D70D% lower case final sigma (maths italic)
13125 \string<\glshex 1D70E% lower case sigma (maths italic)
13126 \string<\glshex 1D70F% lower case tau (maths italic)
13127 \string<\glshex 1D710% lower case upsilon (maths italic)
13128 \string<\glshex 1D711% lower case phi (maths italic)
13129 \string<\glshex 1D719% lower case phi variant (maths italic)
13130 \string<\glshex 1D712% lower case chi (maths italic)
13131 \string<\glshex 1D713% lower case psi (maths italic)
13132 \string<\glshex 1D714% lower case omega (maths italic)
13133 }

```

LowerGreekiRules Lower case only (doesn't includes upright digamma).

```

13134 \newcommand*{\glxtrMathItalicLowerGreekiRules}{%
13135 \glshex 1D6FC% lower case alpha (maths italic)
13136 \string<\glshex 1D6FD% lower case beta (maths italic)
13137 \string<\glshex 1D6FE% lower case gamma (maths italic)
13138 \string<\glshex 1D6FF% lower case delta (maths italic)
13139 \string<\glshex 1D700% lower case epsilon (maths italic)

```

```

13140 \string=\glshex 1D716% lower case epsilon variant (maths italic)
13141 \string<\glshex 1D701% lower case zeta (maths italic)
13142 \string<\glshex 1D702% lower case eta (maths italic)
13143 \string<\glshex 1D703% lower case theta (maths italic)
13144 \string=\glshex 1D717% lower case theta variant (maths italic)
13145 \string<\glshex 1D704% lower case iota (maths italic)
13146 \string<\glshex 1D705% lower case kappa (maths italic)
13147 \string=\glshex 1D718% lower case kappa variant (maths italic)
13148 \string<\glshex 1D706% lower case lambda (maths italic)
13149 \string<\glshex 1D707% lower case mu (maths italic)
13150 \string<\glshex 1D708% lower case nu (maths italic)
13151 \string<\glshex 1D709% lower case xi (maths italic)
13152 \string<\glshex 1D70A% lower case omicron (maths italic)
13153 \string<\glshex 1D70B% lower case pi (maths italic)
13154 \string=\glshex 1D71B% lower case pi variant (maths italic)
13155 \string<\glshex 1D70C% lower case rho (maths italic)
13156 \string=\glshex 1D71A% lower case rho variant (maths italic)
13157 \string<\glshex 1D70D% lower case final sigma (maths italic)
13158 \string=\glshex 1D70E% lower case sigma (maths italic)
13159 \string<\glshex 1D70F% lower case tau (maths italic)
13160 \string<\glshex 1D710% lower case upsilon (maths italic)
13161 \string<\glshex 1D711% lower case phi (maths italic)
13162 \string=\glshex 1D719% lower case phi variant (maths italic)
13163 \string<\glshex 1D712% lower case chi (maths italic)
13164 \string<\glshex 1D713% lower case psi (maths italic)
13165 \string<\glshex 1D714% lower case omega (maths italic)
13166 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

13167 \newcommand*{\glxtrMathGreekIrules}{%
13168 \glxtrMathItalicAlpha
13169 \string;\glxtrUpAlpha
13170 \string<\glxtrMathItalicBeta
13171 \string;\glxtrUpBeta
13172 \string<\glxtrMathItalicGamma
13173 \string;\glxtrUpGamma
13174 \string<\glxtrMathItalicDelta
13175 \string;\glxtrUpDelta
13176 \string<\glxtrMathItalicEpsilon
13177 \string;\glxtrUpEpsilon
13178 \string<\glxtrUpDigamma
13179 \string<\glxtrMathItalicZeta
13180 \string;\glxtrUpZeta
13181 \string<\glxtrMathItalicEta
13182 \string;\glxtrUpEta
13183 \string<\glxtrMathItalicTheta
13184 \string;\glxtrUpTheta
13185 \string<\glxtrMathItalicIota
13186 \string;\glxtrUpIota

```

```

13187 \string<\glxtrMathItalicKappa
13188 \string;\glxtrUpKappa
13189 \string<\glxtrMathItalicLambda
13190 \string;\glxtrUpLambda
13191 \string<\glxtrMathItalicMu
13192 \string;\glxtrUpMu
13193 \string<\glxtrMathItalicNu
13194 \string;\glxtrUpNu
13195 \string<\glxtrMathItalicXi
13196 \string;\glxtrUpXi
13197 \string<\glxtrMathItalicOmicron
13198 \string;\glxtrUpOmicron
13199 \string<\glxtrMathItalicPi
13200 \string;\glxtrUpPi
13201 \string<\glxtrMathItalicRho
13202 \string;\glxtrUpRho
13203 \string<\glxtrMathItalicSigma
13204 \string;\glxtrUpSigma
13205 \string<\glxtrMathItalicTau
13206 \string;\glxtrUpTau
13207 \string<\glxtrMathItalicUpsilon
13208 \string;\glxtrUpUpsilon
13209 \string<\glxtrMathItalicPhi
13210 \string;\glxtrUpPhi
13211 \string<\glxtrMathItalicChi
13212 \string;\glxtrUpChi
13213 \string<\glxtrMathItalicPsi
13214 \string;\glxtrUpPsi
13215 \string<\glxtrMathItalicOmega
13216 \string;\glxtrUpOmega
13217 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

13218 \newcommand*{\glxtrMathGreekIIrules}{%
13219 \glxtrMathItalicAlpha
13220 \string;\glxtrUpAlpha
13221 \string<\glxtrMathItalicBeta
13222 \string;\glxtrUpBeta
13223 \string<\glxtrMathItalicGamma
13224 \string;\glxtrUpGamma
13225 \string<\glxtrMathItalicDelta
13226 \string;\glxtrUpDelta
13227 \string<\glxtrMathItalicEpsilon
13228 \string;\glxtrUpEpsilon
13229 \string<\glxtrMathItalicZeta
13230 \string;\glxtrUpZeta
13231 \string<\glxtrMathItalicEta
13232 \string;\glxtrUpEta
13233 \string<\glxtrMathItalicTheta

```

```

13234 \string;\glxtrUpTheta
13235 \string<\glxtrMathItalicIota
13236 \string;\glxtrUpIota
13237 \string<\glxtrMathItalicKappa
13238 \string;\glxtrUpKappa
13239 \string<\glxtrMathItalicLambda
13240 \string;\glxtrUpLambda
13241 \string<\glxtrMathItalicMu
13242 \string;\glxtrUpMu
13243 \string<\glxtrMathItalicNu
13244 \string;\glxtrUpNu
13245 \string<\glxtrMathItalicXi
13246 \string;\glxtrUpXi
13247 \string<\glxtrMathItalicOmicron
13248 \string;\glxtrUpOmicron
13249 \string<\glxtrMathItalicPi
13250 \string;\glxtrUpPi
13251 \string<\glxtrMathItalicRho
13252 \string;\glxtrUpRho
13253 \string<\glxtrMathItalicSigma
13254 \string;\glxtrUpSigma
13255 \string<\glxtrMathItalicTau
13256 \string;\glxtrUpTau
13257 \string<\glxtrMathItalicUpsilon
13258 \string;\glxtrUpUpsilon
13259 \string<\glxtrMathItalicPhi
13260 \string;\glxtrUpPhi
13261 \string<\glxtrMathItalicChi
13262 \string;\glxtrUpChi
13263 \string<\glxtrMathItalicPsi
13264 \string;\glxtrUpPsi
13265 \string<\glxtrMathItalicOmega
13266 \string;\glxtrUpOmega
13267 }

```

\glxtrUpAlpha

```

13268 \newcommand*{\glxtrUpAlpha}{%
13269 \glshex 03B1,% lower case alpha
13270 \glshex 0391% upper case alpha
13271 }

```

\glxtrUpBeta

```

13272 \newcommand*{\glxtrUpBeta}{%
13273 \glshex 03B2,% lower case beta
13274 \glshex 0392% upper case beta
13275 }

```

\glxtrUpGamma

```

13276 \newcommand*{\glxtrUpGamma}{%

```

```

13277 \glshex 03B3,% lower case gamma
13278 \glshex 0393% upper case gamma
13279 }

```

\glxtrUpDelta

```

13280 \newcommand*{\glxtrUpDelta}{%
13281 \glshex 03B4,% lower case delta
13282 \glshex 0394% upper case delta
13283 }

```

glxtrUpEpsilon

```

13284 \newcommand*{\glxtrUpEpsilon}{%
13285 \glshex 03B5% lower case epsilon
13286 \string=\glshex 03F5,% lower case epsilon variant
13287 \glshex 0395% upper case epsilon
13288 }

```

glxtrUpDigamma

```

13289 \newcommand*{\glxtrUpDigamma}{%
13290 \glshex 03DD,% lower case digamma
13291 \glshex 03DC% upper case digamma
13292 }

```

\glxtrUpZeta

```

13293 \newcommand*{\glxtrUpZeta}{%
13294 \glshex 03B6,% lower case zeta
13295 \glshex 0396% upper case zeta
13296 }

```

\glxtrUpEta

```

13297 \newcommand*{\glxtrUpEta}{%
13298 \glshex 03B7,% lower case eta
13299 \glshex 0397% upper case eta
13300 }

```

\glxtrUpTheta

```

13301 \newcommand*{\glxtrUpTheta}{%
13302 \glshex 03B8% lower case theta
13303 \string=\glshex 03D1,% lower case theta variant
13304 \glshex 0398% upper case theta
13305 }

```

\glxtrUpIota

```

13306 \newcommand*{\glxtrUpIota}{%
13307 \glshex 03B9,% lower case iota
13308 \glshex 0399% upper case iota
13309 }

```

\glxtrUpKappa

```
13310 \newcommand*{\glxtrUpKappa}{%
13311   \glshex 03BA% lower case kappa
13312   \string=\glshex 03F0,% lower case kappa variant
13313   \glshex 039A% upper case kappa
13314 }
```

\glxtrUpLambda

```
13315 \newcommand*{\glxtrUpLambda}{%
13316   \glshex 03BB,% lower lambda
13317   \glshex 039B% upper case lambda
13318 }
```

\glxtrUpMu

```
13319 \newcommand*{\glxtrUpMu}{%
13320   \glshex 03BC,% lower case mu
13321   \glshex 039C% upper case mu
13322 }
```

\glxtrUpNu

```
13323 \newcommand*{\glxtrUpNu}{%
13324   \glshex 03BD,% lower case nu
13325   \glshex 039D% upper case nu
13326 }
```

\glxtrUpXi

```
13327 \newcommand*{\glxtrUpXi}{%
13328   \glshex 03BE,% lower case xi
13329   \glshex 039E% upper case xi
13330 }
```

\glxtrUpOmicron

```
13331 \newcommand*{\glxtrUpOmicron}{%
13332   \glshex 03BF,% lower case omicron
13333   \glshex 039F% upper case omicron
13334 }
```

\glxtrUpPi

```
13335 \newcommand*{\glxtrUpPi}{%
13336   \glshex 03C0% lower case pi
13337   \string=\glshex 03D6,% lower case pi variant
13338   \glshex 03A0% upper case pi
13339 }
```

\glxtrUpRho

```
13340 \newcommand*{\glxtrUpRho}{%
13341   \glshex 03C1% lower case rho
13342   \string=\glshex 03F1,% lower case rho variant
```

```

13343 \glshex 03A1% upper case rho
13344 }

```

\glxtrUpSigma

```

13345 \newcommand*{\glxtrUpSigma}{%
13346 \glshex 03C2% lower case sigma
13347 \string=\glshex 03C3,% lower case sigma
13348 \glshex 03A3% upper case sigma
13349 }

```

\glxtrUpTau

```

13350 \newcommand*{\glxtrUpTau}{%
13351 \glshex 03C4,% lower case tau
13352 \glshex 03A4% upper case tau
13353 }

```

\glxtrUpUpsilon

```

13354 \newcommand*{\glxtrUpUpsilon}{%
13355 \glshex 03C5,% lower case upsilon
13356 \glshex 03A5% upper case upsilon
13357 }

```

\glxtrUpPhi

```

13358 \newcommand*{\glxtrUpPhi}{%
13359 \glshex 03C6% lower case phi
13360 \string=\glshex 03D5,% lower case phi variant
13361 \glshex 03A6% upper case phi
13362 }

```

\glxtrUpChi

```

13363 \newcommand*{\glxtrUpChi}{%
13364 \glshex 03C7,% lower case chi
13365 \glshex 03A7% upper case chi
13366 }

```

\glxtrUpPsi

```

13367 \newcommand*{\glxtrUpPsi}{%
13368 \glshex 03C8,% lower case psi
13369 \glshex 03A8% upper case psi
13370 }

```

\glxtrUpOmega

```

13371 \newcommand*{\glxtrUpOmega}{%
13372 \glshex 03C9,% lower case omega
13373 \glshex 03A9% upper case omega
13374 }

```


MathItalicAlpha

```
13375 \newcommand*{\glxtrMathItalicAlpha}{%  
13376 \glshex 1D6FC,% lower case alpha (maths italic)  
13377 \glshex 1D6E2% upper case alpha (maths italic)  
13378 }
```

rMathItalicBeta

```
13379 \newcommand*{\glxtrMathItalicBeta}{%  
13380 \glshex 1D6FD,% lower case beta (maths italic)  
13381 \glshex 1D6E3% upper case beta (maths italic)  
13382 }
```

MathItalicGamma

```
13383 \newcommand*{\glxtrMathItalicGamma}{%  
13384 \glshex 1D6FE,% lower case gamma (maths italic)  
13385 \glshex 1D6E4% upper case gamma (maths italic)  
13386 }
```

MathItalicDelta

```
13387 \newcommand*{\glxtrMathItalicDelta}{%  
13388 \glshex 1D6FF,% lower case delta (maths italic)  
13389 \glshex 1D6E5% upper case delta (maths italic)  
13390 }
```

thItalicEpsilon

```
13391 \newcommand*{\glxtrMathItalicEpsilon}{%  
13392 \glshex 1D700% lower case epsilon (maths italic)  
13393 \string=\glshex 1D716,% lower case epsilon variant (maths italic)  
13394 \glshex 1D6E6% upper case epsilon (maths italic)  
13395 }
```

rMathItalicZeta

```
13396 \newcommand*{\glxtrMathItalicZeta}{%  
13397 \glshex 1D701,% lower case zeta (maths italic)  
13398 \glshex 1D6E7% upper case zeta (maths italic)  
13399 }
```

trMathItalicEta

```
13400 \newcommand*{\glxtrMathItalicEta}{%  
13401 \glshex 1D702,% lower case eta (maths italic)  
13402 \glshex 1D6E8% upper case eta (maths italic)  
13403 }
```

MathItalicTheta

```
13404 \newcommand*{\glxtrMathItalicTheta}{%  
13405 \glshex 1D703% lower case theta (maths italic)  
13406 \string=\glshex 1D717,% lower case theta variant (maths italic)  
13407 \glshex 1D6E9% upper case theta (maths italic)
```

```

13408 \string=\glshex 1D6F3% upper case theta variant (maths italic)
13409 }

```

MathItalicIota

```

13410 \newcommand*{\glxtrMathItalicIota}{%
13411 \glshex 1D704,% lower case iota (maths italic)
13412 \glshex 1D6EA% upper case iota (maths italic)
13413 }

```

MathItalicKappa

```

13414 \newcommand*{\glxtrMathItalicKappa}{%
13415 \glshex 1D705,% lower case kappa (maths italic)
13416 \string=\glshex 1D718,% lower case kappa variant (maths italic)
13417 \glshex 1D6EB% upper case kappa (maths italic)
13418 }

```

MathItalicLambda

```

13419 \newcommand*{\glxtrMathItalicLambda}{%
13420 \glshex 1D706,% lower case lambda (maths italic)
13421 \glshex 1D6EC% upper case lambda (maths italic)
13422 }

```

xtrMathItalicMu

```

13423 \newcommand*{\glxtrMathItalicMu}{%
13424 \glshex 1D707,% lower case mu (maths italic)
13425 \glshex 1D6ED% upper case mu (maths italic)
13426 }

```

xtrMathItalicNu

```

13427 \newcommand*{\glxtrMathItalicNu}{%
13428 \glshex 1D708,% lower case nu (maths italic)
13429 \glshex 1D6EE% upper case nu (maths italic)
13430 }

```

xtrMathItalicXi

```

13431 \newcommand*{\glxtrMathItalicXi}{%
13432 \glshex 1D709,% lower case xi (maths italic)
13433 \glshex 1D6EF% upper case xi (maths italic)
13434 }

```

thItalicOmicron

```

13435 \newcommand*{\glxtrMathItalicOmicron}{%
13436 \glshex 1D70A,% lower case omicron (maths italic)
13437 \glshex 1D6F0% upper case omicron (maths italic)
13438 }

```

xtrMathItalicPi

```

13439 \newcommand*{\glxtrMathItalicPi}{%

```

```

13440 \glshex 1D70B% lower case pi (maths italic)
13441 \string=\glshex 1D71B,% lower case pi variant (maths italic)
13442 \glshex 1D6F1% upper case pi (maths italic)
13443 }

```

trMathItalicRho

```

13444 \newcommand*{\glsxtrMathItalicRho}{%
13445 \glshex 1D70C% lower case rho (maths italic)
13446 \string=\glshex 1D71A,% lower case rho variant (maths italic)
13447 \glshex 1D6F2% upper case rho (maths italic)
13448 }

```

MathItalicSigma

```

13449 \newcommand*{\glsxtrMathItalicSigma}{%
13450 \glshex 1D70D% lower case final sigma (maths italic)
13451 \string=\glshex 1D70E,% lower case sigma (maths italic)
13452 \glshex 1D6F4% upper case sigma (maths italic)
13453 }

```

trMathItalicTau

```

13454 \newcommand*{\glsxtrMathItalicTau}{%
13455 \glshex 1D70F,% lower case tau (maths italic)
13456 \glshex 1D6F5% upper case tau (maths italic)
13457 }

```

thItalicUpsilon

```

13458 \newcommand*{\glsxtrMathItalicUpsilon}{%
13459 \glshex 1D710,% lower case upsilon (maths italic)
13460 \glshex 1D6F6% upper case upsilon (maths italic)
13461 }

```

trMathItalicPhi

```

13462 \newcommand*{\glsxtrMathItalicPhi}{%
13463 \glshex 1D711% lower case phi (maths italic)
13464 \string=\glshex 1D719,% lower case phi variant (maths italic)
13465 \glshex 1D6F7% upper case phi (maths italic)
13466 }

```

trMathItalicChi

```

13467 \newcommand*{\glsxtrMathItalicChi}{%
13468 \glshex 1D712,% lower case chi (maths italic)
13469 \glshex 1D6F8% upper case chi (maths italic)
13470 }

```

trMathItalicPsi

```

13471 \newcommand*{\glsxtrMathItalicPsi}{%
13472 \glshex 1D713,% lower case psi (maths italic)
13473 \glshex 1D6F9% upper case psi (maths italic)
13474 }

```

MathItalicOmega

```
13475 \newcommand*{\glxtrMathItalicOmega}{%
13476   \glshex 1D714,% lower case omega (maths italic)
13477   \glshex 1D6FA% upper case omega (maths italic)
13478 }
```

thItalicPartial

```
13479 \newcommand*{\glxtrMathItalicPartial}{%
13480   \glshex 1D715% partial differential (maths italic)
13481 }
```

MathItalicNabla

```
13482 \newcommand*{\glxtrMathItalicNabla}{%
13483   \glshex 1D6FB% nabla (maths italic)
13484 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
13485 \newcommand*{\glxtrdigitrules}{%
13486   0\string=\glshex 2080\string=\glshex 2070
13487   \string<1\string=\glshex 2081\string=\glshex 00B9
13488   \string<2\string=\glshex 2082\string=\glshex 00B2
13489   \string<3\string=\glshex 2083\string=\glshex 00B3
13490   \string<4\string=\glshex 2084\string=\glshex 2074
13491   \string<5\string=\glshex 2085\string=\glshex 2075
13492   \string<6\string=\glshex 2086\string=\glshex 2076
13493   \string<7\string=\glshex 2087\string=\glshex 2077
13494   \string<8\string=\glshex 2088\string=\glshex 2078
13495   \string<9\string=\glshex 2089\string=\glshex 2079
13496 }
```

BasicDigitrules Digits from the Basic Latin set.

```
13497 \newcommand*{\glxtrBasicDigitrules}{%
13498   0\string<1\string<2\string<3\string<4%
13499   \string<5\string<6\string<7\string<8\string<9%
13500 }
```

criptDigitrules Subscript digits.

```
13501 \newcommand*{\glxtrSubScriptDigitrules}{%
13502   \glshex 2080% subscript 0
13503   \string<\glshex 2081% subscript 1
13504   \string<\glshex 2082% subscript 2
13505   \string<\glshex 2083% subscript 3
13506   \string<\glshex 2084% subscript 4
13507   \string<\glshex 2085% subscript 5
13508   \string<\glshex 2086% subscript 6
13509   \string<\glshex 2087% subscript 7
13510   \string<\glshex 2088% subscript 8
13511   \string<\glshex 2089% subscript 9
13512 }
```

criptDigitrules Superscript digits.

```
13513 \newcommand*{\glxtrSuperScriptDigitrules}{%
13514 \glshex 2070% superscript 0
13515 \string<\glshex 00B9% superscript 1
13516 \string<\glshex 00B2% superscript 2
13517 \string<\glshex 00B3% superscript 3
13518 \string<\glshex 2074% superscript 4
13519 \string<\glshex 2075% superscript 5
13520 \string<\glshex 2076% superscript 6
13521 \string<\glshex 2077% superscript 7
13522 \string<\glshex 2078% superscript 8
13523 \string<\glshex 2079% superscript 9
13524 }
```

trfractionrules Vulgar fractions.

```
13525 \newcommand*{\glxtrfractionrules}{%
13526 \glshex 215F% fraction numerator one (1/)
13527 \string<\glshex 2189% zero thirds (0/3 = 0)
13528 \string<\glshex 2152% one tenth (1/10 = 0.1)
13529 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
13530 \string<\glshex 215B% one eighth (1/8 = 0.125)
13531 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
13532 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
13533 \string<\glshex 2155% one fifth (1/5 = 0.2)
13534 \string<\glshex 00BC% one quarter (1/4 = 0.25)
13535 \string<\glshex 2153% one third (1/3 ~ 0.333)
13536 \string<\glshex 215C% three eighths (3/8 = 0.375)
13537 \string<\glshex 2156% two fifths (2/5 = 0.4)
13538 \string<\glshex 00BD% one half (1/2 = 0.5)
13539 \string<\glshex 2157% three fifths (3/5 = 0.6)
13540 \string<\glshex 215D% five eighths (5/8 = 0.625)
13541 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
13542 \string<\glshex 00BE% three quarters (3/4 = 0.75)
13543 \string<\glshex 2158% four fifths (4/5 = 0.8)
13544 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
13545 \string<\glshex 215E% seven eighths (7/8 = 0.875)
13546 }
```

sxtrdialecthook Check for scripts associated with the document dialects.

```
13547 \renewcommand{\@glxtrdialecthook}{%
13548 \ifundef\CurrentTrackedScript
13549 {%
13550 \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13551 {%
13552 \edef\CurrentTrackedScript{%
13553 \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13554 }%
13555 {}%
13556 }%
```

```

13557 {}%
13558 \ifdef\CurrentTrackedScript
13559 {%
13560   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
13561   \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
13562   \let\CurrentTrackedTag\CurrentTrackedScript
13563   \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.1df}
13564   {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13565   {}%
13566   \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13567 }%
13568 {}%
13569 }

```

If `\glsxtr@loaddialect` has been defined, then `glossaries-extra-bib2gls` has been loaded after `glossaries-extra`. (For example, through `\glossariesextrasetup`.) Not recommended, but if this has been done try to find the associated language resources.

```

13570 \ifdef\glsxtr@loaddialect
13571 {%
13572   \@ifpackageloaded{tracklang}
13573   {%
13574     \AnyTrackedLanguages
13575     {%
13576       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13577     }%
13578   }%
13579 }
13580 {}
13581 }
13582 {}

```

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:

```
13583 \NeedsTeXFormat{LaTeX2e}
13584 \ProvidesPackage{glossaries-extra-stylemods}[2018/12/01 1.38 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-⟨option⟩.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glxtr@loadstyles`.

`sxtr@loadstyles`

```
13585 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13586 \DeclareOption{all}{%
13587   \appto\@glxtr@loadstyles{%
13588     \RequirePackage{glossary-inline}%
13589     \RequirePackage{glossary-list}%
13590     \RequirePackage{glossary-tree}%
13591     \RequirePackage{glossary-mcols}%
13592     \RequirePackage{glossary-long}%
13593     \RequirePackage{glossary-longragged}%
13594     \RequirePackage{glossary-longbooktabs}%
13595     \RequirePackage{glossary-super}%
13596     \RequirePackage{glossary-superragged}%
13597     \RequirePackage{glossary-bookindex}%
13598     \RequirePackage{glossary-longextra}%
13599   }
13600 }

13601 \DeclareOption*{%
13602   \IfFileExists{glossary-\CurrentOption.sty}
13603   {\eappto\@glxtr@loadstyles{%
13604     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13605   }%
13606   {%
```

```

13607     \PackageError{glossaries-extra-styles}%
13608     {Unknown option ‘\CurrentOption’}{}%
13609   }%
13610 }

```

Process the package options:

```
13611 \ProcessOptions
```

Load the required packages:

```
13612 \@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.

```
13613 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

ewglossarystyle

```

13614 \providecommand{\renewglossarystyle}[2]{%
13615   \ifcsundef{@glsstyle@#1}%
13616   {%
13617     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13618   }%
13619   {%
13620     \csdef{@glsstyle@#1}{#2}%
13621   }%
13622 }

```

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.

```

13623 \ifdef{\@glsstyle@listdotted}
13624 {%
13625   \renewglossarystyle{listdotted}{%
13626     \setglossarystyle{list}%
13627     \renewcommand*{\glossentry}[2]{%
13628       \item[]\makebox[\glslistdottedwidth][l]{%
13629         \glsentryitem{##1}%
13630         \glstarget{##1}{\glossentryname{##1}}%
13631         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13632         \glossentrydesc{##1}\glspostdescription}%
13633     \renewcommand*{\subglossentry}[3]{%
13634       \item[]\makebox[\glslistdottedwidth][l]{%
13635         \glssubentryitem{##2}%
13636         \glstarget{##2}{\glossentryname{##2}}%
13637         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13638         \glossentrydesc{##2}\glspostdescription}%

```



```

13639 }
13640 }
13641 {%

```

Assume the style isn't required if it hasn't already been defined.

```
13642 }
```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

The other list styles would be easier to adapt if the space before the number list wasn't hard coded.

```

13643 \ifdef{\@glsstyle@list}
13644 {%

```

listprelocation Space before number list for top-level entries.

```
13645 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13646 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13647 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13648 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```

13649 \renewglossarystyle{list}{%
13650   \renewenvironment{theglossary}%
13651     {\begin{description}}{\end{description}}%
13652   \renewcommand*{\glossaryheader}{}%
13653   \renewcommand*{\glsgroupheading}[1]{}%
13654   \renewcommand*{\glossentry}[2]{%
13655     \item[\glssentryitem{##1}]%
13656       \glstarget{##1}{\glossentryname{##1}}]
13657     \glslistdesc{##1}\glslistprelocation ##2}%
13658   \renewcommand*{\subglossentry}[3]{%
13659     \glssubentryitem{##2}%
13660     \glstarget{##2}{\strut}\space
13661     \glslistdesc{##2}%
13662     \glslistchildprelocation ##3\glslistchildpostlocation}%
13663   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13664 }
13665 }
13666 {}

```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```

13667 \ifdef{\@glsstyle@altlist}
13668 {%

```

```

13669 \renewglossarystyle{altlist}{%
13670   \setglossarystyle{list}%
13671   \renewcommand*{\glossentry}[2]{%
13672     \item[\glsentryitem{##1}%
13673       \glstarget{##1}{\glossentryname{##1}}]%
13674     \mbox{}\par\nobreak\@afterheading
13675     \glslistdesc{##1}\glslistprelocation ##2}%
13676   \renewcommand{\subglossentry}[3]{%
13677     \par
13678     \glssubentryitem{##2}%
13679     \glstarget{##2}{\strut}\glslistdesc{##2}%
13680     \glslistchildprelocation ##3}%
13681   }
13682 }
13683 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13684 \ifdef{\@glsstyle@listgroup}
13685 {%
13686   \renewglossarystyle{listgroup}{%
13687     \setglossarystyle{list}%
13688     \renewcommand*{\glsgroupheading}[1]{%
13689       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13690       \mbox{}\par\nobreak\@afterheading
13691     }%
13692   }
13693 }
13694 {}

```

Similarly for listhypergroup.

```

13695 \ifdef{\@glsstyle@listhypergroup}
13696 {%
13697   \renewglossarystyle{listhypergroup}{%
13698     \setglossarystyle{list}%
13699     \renewcommand*{\glossaryheader}{%
13700       \glslistnavigationitem{\glsnavigation}}%
13701     \renewcommand*{\glsgroupheading}[1]{%
13702       \item[\glslistgroupheaderfmt
13703         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13704       \mbox{}\par\nobreak\@afterheading
13705     }%
13706   }
13707 }
13708 {}

```

Similarly for altlistgroup.

```

13709 \ifdef{\@glsstyle@altlistgroup}
13710 {%
13711   \renewglossarystyle{altlistgroup}{%
13712     \setglossarystyle{altlist}%
13713     \renewcommand*{\glsgroupheading}[1]{%

```

```

13714     \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13715     \mbox{}\par\nobreak\@afterheading
13716 }%
13717 }
13718 }
13719 {}

```

Similarly for `altlisthypergroup`.

```

13720 \ifdef{\@glsstyle@altlisthypergroup}
13721 {%
13722   \renewglossarystyle{altlisthypergroup}{%
13723     \setglossarystyle{altlist}%
13724     \renewcommand*\glossaryheader{%
13725       \glslistnavigationitem{\glsnavigation}}%
13726     \renewcommand*\glsgroupheading[1]{%
13727       \item[\glslistgroupheaderfmt
13728         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13729       \mbox{}\par\nobreak\@afterheading
13730     }%
13731   }
13732 }
13733 {}

```

2.3 Longtable Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glstrprelocation`.

```

13734 \ifcsdef{@glsstyle@long}
13735 {%
13736   \renewglossarystyle{long}{%
13737     \renewenvironment{theglossary}%
13738       {\begin{longtable}{lp{\glsdescwidth}}}%
13739       {\end{longtable}}%
13740     \renewcommand*\glossaryheader{}%
13741     \renewcommand*\glsgroupheading[1]{}%
13742     \renewcommand{\glossentry}[2]{%
13743       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13744       \glossentrydesc{##1}\glspostdescription
13745       \glstrprelocation ##2\tabularnewline
13746     }%
13747     \renewcommand{\subglossentry}[3]{%
13748       &
13749       \glssubentryitem{##2}%
13750       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13751       \glstrprelocation ##3\tabularnewline
13752     }%
13753     \ifglsnogroupskip
13754       \renewcommand*\glsgroupskip{}%

```

```

13755 \else
13756 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13757 \fi
13758 }
13759 }
13760 {}

```

Three column style:

```

13761 \ifcsdef{@glsstyle@long3col}
13762 {%
13763 \renewglossarystyle{long3col}{%
13764 \renewenvironment{theglossary}%
13765 {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13766 {\end{longtable}}}%
13767 \renewcommand*{\glossaryheader}{}%
13768 \renewcommand*{\glsgroupheading}[1]{}%
13769 \renewcommand{\glossentry}[2]{%
13770 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13771 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13772 }%
13773 \renewcommand{\subglossentry}[3]{%
13774 &
13775 \glssubentryitem{##2}%
13776 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13777 ##3\tabularnewline
13778 }%

```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```

13779 \ifglsnogroupskip
13780 \renewcommand*{\glsgroupskip}{}%
13781 \else
13782 \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13783 \fi
13784 }
13785 }
13786 {}

```

Four column style:

```

13787 \ifcsdef{@glsstyle@long4col}
13788 {%
13789 \renewglossarystyle{long4col}{%
13790 \renewenvironment{theglossary}%
13791 {\begin{longtable}{llll}}}%
13792 {\end{longtable}}}%
13793 \renewcommand*{\glossaryheader}{}%
13794 \renewcommand*{\glsgroupheading}[1]{}%
13795 \renewcommand{\glossentry}[2]{%
13796 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13797 \glossentrydesc{##1}\glspostdescription &
13798 \glossentrysymbol{##1} &

```

```

13799      ##2\tabularnewline
13800    }%
13801    \renewcommand{\subglossentry}[3]{%
13802      &
13803      \glssubentryitem{##2}%
13804      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13805      \glossentrysymbol{##2} & ##3\tabularnewline
13806    }%

13807    \ifglsnogroupskip
13808      \renewcommand*\{glsgroupskip\}{}%
13809    \else
13810      \renewcommand*\{glsgroupskip\}{& & \tabularnewline}%
13811    \fi
13812  }
13813 }
13814 {}

```

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`.

```

13815 \ifcsdef{@glstyle@longragged}
13816 {%
13817   \renewglossarystyle{longragged}{%
13818     \renewenvironment{theglossary}%
13819       {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%
13820       {\end{longtable}}%
13821     \renewcommand*\{glossaryheader\}{}%
13822     \renewcommand*\{glsgroupheading}[1]{}%
13823     \renewcommand{\glossentry}[2]{%
13824       \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13825       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13826       \tabularnewline
13827     }%
13828     \renewcommand{\subglossentry}[3]{%
13829       &
13830       \glssubentryitem{##2}%
13831       \glstarget{##2}{\strut}\glossentrydesc{##2}%
13832       \glspostdescription\glxtrprelocation ##3%
13833       \tabularnewline
13834     }%
13835     \ifglsnogroupskip
13836       \renewcommand*\{glsgroupskip\}{}%
13837     \else

```

```

13838     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13839     \fi
13840   }
13841 }
13842 {}

```

Three and four column styles don't use `\glsxtrprelocation` since the number list is in its own column.

```

13843 \ifcsdef{@glsstyle@longragged3col}
13844 {%
13845   \renewglossarystyle{longragged3col}{%
13846     \renewenvironment{theglossary}%
13847       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13848         >{\raggedright}p{\glspagelistwidth}}}%
13849       {\end{longtable}}}%
13850   \renewcommand*{\glossaryheader}{}%
13851   \renewcommand*{\glsgroupheading}[1]{}%
13852   \renewcommand{\glossentry}[2]{%
13853     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13854     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13855   }%
13856   \renewcommand{\subglossentry}[3]{%
13857     &
13858     \glssubentryitem{##2}%
13859     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13860     ##3\tabularnewline
13861   }%

13862   \ifglsnogroupskip
13863     \renewcommand*{\glsgroupskip}{}%
13864   \else
13865     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13866   \fi
13867 }
13868 }
13869 {}

```

Four column style:

```

13870 \ifcsdef{@glsstyle@altlongragged4col}
13871 {%
13872   \renewglossarystyle{altlongragged4col}{%
13873     \renewenvironment{theglossary}%
13874       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13875         >{\raggedright}p{\glspagelistwidth}}}%
13876       {\end{longtable}}}%
13877   \renewcommand*{\glossaryheader}{}%
13878   \renewcommand*{\glsgroupheading}[1]{}%
13879   \renewcommand{\glossentry}[2]{%
13880     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13881     \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &

```

```

13882     ##2\tabularnewline
13883 }%
13884 \renewcommand{\subglossentry}[3]{%
13885     &
13886     \glssubentryitem{##2}%
13887     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13888     \glossentrysymbol{##2} & ##3\tabularnewline
13889 }%

13890 \ifglsnogroupskip
13891     \renewcommand*{\glsgroupskip}{}%
13892 \else
13893     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13894 \fi
13895 }
13896 }
13897 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded \space changed to \glxtrprelocation.

```

13898 \ifcsdef{@glstyle@super}
13899 {%
13900     \renewglossarystyle{super}{%
13901         \renewenvironment{theglossary}%
13902             {\tablehead{}\tabletail}%
13903             \begin{supertabular}{lp{\glsdescwidth}}%
13904             {\end{supertabular}}%
13905         \renewcommand*{\glossaryheader}{}%
13906         \renewcommand*{\glsgroupheading}[1]{}%
13907         \renewcommand{\glossentry}[2]{%
13908             \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13909             \glossentrydesc{##1}\glspostdescription
13910             \glxtrprelocation ##2\tabularnewline
13911         }%
13912         \renewcommand{\subglossentry}[3]{%
13913             &
13914             \glssubentryitem{##2}%
13915             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13916             \glxtrprelocation ##3\tabularnewline
13917         }%
13918         \ifglsnogroupskip
13919             \renewcommand*{\glsgroupskip}{}%
13920         \else
13921             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13922         \fi
13923     }

```

13924 }
 13925 {}

Three column style:

```

13926 \ifcsdef{@glsstyle@super3col}
13927 {%
13928   \renewglossarystyle{super3col}{%
13929     \renewenvironment{theglossary}%
13930       {\tablehead{}\tabletail{}}%
13931       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13932     {\end{supertabular}}%
13933     \renewcommand*{\glossaryheader}{}%
13934     \renewcommand*{\glsgroupheading}[1]{}%
13935     \renewcommand{\glossentry}[2]{%
13936       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13937       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13938     }%
13939     \renewcommand{\subglossentry}[3]{%
13940       &
13941       \glssubentryitem{##2}%
13942       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13943       ##3\tabularnewline
13944     }%

13945   \ifglsnogroupskip
13946     \renewcommand*{\glsgroupskip}{}%
13947   \else
13948     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13949   \fi
13950 }
13951 }
13952 {}

```

Four column styles:

```

13953 \ifcsdef{@glsstyle@super4col}
13954 {%
13955   \renewglossarystyle{super4col}{%
13956     \renewenvironment{theglossary}%
13957       {\tablehead{}\tabletail{}}%
13958       \begin{supertabular}{llll}}}%
13959     \end{supertabular}}%
13960     \renewcommand*{\glossaryheader}{}%
13961     \renewcommand*{\glsgroupheading}[1]{}%
13962     \renewcommand{\glossentry}[2]{%
13963       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13964       \glossentrydesc{##1}\glspostdescription &
13965       \glossentrysymbol{##1} & ##2\tabularnewline
13966     }%
13967     \renewcommand{\subglossentry}[3]{%
13968       &

```



```

13969     \glssubentryitem{##2}%
13970     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13971     \glossentrysymbol{##2} & ##3\tabularnewline
13972 }%

13973 \ifglsnogroupskip
13974     \renewcommand*{\glsgroupskip}{}%
13975 \else
13976     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13977 \fi
13978 }
13979 }
13980 {}

```

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 `\glxstrprelocation`.

```

13981 \ifcsdef{@glstyle@superragged}
13982 {%
13983     \renewglossarystyle{superragged}{%
13984         \renewenvironment{theglossary}%
13985             {\tablehead{}}\tabletail{}}%
13986         \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
13987         {\end{supertabular}}%
13988     \renewcommand*{\glossaryheader}{}%
13989     \renewcommand*{\glsgroupheading}[1]{}%
13990     \renewcommand{\glossentry}[2]{%
13991         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13992         \glossentrydesc{##1}\glspostdescription\glxstrprelocation ##2%
13993         \tabularnewline
13994     }%
13995     \renewcommand{\subglossentry}[3]{%
13996         &
13997         \glssubentryitem{##2}%
13998         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13999         \glxstrprelocation ##3%
14000         \tabularnewline
14001     }%
14002     \ifglsnogroupskip
14003         \renewcommand*{\glsgroupskip}{}%
14004     \else
14005         \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14006     \fi
14007 }
14008 }
14009 {}

```

Three column style:

```

14010 \ifcsdef{@glsstyle@superragged3col}
14011 {%
14012   \renewglossarystyle{superragged3col}{%
14013     \renewenvironment{theglossary}%
14014       {\tablehead{}\tabletail{}}%
14015       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
14016         >{\raggedright}p{\glspagelistwidth}}}%
14017       {\end{supertabular}}}%
14018   \renewcommand*{\glossaryheader}{}%
14019   \renewcommand*{\glsgroupheading}[1]{}%
14020   \renewcommand{\glossentry}[2]{%
14021     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14022     \glossentrydesc{##1}\glspostdescription &
14023     ##2\tabularnewline
14024   }%
14025   \renewcommand{\subglossentry}[3]{%
14026     &
14027     \glssubentryitem{##2}%
14028     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14029     ##3\tabularnewline
14030   }%

14031   \ifglsnogroupskip
14032     \renewcommand*{\glsgroupskip}{}%
14033   \else
14034     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14035   \fi
14036 }
14037 }
14038 {}

```

Four columns:

```

14039 \ifcsdef{@glsstyle@altsuperragged4col}
14040 {%
14041   \renewglossarystyle{altsuperragged4col}{%
14042     \renewenvironment{theglossary}%
14043       {\tablehead{}\tabletail{}}%
14044       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
14045         >{\raggedright}p{\glspagelistwidth}}}%
14046       {\end{supertabular}}}%
14047   \renewcommand*{\glossaryheader}{}%
14048   \renewcommand{\glossentry}[2]{%
14049     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14050     \glossentrydesc{##1}\glspostdescription &
14051     \glossentrysymbol{##1} & ##2\tabularnewline
14052   }%
14053   \renewcommand{\subglossentry}[3]{%
14054     &
14055     \glssubentryitem{##2}%

```

```

14056      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14057      \glossentrysymbol{##2} & ##3\tabularnewline
14058    }%

14059    \ifglsnogroupskip
14060      \renewcommand*{\glsgroupskip}{}%
14061    \else
14062      \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
14063    \fi
14064  }
14065 }
14066 {}

```

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`.

```

14067 \ifdef{\@glsstyle@inline}
14068 {%
14069   \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}

   Just use \glsxtrpostdescription instead of \glspostdescription.

14070   \renewcommand*{\glsinlinedescformat}[3]{%
14071     \space#1\glsxtrpostdescription}
14072   \renewcommand*{\glsinlinesubdescformat}[3]{%
14073     #1\glsxtrpostdescription}

```

The default settings don't show the location lists, so there's no adjustment for `\glsxtrprelocation`.

```

14074 }
14075 {}

```

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.

```

14076 \ifdef\glstreenamefmt
14077 {

```

```

\glstreedefaultnamefmt

```

```

14078   \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}

```

```

\glstreenamefmt

```

```

14079   \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}

```

egroupheaderfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14080 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

eenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
14081 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

```
14082 }
```

```
14083 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
14084 \ifdef{\@glsstyle@index}
```

```
14085 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
14086 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
14087 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
14088 \renewglossarystyle{index}{%
```

```
14089 \renewenvironment{theglossary}%
```

```
14090 {\setlength{\parindent}{0pt}%
```

```
14091 \setlength{\parskip}{0pt plus 0.3pt}%
```

```
14092 \let\item\glstreeitem
```

```
14093 \let\subitem\glstreesubitem
```

```
14094 \let\subsubitem\glstreesubsubitem
```

```
14095 }%
```

```
14096 {\par}%
```

```
14097 \renewcommand*{\glossaryheader}{}%
```

```
14098 \renewcommand*{\glsgroupheading}[1]{}%
```

```
14099 \renewcommand*{\glossentry}[2]{}%
```

```
14100 \item\glssentryitem{##1}%
```

```
14101 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```
14102 \glstreesymbol{##1}%
```

```
14103 \glstreedesc{##1}%
```

```
14104 \glstreeprelocation ##2%
```

```
14105 }%
```

```
14106 \renewcommand{\subglossentry}[3]{}%
```

```
14107 \ifcase##1\relax
```

```
14108 \item
```

```
14109 \or
```

```
14110 \subitem
```

```
14111 \glssubentryitem{##2}%
```

```
14112 \else
```

```
14113 \subsubitem
```

```
14114 \fi
```

```
14115 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
```

```
14116 \glstreechildsymbol{##2}%
```

```

14117      \glstreechilddesc{##2}%
14118      \glstreechildprelocation ##3%
14119  }%
14120  \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
14121 }
14122 }
14123 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

14124 \ifdef{\@glsstyle@indexgroup}
14125 {%
14126   \renewglossarystyle{indexgroup}{%
14127     \setglossarystyle{index}%
14128     \renewcommand*{\glsgroupheading}[1]{%
14129       \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
14130       \nopagebreak\indexspace
14131       \nobreak\@afterheading
14132     }%
14133   }
14134 }
14135 {}

```

Similarly for indexhypergroup.

```

14136 \ifdef{\@glsstyle@indexhypergroup}
14137 {%
14138   \renewglossarystyle{indexhypergroup}{%
14139     \setglossarystyle{index}%
14140     \renewcommand*{\glossaryheader}{%
14141       \item\glstreenavigationfmt{\glsgroupnavigation}%
14142       \nobreak\@afterheading\indexspace}%
14143     \renewcommand*{\glsgroupheading}[1]{%
14144       \item\glstreegroupheaderfmt
14145         {\glsgroupnavhypertarget{##1}{\glsgrouptitle{##1}}}%
14146       \nopagebreak\indexspace
14147       \nobreak\@afterheading}%
14148   }%
14149 }
14150 {}

```

Adjust tree style to remove hard coded space before number list.

```

14151 \ifdef{\@glsstyle@tree}
14152 {%

```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```

14153   \newcommand{\glstreedesc}[1]{%
14154     \glstreepredesc\glossentrydesc{##1}\glspostdescription
14155   }

```

Similarly for the symbol.

`\glstreesymbol`

```
14156 \newcommand{\glstreesymbol}[1]{%
14157   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
14158 }%
```

And for the child entries:

`lstreechilddesc`

```
14159 \newcommand{\glstreechilddesc}[1]{%
14160   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
14161 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
14162 \newcommand{\glstreechildsymbol}[1]{%
14163   \glstreesymbol{#1}%
14164 }%

14165 \renewglossarystyle{tree}{%
14166   \renewenvironment{theglossary}%
14167     {\setlength{\parindent}{0pt}%
14168     \setlength{\parskip}{0pt plus 0.3pt}}%
14169   {%
14170     \renewcommand*{\glossaryheader}{}%
14171     \renewcommand*{\glsgroupheading}[1]{}%
14172     \renewcommand{\glossentry}[2]{%
14173       \hangindent0pt\relax
14174       \parindent0pt\relax
14175       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14176       \glstreesymbol{##1}%
14177       \glstreedesc{##1}%
14178       \glstreeprelocation##2\par
14179     }%
14180     \renewcommand{\subglossentry}[3]{%
14181       \hangindent##1\glstreeindent\relax
14182       \parindent##1\glstreeindent\relax
14183       \ifnum##1=1\relax
14184         \glssubentryitem{##2}%
14185       \fi
14186       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
14187       \glstreechildsymbol{##2}%
14188       \glstreechilddesc{##2}%
14189       \glstreechildprelocation ##3\par
14190     }%
14191     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14192   }%
14193 }
14194 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
14195 \ifdef{\@glstyle@treegroup}
```

```

14196 {%
14197   \renewglossarystyle{treegroup}{%
14198     \setglossarystyle{tree}%
14199     \renewcommand{\glsgroupheading}[1]{\par
14200       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
14201       \nopagebreak\indexspace\nobreak\@afterheading}%
14202   }
14203 }
14204 {}

```

Similarly for treehypergroup

```

14205 \ifdef{\@glsstyle@treehypergroup}
14206 {%
14207   \renewglossarystyle{treehypergroup}{%
14208     \setglossarystyle{tree}%
14209     \renewcommand*{\glossaryheader}{%
14210       \par\noindent\glstreenavigationfmt{\glstravigation}\par
14211       \nobreak\@afterheading\indexspace}%
14212     \renewcommand*{\glsgroupheading}[1]{%
14213       \par\noindent
14214       \glstreegroupheaderfmt
14215       {\glstravigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
14216       \nopagebreak\indexspace\nobreak\@afterheading}%
14217   }
14218 }
14219 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

14220 \ifdef{\@glsstyle@treenoname}
14221 {%

```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```

14222   \newcommand{\glstreenonamedesc}[1]{%
14223     \glstreepredesc\glossentrydesc{##1}\glspostdescription
14224   }%

```

Similarly for the symbol.

treenonamesymbol

```

14225   \newcommand{\glstreenonamesymbol}[1]{%
14226     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
14227   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

14228   \newcommand{\glstreenonamechilddesc}[1]{%
14229     \glossentrydesc{##1}\glspostdescription
14230   }%

```

```

14231 \renewglossarystyle{treenoname}{%
14232   \renewenvironment{theglossary}%
14233     {\setlength{\parindent}{0pt}%
14234      \setlength{\parskip}{0pt plus 0.3pt}}%
14235   {}%
14236   \renewcommand*{\glossaryheader}{}%
14237   \renewcommand*{\glsgroupheading}[1]{}%
14238   \renewcommand{\glossentry}[2]{%
14239     \hangindent0pt\relax
14240     \parindent0pt\relax
14241     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14242     \glstreenonamesymbol{##1}%
14243     \glstreenonamedesc{##1}%
14244     \glstreeprelocation##2\par
14245   }%
14246   \renewcommand{\subglossentry}[3]{%
14247     \hangindent##1\glstreeindent\relax
14248     \parindent##1\glstreeindent\relax
14249     \ifnum##1=1\relax
14250       \glssubentryitem{##2}%
14251     \fi
14252     \glstarget{##2}{\strut}%
14253     \glstreenonamechilddesc{##2}%
14254     \glstreechildprelocation##3\par
14255   }%
14256   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14257 }
14258 }
14259 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

14260 \ifdef{\@glsstyle@treenonamegroup}
14261 {%
14262   \renewglossarystyle{treenonamegroup}{%
14263     \setglossarystyle{treenoname}%
14264     \renewcommand{\glsgroupheading}[1]{\par
14265       \noindent\glstreegroupheaderfmt
14266       {\glsgetgrouptitle{##1}}}%
14267     \nopagebreak\indexspace\nobreak\@afterheading
14268   }%
14269 }
14270 }
14271 {}

```

Similarly for `treenonamehypergroup`

```

14272 \ifdef{\@glsstyle@treenonamehypergroup}
14273 {%
14274   \renewglossarystyle{treenonamehypergroup}{%
14275     \setglossarystyle{treenoname}%
14276     \renewcommand*{\glossaryheader}{%

```



```

14277      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
14278      \nobreak\@afterheading\indexspace}%
14279      \renewcommand*{\glsgroupheading}[1]{%
14280      \par\noindent
14281      \glstreegroupheaderfmt
14282      {\glsnavhypertarget{##1}{\glsgroupgetgrouptitle{##1}}}%
14283      \nopagebreak\indexspace\nobreak\@afterheading}%
14284  }
14285 }
14286 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

14287 \ifdef{\glstyle@almtree}
14288 {}

```

Only redefine this if it's already been defined.

`mbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

14289 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
14290   {%
14291     \let\par\glxtrAltTreePar
14292     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
14293     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
14294   }%
14295 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

14296 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

14297 \newcommand{\glxtrAltTreePar}{%
14298   \@@par
14299   \glxtrAltTreeSetHangIndent
14300   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
14301 }

```

`mbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

14302 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
14303   \glxtralttreeSymbolDescLocation{#2}{#3}%
14304 }

```

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.

```
14305 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
14306 \newcommand*\glxstralttreeInit{%
14307   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
14308   \glxstrAltTreeIndent=\parindent
14309 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
14310 \newcommand*\gglsetwidest}[2][0]{%
14311   \csgdef{@glswidestname\romannumeral#1}{#2}%
14312 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
14313 \newcommand*\eglssetwidest}[2][0]{%
14314   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14315 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
14316 \newcommand*\xglsetwidest}[2][0]{%
14317   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14318 }
```

glupdatewidest Only sets if new value is wider than old value.

```
14319 \newcommand*\glupdatewidest}[2][0]{%
14320   \ifcsundef{@glswidestname\romannumeral#1}%
14321   {\csdef{@glswidestname\romannumeral#1}{#2}}%
14322   {%
14323     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14324     \settowidth{\dimen@ii}{#2}%
14325     \ifdim\dimen@ii>\dimen@
14326     \csdef{@glswidestname\romannumeral#1}{#2}%
14327     \fi
14328   }%
14329 }
```

glupdatewidest As above but global definition.

```
14330 \newcommand*\gglupdatewidest}[2][0]{%
14331   \ifcsundef{@glswidestname\romannumeral#1}%
14332   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
14333   {%
14334     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
14335     \settowidth{\dimen@ii}{#2}%
14336     \ifdim\dimen@ii>\dimen@
14337     \csgdef{@glswidestname\romannumeral#1}{#2}%
14338     \fi
```

```

14339 }%
14340 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

14341 \newcommand*{\eglsupdatewidest}[2][0]{%
14342 \ifcsundef{@glswidestname\romannumeral#1}%
14343 {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
14344 {%
14345 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
14346 \settowidth{\dimen@ii}{#2}%
14347 \ifdim\dimen@ii>\dimen0
14348 \protected@csedef{@glswidestname\romannumeral#1}{#2}%
14349 \fi
14350 }%
14351 }

```

`glsupdatewidest` As above but global.

```

14352 \newcommand*{\xglsupdatewidest}[2][0]{%
14353 \ifcsundef{@glswidestname\romannumeral#1}%
14354 {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
14355 {%
14356 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
14357 \settowidth{\dimen@ii}{#2}%
14358 \ifdim\dimen@ii>\dimen0
14359 \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
14360 \fi
14361 }%
14362 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

14363 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

14364 \newcommand*{\glsgetwidestsubname}[1]{%
14365 \ifcsundef{@glswidestname\romannumeral#1}%
14366 {\@glswidestname}%
14367 {\csuse{@glswidestname\romannumeral#1}}%
14368 }

```

`glsFindWidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```

14369 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname

```

`glsFindWidestUsedTopLevelName` Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```

14370 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
14371 \dimen@=0pt\relax

```

```

14372 \gls@tmplen=Opt\relax
14373 \foralllglossaries[#1]{\@gls@type}%
14374 {%
14375 \forglsentries[\@gls@type]{\@glo@label}%
14376 {%
14377 \ifglsused{\@glo@label}%
14378 {%
14379 \ifglshasparent{\@glo@label}%
14380 {}%
14381 {%
14382 \settowidth{\dimen@}%
14383 {\glstreenamfmt{\glstryname{\@glo@label}}}%
14384 \ifdim\dimen@>\gls@tmplen
14385 \gls@tmplen=\dimen@
14386 \eglssetwidest{\glstryname{\@glo@label}}%
14387 \fi
14388 }%
14389 }%
14390 {}%
14391 }%
14392 }%
14393 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

14394 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
14395 \dimen@=Opt\relax
14396 \gls@tmplen=Opt\relax
14397 \foralllglossaries[#1]{\@gls@type}%
14398 {%
14399 \forglsentries[\@gls@type]{\@glo@label}%
14400 {%
14401 \ifglsused{\@glo@label}%
14402 {%
14403 \settowidth{\dimen@}%
14404 {\glstreenamfmt{\glstryname{\@glo@label}}}%
14405 \ifdim\dimen@>\gls@tmplen
14406 \gls@tmplen=\dimen@
14407 \eglssetwidest{\glstryname{\@glo@label}}%
14408 \fi
14409 }%
14410 {}%
14411 }%
14412 }%
14413 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

14414 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
14415 \dimen@=Opt\relax

```

```

14416 \gls@tmplen=0pt\relax
14417 \forallglossaries[#1]{\@gls@type}%
14418 {%
14419 \forglsentries[\@gls@type]{\@glo@label}%
14420 {%
14421 \settowidth{\dimen@}%
14422 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14423 \ifdim\dimen@>\gls@tmplen
14424 \gls@tmplen=\dimen@
14425 \eglssetwidest{\glsentryname{\@glo@label}}%
14426 \fi
14427 }%
14428 }%
14429 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

14430 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
14431 \dimen@=0pt\relax
14432 \dimen@i=0pt\relax
14433 \dimen@ii=0pt\relax
14434 \forallglossaries[#1]{\@gls@type}%
14435 {%
14436 \forglsentries[\@gls@type]{\@glo@label}%
14437 {%
14438 \ifglsused{\@glo@label}%
14439 {%
14440 \ifglshasparent{\@glo@label}%
14441 {%
14442 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
14443 \ifglshasparent{\@glo@parent}%
14444 {%
14445 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
14446 \ifglshasparent{\@glo@parent}%
14447 }%
14448 {%
14449 \settowidth{\gls@tmplen}%
14450 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14451 \ifdim\gls@tmplen>\dimen@ii
14452 \dimen@ii=\gls@tmplen
14453 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
14454 \fi
14455 }%
14456 }%
14457 {%
14458 \settowidth{\gls@tmplen}%
14459 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14460 \ifdim\gls@tmplen>\dimen@i
14461 \dimen@i=\gls@tmplen

```

```

14462         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14463     \fi
14464 }%
14465 }%
14466 {%
14467     \settowidth{\gls@tmplen}%
14468         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14469     \ifdim\gls@tmplen>\dimen@
14470         \dimen@=\gls@tmplen
14471         \eglssetwidest{\glsentryname{\@glo@label}}%
14472     \fi
14473 }%
14474 }%
14475 {}%
14476 }%
14477 }%
14478 }

```

dWidestLevelTwo This is like \glsFindWidestUsedLevelTwo but doesn't check if the entry has been used.

```

14479 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
14480     \dimen@=0pt\relax
14481     \dimen@i=0pt\relax
14482     \dimen@ii=0pt\relax
14483     \foralllglossaries[#1]{\@gls@type}%
14484     {%
14485         \forglsentries[\@gls@type]{\@glo@label}%
14486         {%
14487             \ifglshasparent{\@glo@label}%
14488             {%
14489                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
14490                 \ifglshasparent{\@glo@parent}%
14491                 {%
14492                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
14493                     \ifglshasparent{\@glo@parent}%
14494                     {}%
14495                     {%
14496                         \settowidth{\gls@tmplen}%
14497                             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14498                         \ifdim\gls@tmplen>\dimen@ii
14499                             \dimen@ii=\gls@tmplen
14500                             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
14501                         \fi
14502                     }%
14503                 }%
14504             }%
14505             \settowidth{\gls@tmplen}%
14506                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14507             \ifdim\gls@tmplen>\dimen@i
14508                 \dimen@i=\gls@tmplen

```

```

14509         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
14510     \fi
14511 }%
14512 }%
14513 {%
14514     \settowidth{\gls@tmplen}%
14515         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14516     \ifdim\gls@tmplen>\dimen@
14517         \dimen@=\gls@tmplen
14518         \eglssetwidest{\glsentryname{\@glo@label}}%
14519     \fi
14520 }%
14521 }%
14522 }%
14523 }

```

`\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.

```

14524 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
14525     \dimen@=0pt\relax
14526     \gls@tmplen=0pt\relax
14527     #2=0pt\relax
14528     \foralllglossaries[#1]{\@gls@type}%
14529     {%
14530         \forglsentries[\@gls@type]{\@glo@label}%
14531         {%
14532             \ifglsused{\@glo@label}%
14533             {%
14534                 \settowidth{\dimen@}%
14535                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14536                 \ifdim\dimen@>\gls@tmplen
14537                     \gls@tmplen=\dimen@
14538                     \eglssetwidest{\glsentryname{\@glo@label}}%
14539                 \fi
14540                 \settowidth{\dimen@}%
14541                     {\glsentrysymbol{\@glo@label}}%
14542                 \ifdim\dimen@>#2\relax
14543                     #2=\dimen@
14544                 \fi
14545             }%
14546         }%
14547     }%
14548 }%
14549 }

```

`\stAnyNameSymbol` Like the above but doesn't check if the entry has been used.

```

14550 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14551     \dimen@=0pt\relax
14552     \gls@tmplen=0pt\relax

```

```

14553      #2=Opt\relax
14554      \foralllglossaries[#1]{\@gls@type}%
14555      {%
14556        \forglsentries[\@gls@type]{\@glo@label}%
14557        {%
14558          \settowidth{\dimen@}%
14559            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14560          \ifdim\dimen@>\gls@tmplen
14561            \gls@tmplen=\dimen@
14562            \eglssetwidest{\glsentryname{\@glo@label}}%
14563          \fi
14564          \settowidth{\dimen@}%
14565            {\glsentrysymbol{\@glo@label}}%
14566          \ifdim\dimen@>#2\relax
14567            #2=\dimen@
14568          \fi
14569        }%
14570      }%
14571    }

```

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.

```

14572    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14573      \dimen@=Opt\relax
14574      \gls@tmplen=Opt\relax
14575      #2=Opt\relax
14576      #3=Opt\relax
14577      \foralllglossaries[#1]{\@gls@type}%
14578      {%
14579        \forglsentries[\@gls@type]{\@glo@label}%
14580        {%
14581          \ifglsused{\@glo@label}%
14582          {%
14583            \settowidth{\dimen@}%
14584              {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14585            \ifdim\dimen@>\gls@tmplen
14586              \gls@tmplen=\dimen@
14587              \eglssetwidest{\glsentryname{\@glo@label}}%
14588            \fi
14589            \settowidth{\dimen@}%
14590              {\glsentrysymbol{\@glo@label}}%
14591            \ifdim\dimen@>#2\relax
14592              #2=\dimen@
14593            \fi
14594            \settowidth{\dimen@}%
14595              {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14596            \ifdim\dimen@>#3\relax

```



```

14597         #3=\dimen@
14598         \fi
14599     }%
14600     {}%
14601 }%
14602 }%
14603 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14604 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14605     \dimen@=0pt\relax
14606     \gls@tmplen=0pt\relax
14607     #2=0pt\relax
14608     #3=0pt\relax
14609     \forallglossaries[#1]{\@gls@type}%
14610     {%
14611         \forglsentries[\@gls@type]{\@glo@label}%
14612         {%
14613             \settowidth{\dimen@}%
14614             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14615             \ifdim\dimen@>\gls@tmplen
14616                 \gls@tmplen=\dimen@
14617                 \eglssetwidest{\glsentryname{\@glo@label}}%
14618             \fi
14619             \settowidth{\dimen@}%
14620             {\glsentrysymbol{\@glo@label}}%
14621             \ifdim\dimen@>#2\relax
14622                 #2=\dimen@
14623             \fi
14624             \settowidth{\dimen@}%
14625             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14626             \ifdim\dimen@>#3\relax
14627                 #3=\dimen@
14628             \fi
14629         }%
14630     }%
14631 }

```

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.

```

14632 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14633     \dimen@=0pt\relax
14634     \gls@tmplen=0pt\relax
14635     #2=0pt\relax
14636     \forallglossaries[#1]{\@gls@type}%
14637     {%
14638         \forglsentries[\@gls@type]{\@glo@label}%
14639         {%

```

```

14640     \ifglsused{\@glo@label}%
14641     {%
14642         \settowidth{\dimen@}%
14643         {\glstreenamefmt{\glentryname{\@glo@label}}}%
14644         \ifdim\dimen@>\gls@tmplen
14645             \gls@tmplen=\dimen@
14646             \eglssetwidest{\glentryname{\@glo@label}}%
14647         \fi
14648         \settowidth{\dimen@}%
14649         {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14650         \ifdim\dimen@>#2\relax
14651             #2=\dimen@
14652         \fi
14653     }%
14654     {}%
14655 }%
14656 }%
14657 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14658 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14659     \dimen@=0pt\relax
14660     \gls@tmplen=0pt\relax
14661     #2=0pt\relax
14662     \forallglossaries[#1]{\@gls@type}%
14663     {%
14664         \forglsentries[\@gls@type]{\@glo@label}%
14665         {%
14666             \settowidth{\dimen@}%
14667             {\glstreenamefmt{\glentryname{\@glo@label}}}%
14668             \ifdim\dimen@>\gls@tmplen
14669                 \gls@tmplen=\dimen@
14670                 \eglssetwidest{\glentryname{\@glo@label}}%
14671             \fi
14672             \settowidth{\dimen@}%
14673             {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14674             \ifdim\dimen@>#2\relax
14675                 #2=\dimen@
14676             \fi
14677         }%
14678     }%
14679 }

```

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`.

```

14680 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
14681     \glstreeindent=\glstxtrtreetopindent\relax
14682 }

```

uteTreeSubIndent

```
\glxstrComputeTreeSubIndent{<level>}{<label>}{<register>}
```

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
14683 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14684 \ifcsundef{@glswidestname\romannumeral#1}%
14685 {%
14686 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14687 }%
14688 {%
14689 \settowidth{#3}{\glstreenamefmt{
14690 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14691 }%
14692 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14693 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14694 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14695 \renewglossarystyle{alttree}{%
14696 \renewenvironment{theglossary}%
14697 {%
14698 \glxstralttreeInit
14699 \def\@gls@prevlevel{-1}%
14700 \mbox{}\par}%
14701 {\par}%
14702 \renewcommand*{\glossaryheader}{}%
14703 \renewcommand*{\glsgroupheading}[1]{}%
14704 \renewcommand{\glossentry}[2]{%
14705 \ifnum\@gls@prevlevel=0\relax
14706 \else
14707 \glxstrComputeTreeIndent{##1}%
14708 \fi
14709 \parindent\glstreeindent
14710 \glxstrAltTreeSetHangIndent
14711 \makebox[Opt][r]%
14712 {%
14713 \glstreenamebox{\glstreeindent}%
14714 {%
14715 \glsentryitem{##1}%
14716 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14717 }%
14718 }%
```

```

14719 \glxtralttreeSymbolDescLocation{##1}{##2}%
14720 \def\@gls@prevlevel{0}%
14721 }
14722 \renewcommand{\subglossentry}[3]{%
14723 \ifnum##1=1\relax
14724 \glssubentryitem{##2}%
14725 \fi
14726 \ifnum\@gls@prevlevel=##1\relax
14727 \else
14728 \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14729 \ifnum\@gls@prevlevel<##1\relax
14730 \setlength\glstreeindent\gls@tmplen
14731 \addtolength\glstreeindent\parindent
14732 \parindent\glstreeindent
14733 \else
14734 \ifnum\@gls@prevlevel=0\relax
14735 \glxtrComputeTreeIndent{##2}%
14736 \else
14737 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14738 \fi
14739 \addtolength\parindent{-\glstreeindent}%
14740 \setlength\glstreeindent\parindent
14741 \fi
14742 \fi
14743 \glxtrAltTreeSetSubHangIndent{##1}%
14744 \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14745 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14746 \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14747 \def\@gls@prevlevel{##1}%
14748 }%
14749 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14750 }
14751 }%
14752 {%
14753 }

```

Redefine alttreegroup so that it discourages a break after group headings. Can't use \@afterheading here as it messes with the first item of the group.

```

14754 \ifdef{\@glsstyle@alttreegroup}
14755 {%
14756 \renewglossarystyle{alttreegroup}{%
14757 \setglossarystyle{alttree}%
14758 \renewcommand{\glsgroupheading}[1]{\par
14759 \def\@gls@prevlevel{-1}%
14760 \hangindent0pt\relax
14761 \parindent0pt\relax
14762 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14763 \nopagebreak\indexspace\nopagebreak
14764 }%
14765 }%

```

```

14766 }%
14767 {%
14768 }

```

Similarly for `alttreehypergroup`.

```

14769 \ifdef{\@glsstyle@alttreehypergroup}
14770 {%
14771   \renewglossarystyle{alttreehypergroup}{%
14772     \setglossarystyle{alttree}%
14773     \renewcommand*{\glossaryheader}{%
14774       \par
14775       \def\@gls@prevlevel{-1}%
14776       \hangindent0pt\relax
14777       \parindent0pt\relax
14778       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14779     }%
14780     \renewcommand*{\glsgroupheading}[1]{%
14781       \par
14782       \def\@gls@prevlevel{-1}%
14783       \hangindent0pt\relax
14784       \parindent0pt\relax
14785       \glstreegroupheaderfmt
14786       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14787       \nopagebreak\indexspace\nopagebreak
14788     }%
14789   }
14790 }%
14791 {%
14792 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14793 \ifdef{\@glsstyle@mcolindexgroup}
14794 {%
14795   \renewglossarystyle{mcolindexgroup}{%
14796     \setglossarystyle{mcolindex}%
14797     \renewcommand*{\glsgroupheading}[1]{%
14798       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14799       \nopagebreak\indexspace\nobreak\@afterheading
14800     }%
14801   }
14802 }%
14803 {%
14804 }

```

Similarly for `mcolindexhypergroup`.

```

14805 \ifdef{\@glsstyle@mcolindexhypergroup}
14806 {%

```

```

14807 \renewglossarystyle{mcolindexhypergroup}{%
14808   \setglossarystyle{mcolindex}%
14809   \renewcommand*{\glossaryheader}{%
14810     \item\glstreenavigationfmt{\glsnavigation}%
14811     \indexspace
14812   }%
14813   \renewcommand*{\glsgroupheading}[1]{%
14814     \item\glstreegroupheaderfmt
14815       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14816     \nopagebreak\indexspace\nobreak\@afterheading
14817   }%
14818 }
14819 }%
14820 {%
14821 }

```

Similarly for mcolindexspannav.

```

14822 \ifdef{\@glsstyle@mcolindexspannav}
14823 {%
14824   \renewglossarystyle{mcolindexspannav}{%
14825     \setglossarystyle{index}%
14826     \renewenvironment{theglossary}%
14827     {%
14828       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14829       \setlength{\parindent}{0pt}%
14830       \setlength{\parskip}{0pt plus 0.3pt}%
14831       \let\item\glstreeitem}%
14832     {\end{multicols}}}%
14833   \renewcommand*{\glsgroupheading}[1]{%
14834     \item\glstreegroupheaderfmt
14835       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14836     \nopagebreak\indexspace\nobreak\@afterheading
14837   }%
14838 }
14839 }%
14840 {%
14841 }

```

Similarly for mcoltreegroup.

```

14842 \ifdef{\@glsstyle@mcoltreegroup}
14843 {%
14844   \renewglossarystyle{mcoltreegroup}{%
14845     \setglossarystyle{mcoltree}%
14846     \renewcommand{\glsgroupheading}[1]{\par
14847       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14848     \nopagebreak\indexspace\nobreak\@afterheading
14849   }%
14850 }
14851 }%
14852 {%

```

14853 }

Similarly for mcoltreehypergroup.

```
14854 \ifdef{\@glsstyle@mcoltreehypergroup}
14855 {%
14856   \renewglossarystyle{mcoltreehypergroup}{%
14857     \setglossarystyle{mcoltree}%
14858     \renewcommand*{\glossaryheader}{%
14859       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14860     }%
14861     \renewcommand*{\glsgroupheading}[1]{%
14862       \par\noindent
14863       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14864       \nopagebreak\indexspace\nobreak\@afterheading
14865     }%
14866   }
14867 }%
14868 {%
14869 }
```

Similarly for mcoltreespannav.

```
14870 \ifdef{\@glsstyle@mcoltreespannav}
14871 {%
14872   \renewglossarystyle{mcoltreespannav}{%
14873     \setglossarystyle{tree}%
14874     \renewenvironment{theglossary}%
14875     {%
14876       \begin{multicols}{\glsmcols}%
14877       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14878       \setlength{\parindent}{0pt}%
14879       \setlength{\parskip}{0pt plus 0.3pt}%
14880     }%
14881     {\end{multicols}}%
14882     \renewcommand*{\glsgroupheading}[1]{%
14883       \par\noindent
14884       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14885       \nopagebreak\indexspace\nobreak\@afterheading
14886     }%
14887   }
14888 }%
14889 {%
14890 }
```

Similarly for mcoltreenonamegroup.

```
14891 \ifdef{\@glsstyle@mcoltreenonamegroup}
14892 {%
14893   \renewglossarystyle{mcoltreenonamegroup}{%
14894     \setglossarystyle{mcoltreenoname}%
14895     \renewcommand{\glsgroupheading}[1]{\par
14896       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14897     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14898 }%
14899 }
14900 }%
14901 {%
14902 }

```

Similarly for mcoltreenonamehypergroup.

```

14903 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14904 {%
14905   \renewglossarystyle{mcoltreenonamehypergroup}{%
14906     \setglossarystyle{mcoltreenoname}%
14907     \renewcommand*{\glossaryheader}{%
14908       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14909     \renewcommand*{\glsgroupheading}[1]{%
14910       \par\noindent
14911       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14912       \nopagebreak\indexspace\nobreak\@afterheading}%
14913   }
14914 }%
14915 {%
14916 }

```

Similarly for mcoltreenonamespannav.

```

14917 \ifdef{\@glsstyle@mcoltreenonamespannav}
14918 {%
14919   \renewglossarystyle{mcoltreenonamespannav}{%
14920     \setglossarystyle{treenoname}%
14921     \renewenvironment{theglossary}%
14922     {%
14923       \begin{multicols}{\glsmcols}%
14924       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14925       \setlength{\parindent}{0pt}%
14926       \setlength{\parskip}{0pt plus 0.3pt}%
14927     }%
14928     {\end{multicols}}}%
14929   \renewcommand*{\glsgroupheading}[1]{%
14930     \par\noindent
14931     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14932     \nopagebreak\indexspace\nobreak\@afterheading}%
14933   }
14934 }%
14935 {%
14936 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

14937 \ifdef{\@glsstyle@mcolalttree}
14938 {%
14939   \renewglossarystyle{mcolalttree}{%
14940     \setglossarystyle{alttree}%
14941     \renewenvironment{theglossary}%

```



```

14942   {%
14943       \glstralttreeInit
14944       \def\@gls@prevlevel{-1}%
14945       \begin{multicols}{\glsmcols}%
14946   }%
14947   {\par\end{multicols}}%
14948 }
14949 }%
14950 {%
14951 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

14952 \ifdef{\@glsstyle@mcolalmtreegroup}
14953 {%
14954   \renewglossarystyle{mcolalmtreegroup}{%
14955     \setglossarystyle{mcolalmtree}%
14956     \renewcommand{\glsgroupheading}[1]{\par
14957       \def\@gls@prevlevel{-1}%
14958       \hangindent0pt\relax
14959       \parindent0pt\relax
14960       \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14961       \nopagebreak\indexspace\nopagebreak
14962   }%
14963 }
14964 }%
14965 {%
14966 }

```

Similarly for mcolalmtreehypergroup.

```

14967 \ifdef{\@glsstyle@mcolalmtreehypergroup}
14968 {%
14969   \renewglossarystyle{mcolalmtreehypergroup}{%
14970     \setglossarystyle{mcolalmtree}%
14971     \renewcommand*\{\glossaryheader}{%
14972       \par
14973       \def\@gls@prevlevel{-1}%
14974       \hangindent0pt\relax
14975       \parindent0pt\relax
14976       \glstreenavigationfmt{\glsnavigation}%
14977       \par\indexspace
14978   }%
14979   \renewcommand*\{\glsgroupheading}[1]{%
14980     \par
14981     \def\@gls@prevlevel{-1}%
14982     \hangindent0pt\relax
14983     \parindent0pt\relax
14984     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14985     \nopagebreak\indexspace\nopagebreak
14986   }%
14987 }

```

```

14988 }%
14989 {%
14990 }

```

Similarly for mcolalttreespannav.

```

14991 \ifdef{\@glsstyle@mcolalttreespannav}
14992 {%
14993   \renewglossarystyle{mcolalttreespannav}{%
14994     \setglossarystyle{almtree}%
14995     \renewenvironment{theglossary}%
14996     {%
14997       \glstralmtreeInit
14998       \def\@gls@prevlevel{-1}%
14999       \begin{multicols}{\@gls@ncols}%
15000         [\noindent\glstreenavigationfmt{\@glsnavigation}]%
15001     }%
15002     {\par\end{multicols}}%
15003     \renewcommand*{\@gls@groupheading}[1]{%
15004       \par
15005       \def\@gls@prevlevel{-1}%
15006       \hangindent0pt\relax
15007       \parindent0pt\relax
15008       \glstreegroupheaderfmt{\@glsnavhypertarget{##1}{\@glsgetgrouptitle{##1}}}%
15009       \nopagebreak\indexspace\nopagebreak
15010     }%
15011   }
15012 }%
15013 {%
15014 }

```

Reset the default style

```

15015 \ifx\@glossary@default@style\relax
15016 \else
15017   \setglossarystyle{\@glsxtr@current@style}
15018 \fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
15019 \NeedsTeXFormat{LaTeX2e}
15020 \ProvidesPackage{glossary-bookindex}[2018/12/01 1.38 (NLCT)]
```

Load required packages.

```
15021 \RequirePackage{multicol}
15022 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
15023 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
15024 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
15025 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
15026 \providecommand*{\glstrprelocation}{\space}
```

`indexprelocation` Separator used before location list for top-level entries. Version 1.22 has removed the `\ifglsnopostdot` check since this style doesn't display the description.

```
15027 \newcommand*{\glstrbookindexprelocation}[1]{%
15028   \glstrifhasfield{location}{#1}%
15029   {,\glstrprelocation}%
15030   {\glstrprelocation}%
15031 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
15032 \newcommand*{\glstrbookindexsubprelocation}[1]{%
15033   \glstrbookindexprelocation{#1}%
15034 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
15035 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
15036 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
15037 \newcommand{\glstrbookindexbetween}[2]{}
```

`indexsubbetween` Between two level 1 entries identified by the labels in the arguments.

```
15038 \newcommand{\glstrbookindexsubbetween}[2]{}
```

`exsubsubbetween` Between two level 2 entries identified by the labels in the arguments.

```
15039 \newcommand{\glstrbookindexsubsubbetween}[2]{}
```

`indexatendgroup` At the end of a letter group. The argument is the index of the last top-level entry.

```
15040 \newcommand{\glstrbookindexatendgroup}[1]{}
```

`exsubatendgroup` At the end of a letter group. The argument is the index of the last level 1 entry.

```
15041 \newcommand{\glstrbookindexsubatendgroup}[1]{}
```

`subsubatendgroup` At the end of a letter group. The argument is the index of the last level 2 entry.

```
15042 \newcommand{\glstrbookindexsubsubatendgroup}[1]{}
```

`kindexgroupskip` Group separator.

```
15043 \newcommand{\glstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

`indexformatheader` Group separator.

```
15044 \newcommand*{\glstrbookindexformatheader}[1]{%
15045   \par{\centering\glstreegroupheaderfmt{#1}\par}%
15046 }
```

`bookindexbookmark` Book mark group heading if supported.

```
15047 \ifdef\pdfbookmark
15048 {%
15049   \newcommand*{\glstrbookindexbookmark}[2]{%
15050     \ifdefstring{\@@glossarysec}{chapter}%
15051     {\pdfbookmark[1]{#1}{#2}}%
15052     {\pdfbookmark[2]{#1}{#2}}%
15053   }
15054 }
15055 {%
15056   \newcommand*{\glstrbookindexbookmark}[2]{%
15057 }
```

`kindexcolspread`

```
15058 \newcommand*{\glstrbookindexcolspread}{}
```

`indexmulticolenv`

```
15059 \newcommand*{\glstrbookindexmulticolenv}{multicols}
```

Define the style.

```

15060 \newglossarystyle{bookindex}{%
15061   \setglossarystyle{index}%
15062   \renewenvironment{theglossary}%
15063   {%
15064     \ifdefempty{glxstrbookindexcolspread}
15065     {%
15066       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
15067       {\glxstrbookindexcols}%
15068     }%
15069     {%
15070       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
15071       {\glxstrbookindexcols}[\glxstrbookindexcolspread]%
15072     }%
15073     \setlength{\parindent}{0pt}%
15074     \setlength{\parskip}{0pt plus 0.3pt}%
15075     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
15076     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
15077     \let\@glxstr@bookindex@between\@gobble
15078     \let\@glxstr@bookindex@subbetween\@gobble
15079     \let\@glxstr@bookindex@subsubbetween\@gobble
15080     \let\@glxstr@bookindex@atendgroup\relax
15081     \let\@glxstr@bookindex@subatendgroup\relax
15082     \let\@glxstr@bookindex@subsubatendgroup\relax
15083     \let\@glxstr@bookindex@groupskip\relax
15084   }%
15085   {%

```

Do end group hooks.

```

15086     \@glxstr@bookindex@subsubatendgroup
15087     \@glxstr@bookindex@subatendgroup
15088     \@glxstr@bookindex@atendgroup

```

End multicol environment.

```

15089     \expandafter\end\expandafter{\glxstrbookindexmulticolseenv}%
15090   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

15091   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

15092   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

15093     \@glxstr@bookindex@between{##1}%

```

Update separators.

```

15094     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
15095     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
15096     \let\@glxstr@bookindex@subbetween\@gobble
15097     \let\@glxstr@bookindex@subsubbetween\@gobble
15098     \edef\@glxstr@bookindex@between{%

```

```

15099      \noexpand\glxstrbookindexbetween{##1}%
15100  }%
15101      \edef\@glxstr@bookindex@atendgroup{%
15102          \noexpand\glxstrbookindexatendgroup{##1}%
15103      }%
15104      \let\@glxstr@bookindex@subatendgroup\relax
15105      \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

15106      \glstreeitem
15107          \glstryitem{##1}%
15108          \glstarget{##1}{\glxstrbookindexname{##1}}%
15109      \glxstrbookindexprelocation{##1}##2%
15110  }%
15111      \renewcommand{\subglossentry}[3]{%
15112          \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

15113      \glstreeitem
15114      \or

```

Level 1.

```

15115      \@glxstr@bookindex@sep
15116      \@glxstr@bookindex@subbetween{##2}%
15117      \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

15118      \let\@glxstr@bookindex@subsubbetween\@gobble
15119      \let\@glxstr@bookindex@subsep\glxstrbookindexparentsubchildsep
15120      \edef\@glxstr@bookindex@subbetween{%
15121          \noexpand\glxstrbookindexsubbetween{##2}%
15122      }%
15123      \edef\@glxstr@bookindex@atsubendgroup{%
15124          \noexpand\glxstrbookindexatsubendgroup{##1}%
15125      }%

```

Start sub-item.

```

15126      \glstreesubitem
15127          \glssubentryitem{##2}%
15128      \else

```

All other levels.

```

15129      \@glxstr@bookindex@subsep
15130      \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

15131      \let\@glxstr@bookindex@subsep\relax
15132      \edef\@glxstr@bookindex@subsubbetween{%
15133          \noexpand\glxstrbookindexsubsubbetween{##2}%
15134      }%
15135      \edef\@glxstr@bookindex@atsubsubendgroup{%
15136          \noexpand\glxstrbookindexatsubsubendgroup{##1}%
15137      }%

```

Start sub-sub-item.

```
15138 \glstreesubsubitem
15139 \fi
```

Format entry.

```
15140 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
15141 \glxtrbookindexsubprelocation{##2}##3%
15142 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
15143 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
15144 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
15145 \@glxtr@bookindex@subsubatendgroup
15146 \@glxtr@bookindex@subatendgroup
15147 \@glxtr@bookindex@atendgroup
15148 \@glxtr@bookindexgroupskip
```

Update separators.

```
15149 \let\@glxtr@bookindexgroupskip\glxtrbookindexgroupskip
15150 \let\@glxtr@bookindex@between\@gobble
15151 \let\@glxtr@bookindex@atendgroup\relax
15152 \let\@glxtr@bookindex@subatendgroup\relax
15153 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
15154 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
15155 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
15156 \glxtrbookindexformatheader{\thisgrptitle}%
15157 \nopagebreak\indexspace\nopagebreak\@afterheading
15158 }%
15159 }
```

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 `\glxtrbookindexthepage` instead of `\thepage` in case the page numbering has been set to something that contains formatting commands.

`\glxtrbookindexthepage` The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```
15160 \newcommand{\glxtrbookindexthepage}{%
15161 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
15162 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
15163 \newcommand*\glxtrbookindexmarkentry}[1]{%
15164   \protected@write\@auxout
15165   {\let\glxtrbookindexthepage\relax}%
15166   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
15167 }
```

etbookindexmark

```
15168 \newcommand*\glxtr@setbookindexmark}[2]{%
15169   \ifcsundef\glxtr@idxfirstmark@#1}%
15170   {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
15171   {}%
15172   \csgdef\glxtr@idxlastmark@#1}{#2}%
15173 }
```

indexfirstmarkfmt

```
15174 \newcommand*\glxtrbookindexfirstmarkfmt}[1]{%
15175   \glxentryname{#1}%
15176 }
```

indexfirstmark

```
15177 \newcommand*\glxtrbookindexfirstmark}{%
15178   \letcs\glxtr@label\glxtr@idxfirstmark@\glxtrbookindexthepage}%
15179   \ifdef\glxtr@label
15180   {\glxtrbookindexfirstmarkfmt\glxtr@label}}%
15181   {}%
15182 }
```

indexlastmarkfmt

```
15183 \newcommand*\glxtrbookindexlastmarkfmt}[1]{%
15184   \glxentryname{#1}%
15185 }
```

okindexlastmark

```
15186 \newcommand*\glxtrbookindexlastmark}{%
15187   \letcs\glxtr@label\glxtr@idxlastmark@\glxtrbookindexthepage}%
15188   \ifdef\glxtr@label
15189   {\glxtrbookindexlastmarkfmt\glxtr@label}}%
15190   {}%
15191 }
```


4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
15192 \NeedsTeXFormat{LaTeX2e}
15193 \ProvidesPackage{glossary-longextra}[2018/12/01 1.38 (NLCT)]
```

Load required packages.

```
15194 \RequirePackage{glossary-longbooktabs}
```

longextraNameFmt `\glslongextraNameFmt{<label>}`

Governs the way the name is displayed.

```
15195 \newcommand{\glslongextraNameFmt}[1]{%
15196   \glssentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%
15197 }
```

longextraDescFmt `\glslongextraDescFmt{<label>}`

Governs the way the description is displayed.

```
15198 \newcommand{\glslongextraDescFmt}[1]{%
15199   \glossentrydesc{#1}\glspostdescription
15200 }
```

longextraSymbolFmt `\glslongextraSymbolFmt{<label>}`

Governs the way the symbol is displayed.

```
15201 \newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

longextraLocationFmt `\glslongextraLocationFmt{<label>}{<location list>}`

Governs the way the location is displayed.

```
15202 \newcommand{\glslongextraLocationFmt}[2]{#2}
```

gextraSubNameFmt

```
\glslongextraSubNameFmt{<level>}{<label>}
```

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
15203 \newcommand{\glslongextraSubNameFmt}[2]{%
15204   \glssubentryitem{#2}\glstarget{#2}{\strut}%
15205 }
```

gextraSubDescFmt

```
\glslongextraSubDescFmt{<level>}{<label>}
```

Governs the way the child description is displayed.

```
15206 \newcommand{\glslongextraSubDescFmt}[2]{%
15207   \glslongextraDescFmt{#2}%
15208 }
```

extraSubSymbolFmt

```
\glslongextraSubSymbolFmt{<level>}{<label>}
```

Governs the way the child symbol is displayed.

```
15209 \newcommand{\glslongextraSubSymbolFmt}[2]{%
15210   \glslongextraSymbolFmt{#2}%
15211 }
```

raSubLocationFmt

```
\glslongextraSubLocationFmt{<level>}{<label>}{<location list>}
```

Governs the way the child location list is displayed.

```
15212 \newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

gextraNameAlign Alignment for the name column.

```
15213 \newcommand{\glslongextraNameAlign}{l}
```

gextraDescAlign Alignment for the description column.

```
15214 \newcommand{\glslongextraDescAlign}{>\raggedrightp{\glsdescwidth}}
```

extraSymbolAlign Alignment for the symbol column.

```
15215 \newcommand{\glslongextraSymbolAlign}{c}
```

raLocationAlign Alignment for the location column.

```
15216 \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).

```
15217 \newcommand{\glslongextraGroupHeading}[2]{}
```

traHeaderFormat Format for the column headers.

```
15218 \newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}
```

aNameDescHeader

```
15219 \newcommand{\glslongextraNameDescHeader}{%
15220 \glslongextraNameDescTabularHeader\endhead
15221 \glslongextraNameDescTabularFooter\endfoot
15222 }
```

scTabularHeader

```
15223 \newcommand{\glslongextraNameDescTabularHeader}{%
15224 \toprule
15225 \glslongextraHeaderFmt\entryname &
15226 \glslongextraHeaderFmt\descriptionname\tabularnewline
15227 \midrule
15228 }
```

scTabularFooter

```
15229 \newcommand{\glslongextraNameDescTabularFooter}{%
15230 \bottomrule
15231 }
```

Unlike the altree style, there aren't different widths for the hierarchical levels.

gextraSetWidest Provide in case the tree styles haven't been loaded.

```
15232 \newcommand*{\glslongextraSetWidest}[1]{%
15233 \def\@glslongextrawidestname{#1}%
15234 }
```

extrawidestname Pick up the widest name from the altree style if it has been set. (Will expand to nothing otherwise.)

```
15235 \newcommand*{\@glslongextrawidestname}{\csuse{\glswidestname}}
```

traUpdateWidest

```
15236 \newcommand*{\glslongextraUpdateWidest}[1]{%
15237 \ifundef\@glslongextrawidestname
15238 {\def\@glslongextrawidestname{#1}}%
15239 {%
15240 \settowidth{\dimen@}{\@glslongextrawidestname}%
15241 \settowidth{\dimen@ii}{#1}%
15242 \ifdim\dimen@ii>\dimen@
```

```

15243     \def\@glslongextrawidestname{#1}%
15244     \fi
15245 }%
15246 }

```

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`.

```
15247 \newcommand*{\glslongextraUpdateWidestChild}[2]{}
```

extraSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```

15248 \newcommand{\glslongextraSetDescWidth}{%
15249   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%

```

Has the widest name been set.

```

15250   \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
15251   \ifdim\dimen@>\gls@tmplen
15252     \gls@tmplen=\dimen@
15253   \fi

```

Description width is `\linewidth` less `4\tabcolsep` less the width of the name column.

```

15254   \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
15255 }

```

SymSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```
15256 \newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15257   \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```
15258   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```

Subtract `2\tabcolsep` and the symbol header width.

```

15259   \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
15260 }

```

LocSetDescWidth

Computes the value of `\glsdescwidth` for the styles that only have name, location and description columns.

```
15261 \newcommand{\glslongextraLocSetDescWidth}{%
```

Work out the size for just the name and description style.

```
15262   \glslongextraSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15263 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15264 }
```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that have name, symbol, location and description columns.

```
15265 \newcommand{\glslongextraSymLocSetDescWidth}{%
```

Work out the size for just the name, symbol and description style.

```
15266 \glslongextraSymSetDescWidth
```

Subtract 2\tabcolsep and the location list column width.

```
15267 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
15268 }
```

ExtraUseTabular If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```
15269 \newif\ifGlsLongExtraUseTabular
```

```
15270 \GlsLongExtraUseTabularfalse
```

raTabularVAlign Only used with the tabular setting.

```
15271 \newcommand*{\glslongextraTabularVAlign}{c}
```

long-name-desc Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```
15272 \newglossarystyle{long-name-desc}%
15273 {%
15274   \ifGlsLongExtraUseTabular
15275     \renewenvironment{theglossary}%
15276     {%
15277       \glslongextraSetDescWidth
15278       \edef\@glslongextra@begintab{%
15279         \noexpand\begin{tabular}{\glslongextraTabularVAlign}{%
15280           \expandonce\glslongextraNameAlign
15281           \expandonce\glslongextraDescAlign}}%
15282       \@glslongextra@begintab
15283     }%
15284     {%
15285       \glslongextraNameDescTabularFooter
15286       \end{tabular}%
15287     }%
15288     \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
15289   \else
15290     \renewenvironment{theglossary}%
15291     {%
15292       \glspatchLToutput
15293       \glslongextraSetDescWidth
15294       \edef\@glslongextra@begintab{%
15295         \noexpand\begin{longtable}{%
```

```

15296         \expandonce\glslongextraNameAlign
15297         \expandonce\glslongextraDescAlign}}}%
15298     \@glslongextra@begintab
15299     }%
15300     {\end{longtable}}}%
15301     \renewcommand*{\glossaryheader}{\glslongextraNameDescHeader}%
15302     \fi
15303     \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
15304     \renewcommand{\glossentry}[2]{%
15305         \glslongextraNameFmt{##1} &
15306         \glslongextraDescFmt{##1}\tabularnewline
15307     }%
15308     \renewcommand{\subglossentry}[3]{%
15309         \glslongextraSubNameFmt{##1}{##2}
15310         &
15311         \glslongextraSubDescFmt{##1}{##2}%
15312         \tabularnewline
15313     }%
15314     \ifglsnogroupskip
15315         \renewcommand*{\glsgroupskip}{}%
15316     \else
15317         \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15318     \fi
15319 }

```

cLocationHeader

```

15320 \newcommand{\glslongextraNameDescLocationHeader}{%
15321     \glslongextraNameDescLocationTabularHeader\endhead
15322     \glslongextraNameDescLocationTabularFooter\endfoot
15323 }

```

onTabularHeader

```

15324 \newcommand{\glslongextraNameDescLocationTabularHeader}{%
15325     \toprule
15326     \glslongextraHeaderFmt\entryname &
15327     \glslongextraHeaderFmt\descriptionname &
15328     \glslongextraHeaderFmt\pagelistname\tabularnewline
15329     \midrule
15330 }

```

onTabularFooter

```

15331 \newcommand{\glslongextraNameDescLocationTabularFooter}{%
15332     \bottomrule
15333 }

```

g-name-desc-loc Three columns: name, description and location list.

```

15334 \newglossarystyle{long-name-desc-loc}%
15335 {%
15336     \ifGlsLongExtraUseTabular

```

```

15337 \renewenvironment{theglossary}%
15338 {%
15339     \glslongextraLocSetDescWidth
15340     \edef\@glslongextra@begintab{%
15341         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15342             \expandonce\glslongextraNameAlign
15343             \expandonce\glslongextraDescAlign
15344             \expandonce\glslongextraLocationAlign
15345         }}%
15346     \@glslongextra@begintab
15347 }%
15348 {%
15349     \glslongextraNameDescLocationTabularFooter
15350     \end{tabular}%
15351 }%
15352 \renewcommand*\glossaryheader{\glslongextraNameDescLocationTabularHeader}%
15353 \else
15354 \renewenvironment{theglossary}%
15355 {%
15356     \glspatchLToutput
15357     \glslongextraLocSetDescWidth
15358     \edef\@glslongextra@begintab{%
15359         \noexpand\begin{longtable}{%
15360             \expandonce\glslongextraNameAlign
15361             \expandonce\glslongextraDescAlign
15362             \expandonce\glslongextraLocationAlign
15363         }}%
15364     \@glslongextra@begintab
15365 }%
15366 {\end{longtable}}%
15367 \renewcommand*\glossaryheader{\glslongextraNameDescLocationHeader}%
15368 \fi
15369 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15370 \renewcommand{\glossentry}[2]{%
15371     \glslongextraNameFmt{##1} &
15372     \glslongextraDescFmt{##1} &
15373     \glslongextraLocationFmt{##1}{##2}\tabularnewline
15374 }%
15375 \renewcommand{\subglossentry}[3]{%
15376     \glslongextraSubNameFmt{##1}{##2}&
15377     \glslongextraSubDescFmt{##1}{##2}&
15378     \glslongextraSubLocationFmt{##1}{##2}{##3}%
15379     \tabularnewline
15380 }%
15381 \ifglsgroupskip
15382     \renewcommand*\glsgroupskip{}%
15383 \else
15384     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15385 \fi

```

15386 }

aDescNameHeader

```
15387 \newcommand{\glslongextraDescNameHeader}{%
15388   \glslongextraDescNameTabularHeader\endhead
15389   \glslongextraDescNameTabularFooter\endfoot
15390 }
```

meTabularHeader

```
15391 \newcommand{\glslongextraDescNameTabularHeader}{%
15392   \toprule
15393   \glslongextraHeaderFmt\descriptionname&
15394   \glslongextraHeaderFmt\entryname \tabularnewline
15395   \midrule
15396 }
```

meTabularFooter

```
15397 \newcommand{\glslongextraDescNameTabularFooter}{%
15398   \bottomrule
15399 }
```

long-desc-name Like name-desc but swaps the columns.

```
15400 \newglossarystyle{long-desc-name}%
15401 {%
15402   \ifGlsLongExtraUseTabular
15403     \renewenvironment{theglossary}%
15404       {%
15405         \glslongextraSetDescWidth
15406         \edef\@glslongextra@begintab{%
15407           \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
15408             \expandonce\glslongextraDescAlign
15409             \expandonce\glslongextraNameAlign}}%
15410         \@glslongextra@begintab
15411       }%
15412     {%
15413       \glslongextraDescNameTabularFooter
15414       \end{tabular}%
15415     }%
15416     \renewcommand*\glossaryheader{\glslongextraDescNameTabularHeader}%
15417   \else
15418     \renewenvironment{theglossary}%
15419       {%
15420         \glspatchLToutput
15421         \glslongextraSetDescWidth
15422         \edef\@glslongextra@begintab{%
15423           \noexpand\begin{longtable}{%
15424             \expandonce\glslongextraDescAlign
15425             \expandonce\glslongextraNameAlign}}%
15426         \@glslongextra@begintab
```



```

15427 }%
15428 {\end{longtable}}%
15429 \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
15430 \fi
15431 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
15432 \renewcommand{\glossentry}[2]{%
15433   \glslongextraDescFmt{##1} &
15434   \glslongextraNameFmt{##1}\tabularnewline
15435 }%
15436 \renewcommand{\subglossentry}[3]{%
15437   \glslongextraSubDescFmt{##1}{##2} &
15438   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15439 }%
15440 \ifglsgroupskip
15441   \renewcommand*{\glsgroupskip}{}%
15442 \else
15443   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15444 \fi
15445 }

```

nDescNameHeader

```

15446 \newcommand{\glslongextraLocationDescNameHeader}{%
15447   \glslongextraLocationDescNameTabularHeader\endhead
15448   \glslongextraLocationDescNameTabularFooter\endfoot
15449 }

```

meTabularHeader

```

15450 \newcommand{\glslongextraLocationDescNameTabularHeader}{%
15451   \toprule
15452   \glslongextraHeaderFmt\pagelistname&
15453   \glslongextraHeaderFmt\descriptionname&
15454   \glslongextraHeaderFmt\entryname \tabularnewline
15455   \midrule
15456 }

```

meTabularFooter

```

15457 \newcommand{\glslongextraLocationDescNameTabularFooter}{%
15458   \bottomrule
15459 }

```

g-loc-desc-name Three columns: location, description and name.

```

15460 \newglossarystyle{long-loc-desc-name}%
15461 {%
15462   \ifGlsLongExtraUseTabular
15463   {%
15464     \glslongextraLocSetDescWidth
15465     \edef\@glslongextra@begintab{%
15466       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15467         \expandonce\glslongextraLocationAlign

```

```

15468         \expandonce\glslongextraDescAlign
15469         \expandonce\glslongextraNameAlign}}}%
15470     \@glslongextra@begintab
15471 }%
15472 {%
15473     \glslongextraLocationDescNameTabularFooter
15474     \end{tabular}%
15475 }%
15476 \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameTabularHeader}%
15477 \else
15478 \renewenvironment{theglossary}%
15479 {%
15480     \glspatchLToutput
15481     \glslongextraLocSetDescWidth
15482     \edef\@glslongextra@begintab{%
15483         \noexpand\begin{longtable}{%
15484             \expandonce\glslongextraLocationAlign
15485             \expandonce\glslongextraDescAlign
15486             \expandonce\glslongextraNameAlign}}}%
15487     \@glslongextra@begintab
15488 }%
15489 {\end{longtable}}}%
15490 \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameHeader}%
15491 \fi
15492 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15493 \renewcommand{\glossentry}[2]{%
15494     \glslongextraLocationFmt{##1}{##2} &
15495     \glslongextraDescFmt{##1} &
15496     \glslongextraNameFmt{##1}\tabularnewline
15497 }%
15498 \renewcommand{\subglossentry}[3]{%
15499     \glslongextraSubLocationFmt{##1}{##2}{##3} &
15500     \glslongextraSubDescFmt{##1}{##2} &
15501     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15502 }%
15503 \ifglsgroupskip
15504     \renewcommand*{\glsgroupskip}{}%
15505 \else
15506     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15507 \fi
15508 }

```

meDescSymHeader

```

15509 \newcommand{\glslongextraNameDescSymHeader}{%
15510 \glslongextraNameDescSymTabularHeader\endhead
15511 \glslongextraNameDescSymTabularFooter\endfoot
15512 }

```

ymTabularHeader

```

15513 \newcommand{\glslongextraNameDescSymTabularHeader}{%
15514 \toprule
15515 \glslongextraHeaderFmt\entryname &
15516 \glslongextraHeaderFmt\descriptionname &
15517 \glslongextraHeaderFmt\symbolname\tabularnewline
15518 \midrule
15519 }

```

ymTabularFooter

```

15520 \newcommand{\glslongextraNameDescSymTabularFooter}{%
15521 \bottomrule
15522 }

```

g-name-desc-sym Three column style with symbol in the third column.

```

15523 \newglossarystyle{long-name-desc-sym}%
15524 {%
15525 \ifGlsLongExtraUseTabular
15526 \renewenvironment{theglossary}%
15527 {%
15528 \glslongextraSymSetDescWidth
15529 \edef\@glslongextra@begintab{%
15530 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15531 \expandonce\glslongextraNameAlign
15532 \expandonce\glslongextraDescAlign
15533 \expandonce\glslongextraSymbolAlign
15534 }}%
15535 \@glslongextra@begintab
15536 }%
15537 {%
15538 \glslongextraNameDescSymTabularFooter
15539 \end{tabular}%
15540 }%
15541 \renewcommand*\glossaryheader{\glslongextraNameDescSymTabularHeader}%
15542 \else
15543 \renewenvironment{theglossary}%
15544 {%
15545 \glspatchLToutput
15546 \glslongextraSymSetDescWidth
15547 \edef\@glslongextra@begintab{%
15548 \noexpand\begin{longtable}{%
15549 \expandonce\glslongextraNameAlign
15550 \expandonce\glslongextraDescAlign
15551 \expandonce\glslongextraSymbolAlign
15552 }}%
15553 \@glslongextra@begintab
15554 }%
15555 {\end{longtable}}%
15556 \renewcommand*\glossaryheader{\glslongextraNameDescSymHeader}%
15557 \fi

```

```

15558 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15559 \renewcommand{\glossentry}[2]{%
15560   \glslongextraNameFmt{##1} &
15561   \glslongextraDescFmt{##1} &
15562   \glslongextraSymbolFmt{##1}\tabularnewline
15563 }%
15564 \renewcommand{\subglossentry}[3]{%
15565   \glslongextraSubNameFmt{##1}{##2} &
15566   \glslongextraSubDescFmt{##1}{##2} &
15567   \glslongextraSubSymbolFmt{##1}{##2}%
15568   \tabularnewline
15569 }%
15570 \ifglsgnogroupskip
15571   \renewcommand*{\glsgroupskip}{}%
15572 \else
15573   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15574 \fi
15575 }

```

mLocationHeader

```

15576 \newcommand{\glslongextraNameDescSymLocationHeader}{%
15577   \glslongextraNameDescSymLocationTabularHeader\endhead
15578   \glslongextraNameDescSymLocationTabularFooter\endfoot
15579 }

```

onTabularHeader

```

15580 \newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
15581   \toprule
15582   \glslongextraHeaderFmt\entryname &
15583   \glslongextraHeaderFmt\descriptionname &
15584   \glslongextraHeaderFmt\symbolname &
15585   \glslongextraHeaderFmt\pagelistname\tabularnewline
15586   \midrule
15587 }

```

onTabularFooter

```

15588 \newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
15589   \bottomrule
15590 }

```

me-desc-sym-loc Four columns: name, description and location

```

15591 \newglossarystyle{long-name-desc-sym-loc}%
15592 {%
15593   \ifGlsLongExtraUseTabular
15594     \renewenvironment{theglossary}%
15595     {%
15596       \glslongextraSymLocSetDescWidth
15597       \edef\@glslongextra@begintab{%
15598         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%

```

```

15599         \expandonce\glslongextraNameAlign
15600         \expandonce\glslongextraDescAlign
15601         \expandonce\glslongextraSymbolAlign
15602         \expandonce\glslongextraLocationAlign
15603     } }%
15604     \@glslongextra@begintab
15605 }%
15606 {%
15607     \glslongextraNameDescSymLocationTabularFooter
15608     \end{tabular}%
15609 }%
15610 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationTabularHeader}%
15611 \else
15612 \renewenvironment{theglossary}%
15613 {%
15614     \glspatchLToutput
15615     \glslongextraSymLocSetDescWidth
15616     \edef\@glslongextra@begintab{%
15617         \noexpand\begin{longtable}{%
15618             \expandonce\glslongextraNameAlign
15619             \expandonce\glslongextraDescAlign
15620             \expandonce\glslongextraSymbolAlign
15621             \expandonce\glslongextraLocationAlign
15622         } }%
15623     \@glslongextra@begintab
15624 }%
15625 {\end{longtable}}%
15626 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationHeader}%
15627 \fi
15628 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
15629 \renewcommand{\glossentry}[2]{%
15630     \glslongextraNameFmt{##1} &
15631     \glslongextraDescFmt{##1} &
15632     \glslongextraSymbolFmt{##1}&
15633     \glslongextraLocationFmt{##1}{##2}\tabularnewline
15634 }%
15635 \renewcommand{\subglossentry}[3]{%
15636     \glslongextraSubNameFmt{##1}{##2} &
15637     \glslongextraSubDescFmt{##1}{##2} &
15638     \glslongextraSubSymbolFmt{##1}{##2}&
15639     \glslongextraSubLocationFmt{##1}{##2}{##3}%
15640     \tabularnewline
15641 }%
15642 \ifglsgroupskip
15643     \renewcommand*{\glsgroupskip}{}%
15644 \else
15645     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15646 \fi
15647 }

```

meSymDescHeader

```
15648 \newcommand{\glslongextraNameSymDescHeader}{%
15649 \glslongextraNameSymDescTabularHeader\endhead
15650 \glslongextraNameSymDescTabularFooter\endfoot
15651 }
```

scTabularHeader

```
15652 \newcommand{\glslongextraNameSymDescTabularHeader}{%
15653 \toprule
15654 \glslongextraHeaderFmt\entryname &
15655 \glslongextraHeaderFmt\symbolname &
15656 \glslongextraHeaderFmt\descriptionname\tabularnewline
15657 \midrule
15658 }
```

scTabularFooter

```
15659 \newcommand{\glslongextraNameSymDescTabularFooter}{%
15660 \bottomrule
15661 }
```

g-name-sym-desc Three column style with symbol in the second column.

```
15662 \newglossarystyle{long-name-sym-desc}%
15663 {%
15664 \ifGlsLongExtraUseTabular
15665 \renewenvironment{theglossary}%
15666 {%
15667 \glslongextraSymSetDescWidth
15668 \edef\@glslongextra@begintab{%
15669 \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
15670 \expandonce\glslongextraNameAlign
15671 \expandonce\glslongextraSymbolAlign
15672 \expandonce\glslongextraDescAlign
15673 }}%
15674 \@glslongextra@begintab
15675 }%
15676 {%
15677 \glslongextraNameSymDescTabularFooter
15678 \end{tabular}%
15679 }%
15680 \renewcommand*{\glossaryheader}{\glslongextraNameSymDescTabularHeader}%
15681 \else
15682 \renewenvironment{theglossary}%
15683 {%
15684 \glspatchLToutput
15685 \glslongextraSymSetDescWidth
15686 \edef\@glslongextra@begintab{%
15687 \noexpand\begin{longtable}{%
15688 \expandonce\glslongextraNameAlign
15689 \expandonce\glslongextraSymbolAlign
```

```

15690         \expandonce\glslongextraDescAlign
15691     } }%
15692     \@glslongextra@begintab
15693 }%
15694 {\end{longtable}}%
15695 \renewcommand*\glossaryheader{\glslongextraNameSymDescHeader}%
15696 \fi
15697 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
15698 \renewcommand\glossentry[2]{%
15699     \glslongextraNameFmt{##1} &
15700     \glslongextraSymbolFmt{##1} &
15701     \glslongextraDescFmt{##1}\tabularnewline
15702 }%
15703 \renewcommand\subglossentry[3]{%
15704     \glslongextraSubNameFmt{##1}{##2} &
15705     \glslongextraSubSymbolFmt{##1}{##2} &
15706     \glslongextraSubDescFmt{##1}{##2}\tabularnewline
15707 }%
15708 \ifglsgroupskip
15709     \renewcommand*\glsgroupskip{}%
15710 \else
15711     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
15712 \fi
15713 }

```

cLocationHeader

```

15714 \newcommand{\glslongextraNameSymDescLocationHeader}{%
15715     \glslongextraNameSymDescLocationTabularHeader\endhead
15716     \glslongextraNameSymDescLocationTabularFooter\endfoot
15717 }

```

onTabularHeader

```

15718 \newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
15719     \toprule
15720     \glslongextraHeaderFmt\entryname &
15721     \glslongextraHeaderFmt\symbolname &
15722     \glslongextraHeaderFmt\descriptionname &
15723     \glslongextraHeaderFmt\pagelistname\tabularnewline
15724     \midrule
15725 }

```

onTabularFooter

```

15726 \newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
15727     \bottomrule
15728 }

```

me-sym-desc-loc Four column style with symbol in the second column.

```

15729 \newglossarystyle{long-name-sym-desc-loc}%
15730 {%

```

```

15731 \ifGlsLongExtraUseTabular
15732   \renewenvironment{theglossary}%
15733   {%
15734     \glslongextraSymLocSetDescWidth
15735     \edef\@glslongextra@begintab{%
15736       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15737         \expandonce\glslongextraNameAlign
15738         \expandonce\glslongextraSymbolAlign
15739         \expandonce\glslongextraDescAlign
15740         \expandonce\glslongextraLocationAlign
15741       }}%
15742     \@glslongextra@begintab
15743   }%
15744   {%
15745     \glslongextraNameSymDescLocationTabularFooter
15746     \end{tabular}%
15747   }%
15748   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationTabularHeader}%
15749 \else
15750   \renewenvironment{theglossary}%
15751   {%
15752     \glspatchLToutput
15753     \glslongextraSymLocSetDescWidth
15754     \edef\@glslongextra@begintab{%
15755       \noexpand\begin{longtable}{%
15756         \expandonce\glslongextraNameAlign
15757         \expandonce\glslongextraSymbolAlign
15758         \expandonce\glslongextraDescAlign
15759         \expandonce\glslongextraLocationAlign
15760       }}%
15761     \@glslongextra@begintab
15762   }%
15763   {\end{longtable}}%
15764   \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationHeader}%
15765 \fi
15766 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
15767 \renewcommand{\glossentry}[2]{%
15768   \glslongextraNameFmt{##1} &
15769   \glslongextraSymbolFmt{##1} &
15770   \glslongextraDescFmt{##1} &
15771   \glslongextraLocationFmt{##1}{##2}\tabularnewline
15772 }%
15773 \renewcommand{\subglossentry}[3]{%
15774   \glslongextraSubNameFmt{##1}{##2} &
15775   \glslongextraSubSymbolFmt{##1}{##2} &
15776   \glslongextraSubDescFmt{##1}{##2} &
15777   \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
15778 }%
15779 \ifglsgroupskip

```



```

15780 \renewcommand*{\glsgroupskip}{}%
15781 \else
15782 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15783 \fi
15784 }

```

meDescNameHeader

```

15785 \newcommand{\glslongextraSymDescNameHeader}{%
15786 \glslongextraSymDescNameTabularHeader\endhead
15787 \glslongextraSymDescNameTabularFooter\endfoot
15788 }

```

meTabularHeader

```

15789 \newcommand{\glslongextraSymDescNameTabularHeader}{%
15790 \toprule
15791 \glslongextraHeaderFmt\symbolname &
15792 \glslongextraHeaderFmt\descriptionname &
15793 \glslongextraHeaderFmt\entryname\tabularnewline
15794 \midrule
15795 }

```

meTabularFooter

```

15796 \newcommand{\glslongextraSymDescNameTabularFooter}{%
15797 \bottomrule
15798 }

```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```

15799 \newglossarystyle{long-sym-desc-name}%
15800 {%
15801 \ifGlsLongExtraUseTabular
15802 \renewenvironment{theglossary}%
15803 {%
15804 \glslongextraSymSetDescWidth
15805 \edef\@glslongextra@begintab{%
15806 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15807 \expandonce\glslongextraSymbolAlign
15808 \expandonce\glslongextraDescAlign
15809 \expandonce\glslongextraNameAlign
15810 }}%
15811 \@glslongextra@begintab
15812 }%
15813 {%
15814 \glslongextraSymDescNameTabularFooter
15815 \end{tabular}%
15816 }%
15817 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
15818 \else
15819 \renewenvironment{theglossary}%

```

```

15820 {%
15821     \glspatchLToutput
15822     \glslongextraSymSetDescWidth
15823     \edef\@glslongextra@begintab{%
15824         \noexpand\begin{longtable}{%
15825             \expandonce\glslongextraSymbolAlign
15826             \expandonce\glslongextraDescAlign
15827             \expandonce\glslongextraNameAlign
15828         }}%
15829     \@glslongextra@begintab
15830 }%
15831 {\end{longtable}}%
15832 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameHeader}%
15833 \fi
15834 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15835 \renewcommand{\glossentry}[2]{%
15836     \glslongextraSymbolFmt{##1} &
15837     \glslongextraDescFmt{##1} &
15838     \glslongextraNameFmt{##1}\tabularnewline
15839 }%
15840 \renewcommand{\subglossentry}[3]{%
15841     \glslongextraSubSymbolFmt{##1}{##2} &
15842     \glslongextraSubDescFmt{##1}{##2} &
15843     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15844 }%
15845 \ifglsgnogroupskip
15846     \renewcommand*{\glsgroupskip}{}%
15847 \else
15848     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15849 \fi
15850 }

```

meDescNameHeader

```

15851 \newcommand{\glslongextraLocationSymDescNameHeader}{%
15852     \glslongextraLocationSymDescNameTabularHeader\endhead
15853     \glslongextraLocationSymDescNameTabularFooter\endfoot
15854 }

```

meTabularHeader

```

15855 \newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
15856     \toprule
15857     \glslongextraHeaderFmt\pagelistname &
15858     \glslongextraHeaderFmt\symbolname &
15859     \glslongextraHeaderFmt\descriptionname &
15860     \glslongextraHeaderFmt\entryname\tabularnewline
15861     \midrule
15862 }

```

meTabularFooter

```

15863 \newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
15864 \bottomrule
15865 }

```

c-sym-desc-name Four column style with location list, symbol, description and name.

```

15866 \newglossarystyle{long-loc-sym-desc-name}%
15867 {%
15868 \ifGlsLongExtraUseTabular
15869 \renewenvironment{theglossary}%
15870 {%
15871 \glslongextraSymLocSetDescWidth
15872 \edef\@glslongextra@begintab{%
15873 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15874 \expandonce\glslongextraLocationAlign
15875 \expandonce\glslongextraSymbolAlign
15876 \expandonce\glslongextraDescAlign
15877 \expandonce\glslongextraNameAlign
15878 }}%
15879 \@glslongextra@begintab
15880 }%
15881 {%
15882 \glslongextraLocationSymDescNameTabularFooter
15883 \end{tabular}%
15884 }%
15885 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameTabularHeader}%
15886 \else
15887 \renewenvironment{theglossary}%
15888 {%
15889 \glspatchLToutput
15890 \glslongextraSymLocSetDescWidth
15891 \edef\@glslongextra@begintab{%
15892 \noexpand\begin{longtable}{%
15893 \expandonce\glslongextraLocationAlign
15894 \expandonce\glslongextraSymbolAlign
15895 \expandonce\glslongextraDescAlign
15896 \expandonce\glslongextraNameAlign
15897 }}%
15898 \@glslongextra@begintab
15899 }%
15900 {\end{longtable}}%
15901 \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameHeader}%
15902 \fi
15903 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
15904 \renewcommand{\glossentry}[2]{%
15905 \glslongextraLocationFmt{##1}{##2} &
15906 \glslongextraSymbolFmt{##1} &
15907 \glslongextraDescFmt{##1} &
15908 \glslongextraNameFmt{##1}\tabularnewline
15909 }%

```

```

15910 \renewcommand{\subglossentry}[3]{%
15911     \glslongextraSubLocationFmt{##1}{##2}{##3} &
15912     \glslongextraSubSymbolFmt{##1}{##2} &
15913     \glslongextraSubDescFmt{##1}{##2} &
15914     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15915 }%
15916 \ifglsnogroupskip
15917 \renewcommand*\{\glsgroupskip}{}%
15918 \else
15919 \renewcommand*\{\glsgroupskip}{\glspenaltygroupskip}%
15920 \fi
15921 }

```

scSymNameHeader

```

15922 \newcommand{\glslongextraDescSymNameHeader}{%
15923 \glslongextraDescSymNameTabularHeader\endhead
15924 \glslongextraDescSymNameTabularFooter\endfoot
15925 }

```

meTabularHeader

```

15926 \newcommand{\glslongextraDescSymNameTabularHeader}{%
15927 \toprule
15928 \glslongextraHeaderFmt\descriptionname &
15929 \glslongextraHeaderFmt\symbolname &
15930 \glslongextraHeaderFmt\entryname\tabularnewline
15931 \midrule
15932 }

```

meTabularFooter

```

15933 \newcommand{\glslongextraDescSymNameTabularFooter}{%
15934 \bottomrule
15935 }

```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```

15936 \newglossarystyle{long-desc-sym-name}{%
15937 {%
15938 \ifGlsLongExtraUseTabular
15939 \renewenvironment{theglossary}{%
15940 {%
15941 \glslongextraSymSetDescWidth
15942 \edef\@glslongextra@begintab{%
15943 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
15944 \expandonce\glslongextraDescAlign
15945 \expandonce\glslongextraSymbolAlign
15946 \expandonce\glslongextraNameAlign
15947 }}%
15948 \@glslongextra@begintab
15949 }%

```

```

15950   {%
15951       \glslongextraDescSymNameTabularFooter
15952       \end{tabular}%
15953   }%
15954   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameTabularHeader}%
15955   \else
15956   \renewenvironment{theglossary}%
15957   {%
15958       \glspatchLToutput
15959       \glslongextraSymSetDescWidth
15960       \edef\@glslongextra@begintab{%
15961           \noexpand\begin{longtable}{%
15962               \expandonce\glslongextraDescAlign
15963               \expandonce\glslongextraSymbolAlign
15964               \expandonce\glslongextraNameAlign
15965           }}%
15966       \@glslongextra@begintab
15967   }%
15968   {\end{longtable}}%
15969   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameHeader}%
15970   \fi
15971   \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
15972   \renewcommand{\glossentry}[2]{%
15973       \glslongextraDescFmt{##1} &
15974       \glslongextraSymbolFmt{##1} &
15975       \glslongextraNameFmt{##1}\tabularnewline
15976   }%
15977   \renewcommand{\subglossentry}[3]{%
15978       \glslongextraSubDescFmt{##1}{##2} &
15979       \glslongextraSubSymbolFmt{##1}{##2} &
15980       \glslongextraSubNameFmt{##1}{##2}\tabularnewline
15981   }%
15982   \ifglsgnognroupskip
15983       \renewcommand*{\glsgroupskip}{}%
15984   \else
15985       \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
15986   \fi
15987 }

```

scSymNameHeader

```

15988 \newcommand{\glslongextraLocationDescSymNameHeader}{%
15989   \glslongextraLocationDescSymNameTabularHeader\endhead
15990   \glslongextraLocationDescSymNameTabularFooter\endfoot
15991 }

```

meTabularHeader

```

15992 \newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
15993   \toprule
15994   \glslongextraHeaderFmt\pagelistname &

```

```

15995 \glslongextraHeaderFmt\descriptionname &
15996 \glslongextraHeaderFmt\symbolname &
15997 \glslongextraHeaderFmt\entryname\tabularnewline
15998 \midrule
15999 }

```

meTabularFooter

```

16000 \newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
16001 \bottomrule
16002 }

```

c-desc-sym-name Four column style with location list, description, symbol and name.

```

16003 \newglossarystyle{long-loc-desc-sym-name}%
16004 {%
16005 \ifGlsLongExtraUseTabular
16006 \renewenvironment{theglossary}%
16007 {%
16008 \glslongextraSymLocSetDescWidth
16009 \edef\@glslongextra@begintab{%
16010 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16011 \expandonce\glslongextraLocationAlign
16012 \expandonce\glslongextraDescAlign
16013 \expandonce\glslongextraSymbolAlign
16014 \expandonce\glslongextraNameAlign
16015 }}%
16016 \@glslongextra@begintab
16017 }%
16018 {%
16019 \glslongextraLocationDescSymNameTabularFooter
16020 \end{tabular}%
16021 }%
16022 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameTabularHeader}%
16023 \else
16024 \renewenvironment{theglossary}%
16025 {%
16026 \glspatchLToutput
16027 \glslongextraSymLocSetDescWidth
16028 \edef\@glslongextra@begintab{%
16029 \noexpand\begin{longtable}%
16030 \expandonce\glslongextraLocationAlign
16031 \expandonce\glslongextraDescAlign
16032 \expandonce\glslongextraSymbolAlign
16033 \expandonce\glslongextraNameAlign
16034 }}%
16035 \@glslongextra@begintab
16036 }%
16037 {\end{longtable}}%
16038 \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameHeader}%
16039 \fi

```

```

16040 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
16041 \renewcommand{\glossentry}[2]{%
16042   \glslongextraLocationFmt{##1}{##2} &
16043   \glslongextraDescFmt{##1} &
16044   \glslongextraSymbolFmt{##1} &
16045   \glslongextraNameFmt{##1}\tabularnewline
16046 }%
16047 \renewcommand{\subglossentry}[3]{%
16048   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16049   \glslongextraSubDescFmt{##1}{##2} &
16050   \glslongextraSubSymbolFmt{##1}{##2} &
16051   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16052 }%
16053 \ifglsnogroupskip
16054   \renewcommand*{\glsgroupskip}{}%
16055 \else
16056   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16057 \fi
16058 }

```

Glossary

bib2gls A command line Java application that selects entries from a .bib file and converts them to glossary definitions (like bibtex but also performs hierarchical sorting and collation, thus omitting the need for **xindy** or **makeindex**). Further details at: <http://www.dickimaw-books.com/software/bib2gls/..>

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

First use flag A conditional that determines whether or not the entry has been used according to the rules of **first use**.

First use text The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

makeindex An indexing application.

xindy An flexible indexing application with multilingual support written in Perl.

Change History

0.1 (2015-11-22)		\@Glsymbol@: added redefinition	80
General: Initial experimental release	5	\@Glsymbolplural@: added redefinition	81
0.2 (2015-11-30)		\@GLstext@: added redefinition	76
\Glsfmtshort: new	332	\@Glsuseri@: added redefinition	81
\glsfmtshort: new	331	\@Glsuserii@: added redefinition	81
\Glsfmtshortpl: new	332	\@Glsuseriii@: added redefinition	82
\glsfmtshortpl: new	331	\@Glsuseriv@: added redefinition	82
short: switched inline full form to short (long)	234	\@Glsuserv@: added redefinition	82
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	83
\@ACRlong: added redefinition	86	\@acrlong: added redefinition	85
\@ACRlongpl: added redefinition	87	\@acrlongpl: added redefinition	86
\@ACRshort: added redefinition	84	\@acrshort: added redefinition	83
\@ACRshortpl: added redefinition	85	\@acrshortpl: added redefinition	84
\@Acrlong: added redefinition	85	\@gls@field@link: added optional argument	67
\@Acrlongpl: added redefinition	86	\@glsdescplural@: added redefinition	79
\@Acrshort: added redefinition	83	\@glsfirst@: added redefinition	76
\@Acrshortpl: added redefinition	84	\@glsfirstplural@: added redefinition	78
\@GLSdesc@: added redefinition	79	\@glsplural@: added redefinition	77
\@GLSdescplural@: added redefinition	80	\@glssymbolplural@: added redefinition	80
\@GLSfirst@: added redefinition	77	\@glsxtr@defaultnoglossarywarning: new	139
\@GLSfirstplural@: added redefinition	78	\@glsxtr@field@linkdefs: new	75
\@GLSname@: added redefinition	79	\@glsxtr@insertdots: new	202
\@GLSplural@: added redefinition	77	\@print@glossary: added redefinition	136
\@GLSSymbol@: added redefinition	80	\glsabbrvdefaultfont: renamed from \abbrvdefaultfont	208
\@GLSSymbolplural@: added redefinition	81	\glsaccessdesc: new	166
\@GLStext@: added redefinition	76	\glsaccessdescplural: new	166
\@GLSuseri@: added redefinition	81	\glsaccessfirst: new	163
\@GLSuserii@: added redefinition	82	\glsaccessfirstplural: new	164
\@GLSuseriii@: added redefinition	82	\Glsaccesslong: new	168
\@GLSuseriv@: added redefinition	82	\glsaccesslong: new	168
\@GLSuserv@: added redefinition	82	\glsaccessname: new	162
\@GLSuservi@: added redefinition	83	\glsaccessplural: new	163
\@Glsdesc@: added redefinition	79	\Glsaccessshort: new	167
\@Glsdescplural@: added redefinition	80	\glsaccessshort: new	167
\@Glsfirst@: added redefinition	77	\Glsaccessshortpl: new	168
\@Glsfirstplural@: added redefinition	78		
\@Glsname@: added redefinition	79		
\@Glsplural@: added redefinition	77		

\glsaccesssshortpl: new	168	\cGLSpl: new	113
\glsaccessssymbol: new	165	\cGLSpl@: new	113
\glsaccessssymbolplural: new	165	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	162	new	108
\glstentryfmt: added check for short ..	67	\cGLS: new	112
\glslongpltok: new	202	\cGLSformat: new	112
\glsshortpltok: new	202	\cGLSpl: new	113
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	113
name in \setkeys	204	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	18
for plural	199	\glsenableentrycount: new	108
\GLSxtrlongpl: new	218	\glsfirstabbrvdefaultfont: new ..	208
\Glsxtrlongpl: new	218	\glsfirstlongdefaultfont: new ...	208
\glsxtrlongpl: new	217	\Glsfmtfirst: new	334
\glsxtrNoGlossaryWarning: new	23	\glsfmtfirst: new	334
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	335
new	198	\glsfmtfirstpl: new	334
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	334
new	199	\glsfmtplural: new	333
\glsxtrpostlinkendsentence: new ..	198	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	217	\Glsxtrtitleshort	332
\Glsxtrshortpl: new	216	renamed from \glstentryfmtshort ..	332
\glsxtrshortpl: new	215	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	331
\glslabeltok	229	renamed from \glstentryfmtshort ..	331
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	227	\Glsxtrtitleshortpl	332
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	332
redefinition of \acronymtype	20	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	331
\Glsxtrshort	332	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	331
\glsxtrshort	331	\Glsfmttext: new	333
\Glsfmtshortpl: changed to use		\glsfmttext: new	333
\glsxtrshortpl	332	\glshasattribute: new	177
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	176
\glsxtrshortpl	331	\glsxtremsuffix: new	270
\glsxtrifemptyglossary: new	31	\GlsXtrEnableEntryCounting: new ..	107
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	110
argument	180	\glsxtrscfont: new	242
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	242
argument	180	\glsxtrsmfont: new	256
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	257
default type to \acronymtype	121	short-em: new	278
\newterm: fixed name argument	179	short-em-desc: new	279
0.5 (2015-12-07)		short-em-footnote: new	288
\@cGLS: new	112	short-em-long: new	274
\@cGLS@: new	112	short-em-long-desc: new	275

short-em-postfootnote: new	290	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	253	attribute	322
short-sc-postfootnote: new	254	\Glsxtrheadtext: now uses headuc	
short-sm: new	260	attribute	325
short-sm-desc: new	261	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	267	attribute	324
short-sm-long: new	258	short-em-footnote: switch off regular	
short-sm-long-desc: new	259	attribute if set	289
short-sm-postfootnote: new	268	short-long: switch off regular attribute	
long-noshort-em: new	281	if set	228
long-noshort-em-desc: new	285	short-long-desc: switch off regular	
long-noshort-sm: new	263	attribute if set	229
long-noshort-sm-desc: new	265	short-sc-footnote: switch off regular	
long-short-em: new	271	attribute if set	253
long-short-em-desc: new	272	short-sm-footnote: switch off regular	
long-short-sm: new	257	attribute if set	267
long-short-sm-desc: new	258	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	226
\Glsaccessstext: new	163	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	227
\@glxtr@doaccsupp: new	23	long-short-sc-desc: switch off regular	
General: removed \ifglxtruseuchead	322	attribute if set	244
\Glsaccessdesc: new	166	footnote: switch off regular attribute if	
\Glsaccessdescplural: new	167	set	230
\Glsaccessfirst: new	164	postfootnote: switch off regular	
\Glsaccessfirstplural: new	164	attribute if set	232
\Glsaccessname: new	162	0.5.2 (2015-12-08)	
\Glsaccessplural: new	163	\@GLSdesc@: added accessibility support	79
\Glsaccesssymbol: new	165	\@GLSdescplural@: added accessibility	
\Glsaccesssymbolplural: new	165	support	80
\glxtrheadfirst: now uses headuc		\@GLSfirst@: added accessibility	
attribute	326	support	77
\glxtrheadfirst: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	326	support	78
\Glsxtrheadfirstplural: now uses		\@GLSname@: added accessibility support	79
headuc attribute	327	\@GLSplural@: added accessibility	
\glxtrheadfirstplural: now uses		support	77
headuc attribute	327	\@GLSsymbol@: added accessibility	
\Glsxtrheadplural: now uses headuc		support	80
attribute	326	\@GLSsymbolplural@: added	
\glxtrheadplural: now uses headuc		accessibility support	81
attribute	325	\@GLStext@: added accessibility support	76
\Glsxtrheadshort: now uses headuc		\@GLSdesc@: added accessibility support	79
attribute	323	\@GLSdescplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	80
attribute	322	\@GLSfirst@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	77
attribute	323	\@GLSfirstplural@: added accessibility	
		support	78

\@Glsname@: add accessibility support ..	79	\GLSaccesssymbolplural: new ..	166, 174
\@Glsplural@: added accessibility support	77	\GLSaccessstext: new	163, 172
\@Glsymbol@: added accessibility support	80	\glstentryfmt: moved	
\@Glsymbolplural@: added accessibility support	81	\glstetabbrvfmt from	
\@Glstext@: added accessibility support	76	\glstxtrabbrvfmt to here	67
\@glstext@: added accessibility support	79	\GlsXtrEnableInitialTagging: new	194
\@glstextplural@: added accessibility support	79	\glstxtrfieldtitlecase: new	181
\@glstext@: added accessibility support	76	\GlsXtrFormatLocationList: new ...	64
\@glstextplural@: added accessibility support	78	\glstxtrnewabbrevpresetkeyhook:	
\@Glsname@: added accessibility support	79	new	206
\@Glsplural@: added accessibility support	77	\glstxtrtagfont: new	196
\@Glsymbol@: added accessibility support	80	\KV@printgloss@nonumberlist: added	66
\@Glsymbolplural@: added accessibility support	80	\mfu@checkword@do: added	195
\@Glstext@: added accessibility support	76	\setabbreviationstyle: added check	
\@glstxtr@activate@initialtagging:		for post-definition style switch	222
new	196	0.5.3 (2015-12-09)	
\@glstxtr@do@titlecaps@warn: new ..	196	\@glstxtr@autoindex@at: new	192
\@glstxtr@tag: new	196	\@glstxtr@autoindex@encap: new ...	192
General: fixed typo in glossaries-accsupp		\@glstxtr@autoindex@esc: new	193
and tidied up code to use just one		\@glstxtr@autoindex@level: new ...	192
\@ifpackageloaded	162	\@glstxtr@autoindex@setname: new ..	190
removed \glstxtrabbrvfmt	219	\@glstxtr@doabbreviationsdef: new ..	20
\glossaryentrynumbers: added	64	General: removed	
\Glossentrydesc: added	194	\GlsXtrNoGlsWarningNoAutoMakeMain	
\Glossentryname: added	185	138
\Glossentrysymbol: added	194	\glstxtr@descwidth: added	63
\GLSaccessdesc: new	166, 174	\glstxtr@pagelistwidth: added	64
\GLSaccessdescplural: new ...	167, 174	\glstxtr@doautoindexname: new	190
\GLSaccessfirst: new	164, 173	\glstxtr@postnamehook: new	187
\GLSaccessfirstplural: new ..	164, 173	\if@glstxtr@format@override: new ..	189
\GLSaccesslong: new	168, 175	\ProvidesGlossariesExtraLang: new	338
\GLSaccesslongpl: new	169, 175	\RequireGlossariesExtraLang: new	337
\Glsaccesslongpl: new	169	0.5.4 (2015-12-15)	
\glstxtr@longpl: new	169	\@newglossaryentry@defunitcounters:	
\GLSaccessname: new	162, 172	new	114
\GLSaccessplural: new	163, 173	\@GLSxtr@p@acrlong@: new	99
\GLSaccessshort: new	167, 174	\@GLSxtr@p@acrlongpl@: new	99
\GLSaccessshortpl: new	168, 175	\@GLSxtr@p@acrshort@: new	98
\GLSaccesssymbol: new	165, 173	\@GLSxtr@p@acrshortpl@: new	99
		\@GLSxtr@p@long@: new	98
		\@GLSxtr@p@longpl@: new	98
		\@GLSxtr@p@plural@: new	97
		\@GLSxtr@p@short@: new	97
		\@GLSxtr@p@shortpl@: new	98
		\@GLSxtr@p@text@: new	97
		\@GlsXtrEnableOnTheFly: new	60
		\@Glsxtr: new	60
		\@Glsxtr@p@acrlong@: new	99
		\@Glsxtr@p@acrlongpl@: new	99

\@Glsxtr@p@acrshort@: new	98	\glsenableentryunitcount: new	116
\@Glsxtr@p@acrshortpl@: new	99	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	98	entry's existence	177
\@Glsxtr@p@longpl@: new	98	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	97	entry's existence	177
\@Glsxtr@p@short@: new	97	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	98	check	197
\@Glsxtr@p@text@: new	97	\Glsxtr: new	60
\@Glsxtrpl: new	61	\glxtr: new	60
\@alt@glshyp@opt: new	93	\glxtrcat: new	60
\@glscalt@hyp@opt: new	92	\glxtrdowrglossaryhook: new	92
\@glscalt@hyp@opt@char: new	93	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	93	new	119
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	59
new	114	\Glsxtrpl: new	61
\@glsc@local@increment@currunitcount:		\glxtrpl: new	61
new	115	\glxtrpostlocalreset: new	107
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	106
new	90	\glxtrpostreset: new	106
\@glsxtr: new	60	\glxtrpostunset: new	105
\@glsxtr@addunitcounter: new	114	\glxtrprotectlinks: new	96
\@glsxtr@currunitcount: new	115	\GlsXtrSetAltModifier: new	93
\@glsxtr@ifunitcounter: new	114	\GlsXtrSetDefaultGlsOpts: new	91
\@glsxtr@p@acrlong@: new	99	\glxtrstarflywarn: new	60
\@glsxtr@p@acrlongpl@: new	99	\GlsXtrWarning: new	61
\@glsxtr@p@acrshort@: new	98	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new	99	disables \setacronymstyle	121
\@glsxtr@p@long@: new	98	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new	98	\@glsxtr@autoindexcrossrefs: new	17
\@glsxtr@p@plural@: new	97	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new	97	new	130
\@glsxtr@p@shortpl@: new	97	\@glsxtr@idx@entrynumberlist: new	131
\@glsxtr@p@text@: new	97	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new	115	new	130
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new	119	new	131
\@glsxtr@unitcountlist: new	114	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new	61	new	130
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new	122
empty see value if not set and added		\makeglossaries: new	122
'see' to field key map	49	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	59	\glxtrdiscardperiod: added check	
\cGlsformat: added	113	for first use	199
\cGlsformat: added	113	short-desc: fixed typo in	
\cGlsplformat: added	113	\glxtrinlinfullformat and	
\cGlsplformat: added	113	added missing second argument	236
\glstdisablehyper: added	95	1.02 (2016-04-25)	
\glstdohyperlink: added	94	\@glsxtr@current@style: new	62
\glstdonohyperlink: added	96	\Glsfmtfull: new	336

\glsfmtfull: new	336	\@GLSdescplural@: set abbreviation and regular format	80
\Glsfmtfullpl: new	337	\@GLSfirst@: set abbreviation format ..	77
\glsfmtfullpl: new	337	\@GLSfirstplural@: set abbreviation and regular format	78
\Glsfmtlong: new	335	\@GLSname@: set abbreviation and regular format	79
\glsfmtlong: new	335	\@GLSplural@: set abbreviation and regular format	77
\Glsfmtlongpl: new	336	\@GLSsymbol@: set regular format	80
\glsfmtlongpl: new	336	\@GLSsymbolplural@: set regular format	81
\Glsxtrheadfull: new	330	\@GLStext@: set abbreviation and regular format	76
\glsxtrheadfull: new	329	\@GLSuseri@: set regular format	81
\Glsxtrheadfullpl: new	331	\@GLSuserii@: set regular format	82
\glsxtrheadfullpl: new	330	\@GLSuseriii@: set regular format	82
\Glsxtrheadlong: new	328	\@GLSuseriv@: set regular format	82
\glsxtrheadlong: new	328	\@GLSuservi@: set regular format	83
\Glsxtrheadlongpl: new	329	\@GLSdesc@: set abbreviation and regular format	79
\glsxtrheadlongpl: new	328	\@GLSdescplural@: set abbreviation and regular format	80
\Glsxtrtitlefull: new	330	\@GLSfirst@: set abbreviation and regular format	77
\glsxtrtitlefull: new	330	\@GLSfirstplural@: set abbreviation and regular format	78
\Glsxtrtitlefullpl: new	331	\@GLSname@: set abbreviation and regular format	79
\glsxtrtitlefullpl: new	330	\@GLSplural@: set abbreviation and regular format	77
\Glsxtrtitlelong: new	329	\@GLSsymbol@: set regular format	80
\glsxtrtitlelong: new	328	\@GLSsymbolplural@: set regular format	81
\Glsxtrtitlelongpl: new	329	\@GLStext@: set abbreviation and regular format	76
\glsxtrtitlelongpl: new	328	\@GLSuseri@: set regular format	81
\ifglxtrinsertinside: new	225	\@GLSuserii@: set regular format	81
postfootnote: added redef of		\@GLSuseriii@: set regular format	82
\glsxtrsetupfulldefs	233	\@GLSuseriv@: set regular format	82
stylemods: new	24	\@GLSuservi@: set regular format	83
1.03 (2016-04-27)		\@GLSdesc@: set abbreviation and regular format	79
\@GLSfirstplural@: bug fix: misspelt cs name	78	\@GLSdescplural@: set abbreviation and regular format	77
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	77	\@GLSsymbol@: set regular format	80
\@GLSfirstplural@: bug fix: misspelt cs name	78	\@GLSsymbolplural@: set regular format	81
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	77	\@GLStext@: set abbreviation and regular format	76
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	77	\@GLSuseri@: set regular format	81
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl ..	328	\@GLSuserii@: set regular format	81
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl	323	\@GLSuseriii@: set regular format	82
1.04 (2015-04-30)		\@GLSuseriv@: set regular format	82
short-em-footnote: renamed from “footnote-em”	288	\@GLSuservi@: set regular format	83
1.04 (2016-05-02)		\@gls@preglossaryhook: added check for entry’s existence	196
\@glsxtrpostloctag: new	66	\@GLSdesc@: set abbreviation and regular format	79
\@GLSdesc@: set abbreviation and regular format	79	\@GLSdescplural@: set abbreviation and regular format	79
		\@GLSfirst@: set abbreviation and regular format	76

\@glsfirstplural@: set abbreviation and regular format	78	\glsxtruserparen: new	292
\@glsname@: set abbreviation and regular format	79	\glsxtrusersuffix: new	293
\@glsplural@: set abbreviation and regular format	77	\GlsXtrWarnDeprecatedAbbrStyle: new	224
\@glsymbol@: set regular format	80	short-em-long-em: new	276
\@glsymbolplural@: set regular format	80	short-em-long-em-desc: new	277
\@glstext@: set abbreviation and regular format	76	short-em-nolong: new	279
\@glsxtr@deprecated@abbrstyle: new	224	short-em-nolong-desc: new	280
\@glsxtr@do@style: new	25	short-em-postfootnote: renamed from “postfootnote-em”	290
\@glsxtr@doloctag: new	66	short-footnote: new	232
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand	131	short-long-user: new	299
\@glsxtr@pagetag: new	66	short-long-user-desc: new	301
\@glsxtr@pagetag: new	66	short-nolong: new	235
\@glsxtr@preloctag: new	66	short-nolong-desc: new	237
\@glsxtr@preloctag: new	65	short-postfootnote: new	234
\glossentrydesc: added glossdescfont attribute check	181	short-sc-footnote: renamed from “footnote-sc”	253
\Glossentryname: added glossnamefont attribute check	185	short-sc-nolong: new	247
\glossentryname: added glossnamefont attribute check	183	short-sc-nolong-desc: new	249
moved post name hook inside condition	185	short-sc-postfootnote: renamed from “postfootnote-sc”	254
\glsabbrvmfont: new	270	short-sm-footnote: renamed from “footnote-sm”	267
\glsabbrvuserfont: new	292	short-sm-nolong: new	261
\glsfirstabbrvmfont: new	270	short-sm-nolong-desc: new	263
\glsfirstabbrvuserfont: new	292	short-sm-postfootnote: renamed from “postfootnote-sm”	268
\glsfirstlongemfont: new	271	\letabbreviationstyle: new	224
\glsfirstlonguserfont: new	293	\newabbreviationstyle: bug fix: corrected test for existence	223
\glsifnotregularcategory: new ...	178	long-em-noshort-em: new	283
\glslongdefaultfont: new	208	long-em-noshort-em-desc: new	286
\glslongemfont: new	271	long-em-short-em: new	272
\glslongfont: new	208	long-em-short-em-desc: new	273
\glslonguserfont: new	293	long-noshort: new	241
\glsxtrassignfieldfont: new	75	long-noshort-desc: new	240
\GlsXtrEnablePreLocationTag: new .	65	long-noshort-em: renamed from “long-em”	281
\glsxtrfirstscfont: new	242	long-noshort-em-desc: renamed from “long-desc-em”	285
\glsxtrfirstsmfont: new	256	long-noshort-sc: renamed from “long-sc”	249
\glsxtrlongshortdescsort: new ...	227	long-noshort-sc-desc: renamed from “long-desc-sc”	251
\glsxtrpostnamehook: added category check	187	long-noshort-sm: renamed from “long-sm”	263
\glsxtrregularfont: new	67	long-noshort-sm-desc: renamed from \long-desc-sm	265
\glsxtruserfield: new	292		

long-short-user: new	293	docdef option changed to choice	16
long-short-user-desc: new	299	\glxtr@usesee: new	49
\renewabbreviationstyle: new	224	\glxtrusesee: new	49
style: new	25	\glxtruseseeformat: new	49
1.05 (2016-06-10)		\if@glxtrdocdefrestricted: new ..	17
\eglssetwidest: new	402	1.07 (2016-08-15)	
\glFindWidestAnyName: new	404	\@@glxtrp: new	99
\glFindWidestAnyNameLocation:		\@GLSfirst@: added check for	
new	410	nohyperfirst attribute	77
\glFindWidestAnyNameSymbol: new	407	\@GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute	78
new	409	\@GLSxtrp: new	100
\glFindWidestLevelTwo: new	406	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	404	nohyperfirst attribute	77
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new	409	nohyperfirst attribute	78
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new	100
new	407	\@gls@preglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts	197
new	408	\@glsfirst@: added check for	
\glFindWidestUsedLevelTwo: new .	405	nohyperfirst attribute	76
\glFindWidestUsedTopLevelName:		\@glsfirstplural@: added check for	
new	403	nohyperfirst attribute	78
\glsfirstlongfootnotefont: new ..	230	\@glxtrinmark: new	320
\glsgetwidestname: new	403	\@glxtrnotinmark: new	320
\glsgetwidestsubname: new	403	\@glxtrp: new	100
\glslongfootnotefont: new	230	\@glxtrp@opt: new	99
\glxtrAltTreeIndent: new	401	\glossxtrsetpopts: new	99
\glxtralttreeInit: new	402	\glsp: new	102
\glxtrAltTreePar: new	401	\glsp: new	102
\glxtrAltTreeSetHangIndent: new	411	\glxtr@entry@p: new	101
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new	230
new	411	\glxtrchecknohyperfirst: new	76
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new	181
new	401	\glxtrifinmark: new	319
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new	103
new	401	\Glsxtrp: new	102
\glxtrComputeTreeIndent: new ...	410	\glxtrp: new	101
\glxtrComputeTreeSubIndent: new	411	\glxtrsetpopts: new	99
\glxtrtreetopindent: new	402	short-long-desc: added text key	229
short-em-long: fixed incorrect font used		fixed misspelling of \glsabbrvfont in	
by long form	275	plural key	229
\xglsetwidest: new	402	long-short-desc: added missing text	
1.06 (2016-06-18)		key	227
\@glsdoifexistsorwarn: new	17	fixed misspelling of \glsabbrvfont .	227
\@glxtr@docdefval: new	16	footnote: changed first forms to use	
\@glxtr@usesee: new	49	\glsfirstlongfootnotefont ...	230
General: disabled docdef key at the start		postfootnote: removed \footnote	
of the document	30	from first keys	232

switched from \glsfirstlongfont to \glsfirstlongfootnotefont ...	233	1.10 (2016-12-17)	\@GLSp1@: fixed bug caused by typo in command name	68
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse	122	1.11 (2017-01-19)	\@glsxtr@do@redef@forglsentries: new	6
1.08 (2016-12-13)			\@glsxtr@noidx@do: new	151
\@glsxtr@record: new	8		\@glsxtr@redef@forglsentries: new ..	6
\@GLS@: added \@glsxtr@record	68		\@glsxtr@shortcutsval: new	22
\@GLSp1@: added \@glsxtr@record ...	68		\@glsxtr@unsrt@getgrouptitle: new	150
\@Gls@: added \@glsxtr@record	68		\@print@noidx@glossary: added redefinition	133
\@Glspl@: added \@glsxtr@record ...	68		\glsxtr@addloclistfield: added group key	14
\@gls@: added \@glsxtr@record	68		added location key	14
\@gls@@link@: added \@glsxtr@record	69		\glsxtr@fields: new	142
\@gls@field@link: added \@glsxtr@record	67		\glsxtr@linkprefix: new	142
\@gls@saveentrycounter: new	29		\glsxtr@org@newignoredglossary: new	44
\@glsdisp: added \@glsxtr@record ..	69		\glsxtr@s@newignoredglossary: new	45
\@glspl@: added \@glsxtr@record ...	68		\glsxtr@shortcutsval: new	142
\@glsxtr@dorecord: new	10		\glsxtr@texencoding: new	142
\@glsxtr@err@undefaction: new	6		\glsxtr@writefields: new	142
\@glsxtr@record: new	7		\GlsXtrLoadResources: new	142
\@glsxtr@warn@onexistsordo: new ...	6		\glsxtrpageref: new	41
\@glsxtr@warn@undefaction: new	6		\glsxtrresourcefile: changed extension to .glstex	141
\@print@unsrt@glossary: new	147		\newignoredglossary: added starred version	44
record: added record package option ...	15	1.12 (2017-02-03)		
\glsadd: added \@glsxtr@record	74		\@glsxtr@recordcounter: new	12
\glsdoifexists: now defines \glslabel	47		\@gls@preglossaryhook: check for definition	196
\glsxtr@do@wrglossary: new	29		\@glsxtr@counterrecordhook: new ..	144
\glsxtr@addloclistfield: new	13		\@glsxtr@display@loc: new	134
\glsxtr@indexonly@saveentrycounter: new	13		\@glsxtr@docounterrecord: new ...	145
\glsxtr@record: new	144		\@glsxtr@longnewglossaryentry: new	43
\glsxtr@resource: new	142		\@glsxtr@noop@recordcounter: new ..	12
\glsxtr@saveentrycounter: new	29		\@glsxtr@op@recordcounter: new ...	13
\glsxtr@setup@record: new	13		\@glsxtr@provide@storagekey: new ..	31
\glsxtrassignfieldfont: added check for existence	75		\@glsxtr@s@longnewglossaryentry: new	43
\glsxtrresourcefile: new	141		\@glsxtrentryfmt: new	34
\printunsrtglossaries: new	147		\@glsxtrindexaliased: new	91
\printunsrtglossary: new	147		\@glsxtrsetaliasnoindex: new	90
1.09 (2016-12-16)			\@newglossaryentryposthook: added check for alias key	55
\@glsxtr@gettype: new	129		\@no@glsxtrindexaliased: new	91
\@glsxtr@mixed@assign@sortkey: new	129			
\@printglossary: redefined to save options	128			
\glsxtr@makeglossaries: new	129			

\@printunsrtglossary: new	147	\GlsXtrLoadResources: removed	
General: added target key to printgloss		restriction on only one per document	142
family	128	\glxtrlocrangefmt: new	135
\apptoglossarypreamble: new	42	\glxtrpostlongdescription: new	44
\csGlsXtrLetField: new	38	\glxtrprovidestoragekey: new	31
\eGlsXtrSetField: new	38	\GlsXtrRecordCounter: new	144
\gGlsXtrSetField: new	38	\glxtrresourcecount: new	142
\glsdohyperlink: added check for alias		\glxtrresourcefile: added catcode	
field	94	change for @	141
\glsnoidxdisplayloc: added		\glxtrsetaliasnoindex: new	90
redefinition	133	\GlsXtrSetField: new	38
\glissettoctitle: added patch	45	\glxtrsetfieldifexists: new	37
\glxtr@counterrecord: new	144	\glxtrunsrtdo: new	150
\glxtr@langtag: new	142	\GlsXtrusefield: new	37
\glxtr@newabbreviation: new	204	\glxtrusefield: new	37
\glxtr@org@newignoredglossary:		short-postlong-user: new	296
Added check for existence	44	short-postlong-user-desc: new	298
\glxtr@pluralsuffixes: new	142	\longnewglossaryentry: added starred	
\glxtr@provideignoredglossary:		version	43
new	46	long-postshort-user: new	294
\glxtr@s@newignoredglossary:		long-postshort-user-desc: new	296
Added check for existence	45	postdot: new	18
\glxtr@s@provideignoredglossary:		\pretoglossarypreamble: new	42
new	46	\print@noop@unsrtglossaryunit:	
\glxtrabbrvpluralsuffix: new	208	new	150
\glxtralias: new	55	\print@op@unsrtglossaryunit: new	149
\glxtrcopytoglossary: new	47	\printunsrtglossary: added starred	
\glxtrdeffield: new	37	form	147
\glxtrdisplayendloc: new	134	\printunsrtglossaryhandler: new	149
\glxtrdisplayendloohook: new	135	\printunsrtglossaryunit: new	13
\glxtrdisplayingleloc: new	134	\printunsrtglossaryunitsetup: new	149
\glxtrdisplaystartloc: new	134	\provideignoredglossary: new	46
\glxtrdeffield: new	37	\s@glxtr@provide@storagekey: new	32
\glxtrentryfmt: new	34	\s@printunsrtglossary: new	147
\glxtrfieldddolistloop: new	35	\xGlsXtrSetField: new	38
\glxtrfieldforlistloop: new	35		
\glxtrfielddifylist: new	35	1.13 (2017-02-07)	
\glxtrfieldlistadd: new	34	\@glsdisp: removed	
\glxtrfieldlistead: new	34	\@glxtr@org@glsdisp	69
\glxtrfieldlistgadd: new	34	\glxtrsetaliasnoindex: switched to	
\glxtrfieldlistxadd: new	34	\providecommand	90
\glxtrfieldxifylist: new	35		
\glxtrfmt: new	32	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	32	\@gls@link: added redefinition	72
\GlsXtrFmtField: new	32	\@gls@noidx@getgrouptitle: new	131
\glxtrifkeydefined: new	31	\@gls@removespaces: new	135
\glxtrindexaliased: new	91	\@glxtr@do@automake@err: new	144
\GlsXtrLetField: new	38	\@glxtr@org@gloautosee: new	28
\GlsXtrLetFieldToField: new	38	\@glxtr@record: added third arg	7
		\@glxtr@recordsee: new	13

General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	74	\@glo@autosee: added redefinition	28
added \glsadd option thevalue	74	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	95	bug	131
\glsenableentrycount: fixed		\@glsxtr@addunusedxrefs: added	
assignment of \@cGls@	109	check for seealso field	56
\glsenableentryunitcount: fixed		\@glsxtr@checkgroup: use \csuse	
assignment of \@cGls@	117	instead of \csname	151
\glsnavigation: new	132	\@glsxtr@dorecordnodefer: new	11
\glsxtr@org@getgrouptitle: new ..	132	\@glsxtr@record@only@setup: added	
\glsxtr@recordsee: new	7	check for \@gls@setupsort@none ..	15
\glsxtr@writefields: added check for		\@print@unsrt@glossary: corrected	
automake	144	misspelt command	148
\glsxtrdisplayendloc: added check		\@printunsrt@glossary@handler:	
for empty format	134	new	149
\glsxtrgetgrouptitle: new	132	\gls@checkseeallowed: added	
\glsxtrinitwrgloss: new	69	redefinition	28
\glsxtrlocationhyperlink: new ...	135	\glsxtr@writefields: added	
\glsxtrsetgrouptitle: new	132	\providecommand lines	143
\glsxtrsupphypertext: new	136	\glsxtrautoindex: new	190
\ifglsxtrwrglossbefore: new	70	\glsxtrautoindexassignsort: new .	191
1.15 (2017-05-10)		\glsxtrautoindexentry: new	190
\@glsxtr@dorecord: corrected		\glsxtrindexseealso: new	52
premature expansion of \@glslocref	10	\glsxtrseealsolabels: new	55
short-em-long-em: fixed spelling of		\glsxtrseelist: new	52
\glsabbrvfont	276	\glsxtruseseealso: new	51
short-long: fixed spelling of		\glsxtruseseealsoformat: new	52
\glsabbrvfont	228	\seealso name: new	52
short-long-user: fixed spelling of		autoseeindex: new	17
\glsabbrvfont	300	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@glsxtr@mark@wordseps: new	203
\glsabbrvfont	297	\@glsxtr@markwordseps: new	203
short-postlong-user-desc: fixed		\@glsxtr@noidx@displaynumberlist:	
spelling of \glsabbrvfont	298	replace hard-coded ?? with	
long-em-short-em: fixed spelling of		\glsxtrundeftag	130
\glsabbrvfont	273	\@glsxtr@noidx@entrynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	294	\glsxtrundeftag	131
long-postshort-user-desc: fixed		\@glsxtr@noidx@numberlistloop:	
spelling of \glsabbrvfont	296	replace hard-coded ?? with	
long-short: fixed spelling of		\glsxtrundeftag	131
\glsabbrvfont	226	\@glsxtrifhyphenstart: new	302
long-short-user: fixed spelling of		General: removed some inconsistencies	
\glsabbrvfont	293	in the abbreviation styles	225
footnote: fixed spelling of		\glsabbrvhyphenfont: new	302
\glsabbrvfont	230	\glsabbrvonlyfont: new	315
postfootnote: fixed spelling of		\glsabbrvscfont: new	242
\glsabbrvfont	232	\glsabbrvsmfont: new	256

\glsabbrvuserfont: initialised to default font	292	short-long-user-desc: corrected first forms	301
\glsfirstabbrvhyphenfont: new ...	302	short-nolong-desc-noreg: new ...	237
\glsfirstabbrvonlyfont: new	315	short-nolong-noreg: new	235
\glsfirstabbrvscfont: new	242	long-em-noshort-em-desc-noreg: new	288
\glsfirstabbrvsmfont: new	256	long-em-noshort-em-noreg: new ...	284
\glsfirstlonghyphenfont: new ...	302	long-hyphen-noshort-desc-noreg: new	304
\glsfirstlongonlyfont: new	316	long-hyphen-postshort-hyphen: new	308
\glslonghyphenfont: new	302	long-hyphen-postshort-hyphen-desc: new	309
\glslongonlyfont: new	315	long-hyphen-short-hyphen: new ...	303
\glslonguserfont: initialised to default font	293	long-hyphen-short-hyphen-desc: new	303
\glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong	204	long-noshort-desc-noreg: new ...	240
\GlsXtrDefineAcShortcuts: new ...	21	long-noshort-noreg: new	241
\glsxtrgenabbrvfmt: added check for \ifglsxtrininsertinside	219	long-only-short-only: new	316
\glsxtrhyphensuffix: new	302	long-only-short-only-desc: new ..	317
\glsxtrifhyphenstart: new	301	long-short-user-desc: corrected first forms	299
\glsxtrlonghyphen: new	307	1.18 (2017-08-10)	
\glsxtrlonghyphennoshort: new ...	304	stylemods: changed default value to "default"	24
\glsxtrlonghyphenshort: new	302	1.19 (2017-09-09)	
\glsxtrlongshortdescname: new ...	227	\@glsxtr@defaultnumberformat: new .	7
\glsxtronlydescname: new	317	\@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref	10
\glsxtronlydescsort: new	317	\@glsxtr@dorecordnodefer: Use \theglstrycounter for the location rather than \@glslocref ..	11
\glsxtronlysuffix: new	316	\@glsxtr@record@setting: new	14
\glsxtrparen: new	206	\@glsxtr@record@setting@alsoindex: new	14
\glsxtrposthyphenlong: new	312	\@glsxtrifhasfield: new	35
\glsxtrposthyphenshort: new	307	General: added \glslink option theHvalue	70
\glsxtrposthyphensubsequent: new	307	added \glslink option thevalue ...	70
\glsxtrshortdescname: new	236	\glsxtr@writefields: removed double-quotes around \jobname ..	144
\glsxtrshorthyphen: new	312	\glsxtrdoautoindexname: changed format test	190
\glsxtrshorthyphenlong: new	310	\glsxtrhyperlink: new	95
\glsxtrshortlongdescname: new ...	229	\glsxtrifhasfield: new	35
\glsxtrshortlongdescsort: new ...	229	\GlsXtrSetDefaultNumberFormat: new	7
\Glsxtrsubsequentfmt: new	222	\s@glsxtrifhasfield: new	36
\glsxtrsubsequentfmt: new	221		
\Glsxtrsubsequentplfmt: new	222		
\glsxtrsubsequentplfmt: new	222		
\glsxtrword: new	203		
\glsxtrwordsep: new	203		
short-hyphen-long-hyphen: new ...	311		
short-hyphen-long-hyphen-desc: new	311		
short-hyphen-postlong-hyphen: new	313		
short-hyphen-postlong-hyphen-desc: new	315		

1.20 (2017-09-11)			
\@glxstrhypernameprefix: new	129	
\glsdohypertarget: added redefinition		129	
\printunsertglossaryunitsetup:			
switched from redefining			
\glolinkprefix to			
\@glxstrhypernameprefix	150	
1.21 (2017-11-03)			
\@@glxstr@record: added check for			
default options	9	
\@@glxstrwrglossmark: new	26	
\@glslink: changed \let to \def	96	
\@glxstr@checkgroup: new	150	
\@glxstr@defpostpunc: new	18	
\@glxstr@do@record@wrglossary:			
new	8	
\@glxstr@dosee@alsoindex@glossary:			
new	28	
\@glxstr@doseeglossary: new	27	
\@glxstr@noidx@do: removed code			
dealing with the group	152	
\@glxstr@record@setting@off: new	.	15	
\@glxstr@record@setting@only: new		14	
\@glxstr@rglstrigger@record: new		156	
\@glxstrglossentry: new	145	
\@glxstrnewgls: new	153	
\@glxstrsetaliasnoindex: changed to			
use \glxstrifhasfield instead of			
\ifglshasfield	90	
\@glxstrwrglossmark: new	26	
\@rGLS: new	158	
\@rGLS@: new	158	
\@rGLSpl: new	159	
\@rGLSpl@: new	159	
\@rGls: new	158	
\@rGls@: new	158	
\@rGlspl: new	158	
\@rGlspl@: new	158	
\@rgls: new	157	
\@rgls@: new	157	
\@rglspl: new	157	
\@rglspl@: new	157	
General: adjusted mcolalttree	416	
modified index to remove hard coded			
\space	396	
modified list to remove hard coded			
\space	385	
moved conditional outside of			
\glsgroupskip	388–395	
new	419	
redefined altlistgroup to discourage			
breaks after group headings	386	
redefined altlisthypergroup to			
discourage breaks after group			
headings	387	
redefined alttreegroup to discourage			
breaks after group headings	412	
redefined alttreehypergroup to			
discourage breaks after group			
headings	413	
redefined indexgroup to discourage			
breaks after group headings	397	
redefined indexhypergroup to			
discourage breaks after group			
headings	397	
redefined listgroup to discourage			
breaks after group headings	386	
redefined listhypergroup to			
discourage breaks after group			
headings	386	
redefined mcolalttreegroup to			
discourage breaks after group			
headings	417	
redefined mcolalttreehypergroup to			
discourage breaks after group			
headings	417	
redefined mcolalttreespannav to			
discourage breaks after group			
headings	418	
redefined mcolindexgroup to			
discourage breaks after group			
headings	413	
redefined mcolindexhypergroup to			
discourage breaks after group			
headings	413	
redefined mcolindexspannav to			
discourage breaks after group			
headings	414	
redefined mcoltreegroup to			
discourage breaks after group			
headings	414	
redefined mcoltreehypergroup to			
discourage breaks after group			
headings	415	
redefined mcoltreenamegroup to			
discourage breaks after group			
headings	415	
redefined			

mcoltreenonamehypergroup to		\glxstrbookindexcols: new	419
discourage breaks after group		\glxstrbookindexcolspread: new ..	420
headings	416	\glxstrbookindexfirstmark: new ..	424
redefined mcoltreenonamespannav to		\glxstrbookindexfirstmarkfmt: new	424
discourage breaks after group		\glxstrbookindexformatheader: new	420
headings	416	\glxstrbookindexgroupskip: new ..	420
redefined mcoltreespannav to		\glxstrbookindexlastmark: new ...	424
discourage breaks after group		\glxstrbookindexlastmarkfmt: new	424
headings	415	\glxstrbookindexmarkentry: new ..	424
redefined treegroup to discourage		\glxstrbookindexname: new	419
breaks after group headings	398	\glxstrbookindexparentchildsep:	
redefined treehypergroup to		new	419
discourage breaks after group		\glxstrbookindexparentschildsep:	
headings	399	new	419
redefined treenonamegroup to		\glxstrbookindexprelocation: new	419
discourage breaks after group		\glxstrbookindexsubatendgroup:	
headings	400	new	420
redefined treenonamehypergroup to		\glxstrbookindexsubbetween: new ..	420
discourage breaks after group		\glxstrbookindexsubname: new	419
headings	400	\glxstrbookindexsubprelocation:	
debug: new	26	new	419
\gglsetwidest: new	402	\glxstrbookindexsubsubatendgroup:	
\glsdisablehyper: added check for		new	420
existence	95	\glxstrbookindexsubsubbetween:	
changed to use \def rather than \let .	95	new	420
\glsenablehyper: changed to use \def		\glxstrbookindexthepage: new	423
rather than \let	95	\glxstrdetoklocation: new	155
\Glsfmtname: new	332	\glxtrenablerecordcount: new ...	155
\glsfmtname: new	332	\glxstrglossentry: new	145
\glshex: new	339	\glxstrgroupfield: new	150
\glslstchildpostlocation: new ..	385	\Glsxtrheadname: new	324
\glslstchildprelocation: new ...	385	\glxtrheadname: new	323
\glslstprelocation: new	385	\GlsXtrIfFieldEqStr: new	38
\glsnahyperlink: patched	93	\glxstriflabelinlist: new	149
\glssseeitemformat: new	49	\glxstrifrecordtrigger: new	156
\glsshowtarget: new	27	\glxstrindexseealso: added check	
\glstreechildprelocation: new ...	396	that the entry exists	52
\glstreeprelocation: new	396	\glxstrinithyperoutside: new	70
\glstriggerrecordformat: new	157	\GlsXtrLocationRecordCount: new ..	155
\glseuseabbrvfont: new	219	\glxtrnewgl: new	152, 154
\glseuselongfont: new	219	\glxtrnewGLSlike: new	154
\glxstr@do@alsoindex@wrglossary:		\glxtrnewglslke: new	154
new	8	\glxtrnewrgls: new	154
\glxstr@org@@do@wrglossary: new ..	29	\glxtrnewrGLSlike: new	154
\glxstr@org@dohyperlink: new	93	\glxtrnewrglslike: new	154
\glxstr@setbookindexmark: new ...	424	\glxstrprelocation: new	384, 419
\glxstrbookindexatendgroup: new ..	420	\GlsXtrRecordCount: new	155
\glxstrbookindexbetween: new	420	\glxstrrecordtriggervalue: new ..	155
\glxstrbookindexbookmark: new ...	420		

\glxtrresourcefile: now disables record key	141	\glossentrynameother: new	188
\glxtrresourceinit: new	142	\glseeitemformat: switched check from regular to short	49
\GlsXtrSetRecordCountAttribute: new	155	\glxtr@setaccessdisplay: new ...	187
\glxtrtitlename: new	324	\glxtr@writefields: provide \glxtr@record in aux file	143
\glxtrtitleorpdforheading: new .	320	\glxtractivatenopost: new	127
\GlsXtrTotalRecordCount: new	154	\glxtrbookindexprelocation: removed check for no post dot	419
\glxtrwrglossmark: new	26	\glxtrglossentryother: new	146
short-em: new	278	\glxtrnopostpunc: new	127
short-sc: corrected first letter uppercasing	247	1.23 (2017-11-12)	
short-sm: corrected first letter uppercasing	261	\@@glxtrfmt: added check for indexing	33
shortcuts: ac	23	added grouping	33
\ifglxtr@hyperoutside: new	70	new	33
all: new	383	\@glxtr@nopostpunc@postdesc: new	128
nolong-short: new	238	\@glxtr@restore@postpunc: new ..	128
nolong-short-em: new	280	\@glxtrentryfmt: fixed missing label argument	34
nolong-short-noreg: new	238	\@glxtrfmt: new	33
nolong-short-sc: new	249	\eglupdatewidest: new	403
nolong-short-sm: new	263	\gglupdatewidest: new	402
nopostdot: new	19	\glupdatewidest: new	402
postpunc: new	19	\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand	21
\printunsrtglossaryentryprocesshook: new	148, 149	\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand	21
\printunsrtglossarypredoglossary: new	149	\glxtrfmtdisplay: new	33
\printunsrtglossaryskipentry: new	148	\glxtrifcustomdiscardperiod: new	198
\rGLS: new	158	\GlsXtrIfFieldUndef: new	36
\rGls: new	157	\glxtrrestorepostpunc: new	128
\rgls: new	157	\s@glxtrfmt: new	33
\rGLSformat: new	160	\s@glxtrfmt: new	33
\rGlsformat: new	159	\xglupdatewidest: new	403
\rglsformat: new	159	1.24 (2017-11-14)	
\rGLSpl: new	159	\glsadd: added \@gls@setsort	75
\rGlspl: new	158	\glxtrforcsvfield: new	35
\rglspl: new	157	\glxtrlocalsetgrouptitle: new ..	132
\rGLSplformat: new	160	1.25 (2017-11-14)	
\rGlsplformat: new	159	\glxtrbookindexmulticolsest: new	420
\rglsplformat: new	159	1.25 (2017-11-24)	
\s@glxtrifhasfield: switched from \ifdef to \ifundef	36	\glsextrapostnamehook: new	187
1.22 (2017-11-08)		\glxtrfootnotename: new	230
\@glxtr@nopostpunc: new	128	\glxtrlongnoshortdescname: new .	239
\@glxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glxtractivatenopost	127	\glxtrlongnoshortname: new	241
\@glxtrglossentryother: new	146	\glxtrlongshortname: new	225
		\glxtrlongshortuserdescname: new	296

\glxstronlyname:new	316	\glxstrGeneralLatinVIIrules:new	362
\glxstrpostlinkAddDescOnFirstUse:		\glxstrGeneralLatinVIrules:new .	361
changed to use \glxstrparen	198	\glxstrGeneralLatinVrules:new ..	361
\glxstrpostlinkAddSymbolOnFirstUse:		\glxstrgeneralpuncIIrules:new ..	357
changed to use \glxstrparen	199	\glxstrgeneralpuncIrules:new ...	356
\glxstrshortlongname:new	228	\glxstrgeneralpuncrules:new	356
\glxstrshortlonguserdescname:new	298	\glxstrhyphenrules:new	356
\glxstrshortnolongname:new	234	\glxstrLatinA:new	363
1.26 (2018-01-05)		\glxstrLatinAA:new	366
\@glxstr@do@inc@linkcount:new ..	160	\glxstrLatinAELigature:new	365
\glslinkpresetkeys:new	71	\glxstrLatinE:new	363
\glxstr@inc@linkcount:new	71	\glxstrLatinEszettSs:new	365
\GlsXtrEnableLinkCounting:new ..	161	\glxstrLatinEszettSz:new	365
\GlsXtrIfLinkCounterDef:new	161	\glxstrLatinEth:new	365
\glxstrinclinkcounter:new	161	\glxstrLatinH:new	364
\GlsXtrLinkCounterName:new	161	\glxstrLatinI:new	364
\GlsXtrLinkCounterValue:new	161	\glxstrLatinInsularG:new	366
\GlsXtrTheLinkCounter:new	161	\glxstrLatinK:new	364
1.27 (2018-02-26)		\glxstrLatinL:new	364
\@glxstrdialecthook:new	30	\glxstrLatinLslash:new	366
General: added		\glxstrLatinM:new	364
glossaries-extra-bib2gls.sty	338	\glxstrLatinN:new	364
\Alpha:new	350	\glxstrLatinO:new	364
\Beta:new	350	\glxstrLatinOELigature:new	365
\Chi:new	351	\glxstrLatinOslash:new	366
\Digamma:new	351	\glxstrLatinP:new	364
\Epsilon:new	350	\glxstrLatinS:new	364
\Eta:new	350	\glxstrLatinSchwa:new	365
\glxstr@loaddialect:new	338	\glxstrLatinT:new	365
\glxstrBasicDigitrules:new	380	\glxstrLatinThorn:new	365
\glxstrcombiningdiacriticIIrules:		\glxstrLatinWynn:new	366
new	354	\glxstrLatinX:new	365
\glxstrcombiningdiacriticIrules:		\glxstrMathGreekIIrules:new	372
new	354	\glxstrMathGreekIrules:new	371
\glxstrcombiningdiacriticIrules:		\glxstrMathItalicAlpha:new	377
new	354	\glxstrMathItalicBeta:new	377
\glxstrcombiningdiacriticIVrules:		\glxstrMathItalicChi:new	379
new	355	\glxstrMathItalicDelta:new	377
\glxstrcombiningdiacriticrules:		\glxstrMathItalicEpsilon:new ...	377
new	353	\glxstrMathItalicEta:new	377
\glxstrcontrolrules:new	352	\glxstrMathItalicGamma:new	377
\glxstrcurrencyrules:new	357	\glxstrMathItalicGreekIIrules:	
\glxstrdigitrules:new	380	new	368
\glxstrfractionrules:new	381	\glxstrMathItalicGreekIrules:new	367
\glxstrGeneralLatinIIIrules:new	359	\glxstrMathItalicIota:new	378
\glxstrGeneralLatinIIrules:new .	358	\glxstrMathItalicKappa:new	378
\glxstrGeneralLatinIrules:new ..	358	\glxstrMathItalicLambda:new	378
\glxstrGeneralLatinIVrules:new .	360	\glxstrMathItalicLowerGreekIIrules:	
\glxstrGeneralLatinVIIrules:new	363	new	370

<code>\glxtrMathItalicLowerGreekIrules:</code>	<code>\glxtrUpTau:</code>	new	376
new	<code>\glxtrUpTheta:</code>	new	374
<code>\glxtrMathItalicMu:</code>	<code>\glxtrUpUpsilon:</code>	new	376
new	<code>\glxtrUpXi:</code>	new	375
<code>\glxtrMathItalicNabla:</code>	<code>\glxtrUpZeta:</code>	new	374
new	<code>\Iota:</code>	new	350
<code>\glxtrMathItalicNu:</code>	<code>\Kappa:</code>	new	350
new	<code>\Mu:</code>	new	350
<code>\glxtrMathItalicOmega:</code>	<code>\Nu:</code>	new	350
new	<code>\Omicron:</code>	new	350
<code>\glxtrMathItalicOmicron:</code>	<code>\omicron:</code>	new	351
new	<code>\Rho:</code>	new	350
<code>\glxtrMathItalicPartial:</code>	<code>\Tau:</code>	new	351
new	<code>\Upalpha:</code>	new	351
<code>\glxtrMathItalicPhi:</code>	<code>\Upbeta:</code>	new	351
new	<code>\Upchi:</code>	new	352
<code>\glxtrMathItalicPi:</code>	<code>\Upsilon:</code>	new	351
new	<code>\Upeta:</code>	new	351
<code>\glxtrMathItalicPsi:</code>	<code>\Upiota:</code>	new	351
new	<code>\Upkappa:</code>	new	351
<code>\glxtrMathItalicRho:</code>	<code>\Upmu:</code>	new	351
new	<code>\Upnu:</code>	new	351
<code>\glxtrMathItalicSigma:</code>	<code>\Upomicron:</code>	new	351
new	<code>\upomicron:</code>	new	352
<code>\glxtrMathItalicTau:</code>	<code>\Uprho:</code>	new	352
new	<code>\Uptau:</code>	new	352
<code>\glxtrMathItalicTheta:</code>	<code>\Upzeta:</code>	new	351
new	<code>\Zeta:</code>	new	350
<code>\glxtrMathItalicUpperGreekIrules:</code>			
new			
<code>\glxtrMathItalicUpperGreekIrules:</code>			
new			
<code>\glxtrMathItalicUpsilon:</code>			
new			
<code>\glxtrMathItalicXi:</code>			
new			
<code>\glxtrMathItalicZeta:</code>			
new			
<code>\glxtrMathUpGreekIrules:</code>			
new			
<code>\glxtrMathUpGreekIrules:</code>			
new			
<code>\glxtrnonprintablerules:</code>			
new			
<code>\glxtrprovidecommand:</code>			
new			
<code>\glxtrspacerules:</code>			
new			
<code>\glxtrSubScriptDigitrules:</code>			
new			
<code>\glxtrSuperScriptDigitrules:</code>			
new			
<code>\glxtrUpAlpha:</code>			
new			
<code>\glxtrUpBeta:</code>			
new			
<code>\glxtrUpChi:</code>			
new			
<code>\glxtrUpDelta:</code>			
new			
<code>\glxtrUpDigamma:</code>			
new			
<code>\glxtrUpEpsilon:</code>			
new			
<code>\glxtrUpEta:</code>			
new			
<code>\glxtrUpGamma:</code>			
new			
<code>\glxtrUpIota:</code>			
new			
<code>\glxtrUpKappa:</code>			
new			
<code>\glxtrUpLambda:</code>			
new			
<code>\glxtrUpMu:</code>			
new			
<code>\glxtrUpNu:</code>			
new			
<code>\glxtrUpOmega:</code>			
new			
<code>\glxtrUpOmicron:</code>			
new			
<code>\glxtrUpPhi:</code>			
new			
<code>\glxtrUpPi:</code>			
new			
<code>\glxtrUpPsi:</code>			
new			
<code>\glxtrUpRho:</code>			
new			
<code>\glxtrUpSigma:</code>			
new			

1.28 (2018-03-06)	
<code>\@glxtr@docdefval:</code>	changed from count register to macro
<code>\@glxtr@dialecthook:</code>	save and restore <code>\TrackLangRequireDialectPrefix</code>
<code>\glxtr@redeffield:</code>	changed <code>\csedef</code> to <code>\protected@csedef</code>
<code>\glxtr@localsetgrouptitle:</code>	changed <code>\csedef</code> <code>\protected@csedef</code>
<code>\glxtr@rsetgrouptitle:</code>	changed <code>\csxdef</code> <code>\protected@csxdef</code>
1.29 (2018-04-09)	
<code>\@glxtr@removespaces:</code>	added expansion
<code>\@glxtr@dorecord:</code>	don't suppress expansion of <code>\@glxtr@recordloc</code> if counter isn't page
<code>\@glxtr@wrglossary@locationhyperlink:</code>	new
<code>\glxtr@inc@wrglossaryctr:</code>	new ...
<code>\glxtr@wrglossarylocation:</code>	new

\GlsXtrBibTeXEntryAliases: new .. 340	\GLSxtrshort: added
\glxtrfieldforlistloop: corrected	\@glxtr@record 213
argument order in \forlistcsloop 35	\Glsxtrshort: added
\GlsXtrIndexCounterLink: new 339	\@glxtr@record 213
\GlsXtrInternalLocationHyperlink:	\glxtrshort: added
new 25	\@glxtr@record 212
\GlsXtrProvideBibTeXFields: new . 341	\GLSxtrshortpl: added
indexcounter: new 25	\@glxtr@record 217
\setentrycounter: new 135	\Glsxtrshortpl: added
1.30 (2018-04-25)	\@glxtr@record 216
\@@glxtr@record: added check for	\glxtrshortpl: added
post-key hook 9	\@glxtr@record 216
added check for pre-key hook 9	\GlsXtrStartUnsetBuffering: new . 105
\@GLSxtr@fullpl: added	\GlsXtrStopUnsetBuffering: new .. 106
\@glxtr@record 211	indexcounter: added check for
\@GlsXtrStopUnsetBuffering: new . 106	wrglossary counter 26
\@Glsxtr@fullpl: added	\s@GlsXtrStopUnsetBuffering: new 106
\@glxtr@record 211	1.31 (2018-05-09)
\@glxtr@dorecord: don't suppress	\@GlsXtrStartUnsetBuffering: new 105
expansion of \@glsrecordlocref .. 11	\@gls@ifaccessattribute@set: new 170
\@glxtr@full: added	\@gls@initaccesskeys: new ... 169, 175
\@glxtr@record 209	\@gls@setup@default@short@access:
\@glxtr@fullpl: added	new 170, 175
\@glxtr@record 210	\@glxtr@record@noglossarywarning:
\@glxtr@glossadd@postkeys: new .. 10	new 140
\@glxtr@glossadd@prekeys: new ... 10	\@glxtrbuffer@nodup@unset: new . 105
\@glxtr@glslink@postkeys: new ... 10	General: added prefix key for glslink . 70
\@glxtr@glslink@prekeys: new 10	added prefix key for printgloss .. 129
\@glxtr@local@textformat: new ... 70	changed \let to \def 129
\@glxtr@unset: new 105	\glsaddeach: new 75
\@glxtrbuffer@unset: new 105	\glsapturedgroup: new 339
\glsadd: added \glsaddpostsetkeys . 75	\glsdefpostdesc: new 197
added \glsaddpresetkeys 74	\glsdefpostlink: new 198
\glsaddpostsetkeys: new 74	\glsdefpostname: new 187
\glsaddpresetkeys: new 74	\glsdohypertarget: bug fix: ensure that
\glsuserdescription: new 293	new version is picked up 129
\glxtrabbreviationfont: new 67	\glslistdesc: new 385
\GlsXtrDualBackLink: new 340	\glslocalreseteach: new 107
\GlsXtrDualField: new 340	\glslocalunseteach: new 107
\GlsXtrExpandedFmt: new 71	\glstreechilddesc: new 398
\GLSxtrlong: added \@glxtr@record 215	\glstreechildsymbol: new 398
\Glsxtrlong: added \@glxtr@record 214	\glstreedefaultnamefmt: new 395
\glxtrlong: added \@glxtr@record 214	\glstreedesc: new 397
\GLSxtrlongpl: added	\glstreegroupheaderfmt: added
\@glxtr@record 219	redefinition 396
\Glsxtrlongpl: added	\glstreenamefmt: added redefinition . 395
\@glxtr@record 218	\glstreenavigationfmt: added
\glxtrlongpl: added	redefinition 396
\@glxtr@record 218	\glstreenonamechilddesc: new 399

\glstreenonamedesc: new	399	\@dGls: new	349
\glstreenonamesymbol: new	399	\@dGlspl: new	349
\glstreesymbol: new	398	\@dglS: new	348
\glsxtr@newabbreviation: added		\@dglSpl: new	349
\ExtraCustomAbbreviationFields		\@glS@getcounterprefix: new	29
.....	204	\@glSlongextrawidestname: new ...	427
\GlsXtrForUnsetBufferedList: new	106	\@glSxtr@bibglS@removespaces: new	343
\GlsXtrIfFieldCmpNum: new	36	\@glSxtr@check@bibglS@nameref:	
\GlsXtrIfFieldEqNum: new	36	new	141
\GlsXtrIfFieldEqXpStr: new	39	\@glSxtr@do@nameref@record: new ..	12
\GlsXtrIfFieldNonZero: new	36	\@glSxtr@get@prefixedlabel: new ..	348
\GlsXtrIfHasNonZeroChildCount:		\@glSxtr@if@record@only: new	14
new	339	\@glSxtr@ifnum@mmode: new	12
\GlsXtrIfXpFieldEqXpStr: new	39	\@glSxtr@labelprefixes: new	347
\glSxtrpostlinkAddSymbolDescOnFirstUse:		\@glSxtr@prefixlabellist: new ...	348
new	199	\@glSxtr@providenewglS: new	152
\GlsXtrRecordWarning: new	139	\@glSxtr@record@only@setup: new ..	15
\glSxtrRevertToCmarks: new	319	\@glSxtr@record@setting@nameref:	
\GlsXtrStandaloneGlossaryType:		new	14
new	145	\@glSxtr@use@equation@counter@or:	
\GlsXtrStandaloneSubEntryItem:		new	71
new	146	General: new	425
\s@GlsXtrStartUnsetBuffering: new	105	page: nameref	11
1.32 (2018-05-24)		\dGLS: new	349
\GlsXtrForeignText: new	39	\dglS: new	348
\GlsXtrForeignTextField: new	41	\dglSdisp: new	350
\GlsXtrUnknownDialectWarning: new	41	\dglSlink: new	349
1.33 (2018-07-26)		\dGLSpl: new	349
\ifglSused: added redefinition	43	\glSadd: added grouping	74
1.34 (2018-07-29)		ensure that \glSadd performs indexing	75
\glS@begindocdefs: atom	57	\glSlongextraDescAlign: new	426
\GlsXtrIfUnusedOrUndefined: new ..	30	\glSlongextraDescFmt: new	425
\glSxtrNoGlossaryWarning: added		\glSlongextraDescNameHeader: new	432
package warning	23	\glSlongextraDescNameTabularFooter:	
\if@glSxtrdocdefrestricted:		new	432
changed to allow for atom as well ...	17	\glSlongextraDescNameTabularHeader:	
docdef: atom	17	new	432
1.35 (2018-08-13)		\glSlongextraDescSymNameHeader:	
\@glS@@link@: initialise post-link hook		new	444
commands	69	\glSlongextraDescSymNameTabularFooter:	
1.36 (2018-08-18)		new	444
\glSxtrautoindexesc: new	190	\glSlongextraDescSymNameTabularHeader:	
\glSxtrdisplaysupplc: new	341	new	444
\glSxtrmultisupplcation: new ...	341	\glSlongextraGroupHeading: new ..	427
1.37 (2018-11-30)		\glSlongextraHeaderFormat: new ..	427
\@glSxtr@record: added check for		\glSlongextraLocationAlign: new ..	427
auto-add	9	\glSlongextraLocationDescNameHeader:	
\@dGLS: new	349	new	433
\@dGLSpl: new	349		

\glslongextraLocationDescNameTabularFooter:	\glslongextraNameSymDescLocationTabularHeader:
new 433	new 439
\glslongextraLocationDescNameTabularHeader:	\glslongextraNameSymDescTabularFooter:
new 433	new 438
\glslongextraLocationDescSymNameHeader:	\glslongextraNameSymDescTabularHeader:
new 445	new 438
\glslongextraLocationDescSymNameTabularFooter:	\glslongextraSetDescWidth: new .. 428
new 446	\glslongextraSetWidest: new 427
\glslongextraLocationDescSymNameTabularHeader:	\glslongextraSubDescFmt: new 426
new 445	\glslongextraSubLocationFmt: new 426
\glslongextraLocationFmt: new ... 425	\glslongextraSubNameFmt: new 426
\glslongextraLocationSymDescNameHeader:	\glslongextraSubSymbolFmt: new .. 426
new 442	\glslongextraSymbolAlign: new ... 426
\glslongextraLocationSymDescNameTabularFooter:	\glslongextraSymbolFmt: new 425
new 442	\glslongextraSymDescNameHeader:
\glslongextraLocationSymDescNameTabularHeader:	new 441
new 442	\glslongextraSymDescNameTabularFooter:
\glslongextraLocSetDescWidth: new 428	new 441
\glslongextraNameAlign: new 426	\glslongextraSymDescNameTabularHeader:
\glslongextraNameDescHeader: new 427	new 441
\glslongextraNameDescLocationHeader:	\glslongextraSymLocSetDescWidth:
new 430	new 429
\glslongextraNameDescLocationTabularFooter:	\glslongextraSymSetDescWidth: new 428
new 430	\glslongextraTabularVAlign: new . 429
\glslongextraNameDescLocationTabularHeader:	\glslongextraUpdateWidest: new .. 427
new 430	\glslongextraUpdateWidestChild:
\glslongextraNameDescSymHeader:	new 428
new 434	\glsrenewcommand: new 339
\glslongextraNameDescSymLocationHeader:	\glsseeitemformat: removed reference
new 436	to \glslabel 49
\glslongextraNameDescSymLocationTabularFooter:	\glsxtr@dblfloat: new 18
new 436	\glsxtr@do@autoadd: new 71
\glslongextraNameDescSymLocationTabularHeader:	\glsxtr@float: new 18
new 436	\glsxtr@record@nameref: new 144
\glslongextraNameDescSymTabularFooter:	\glsxtr@renewcommand: new 339
new 435	\glsxtr@writefields: provide
\glslongextraNameDescSymTabularHeader:	\glsxtr@record@nameref in aux
new 434	file 143
\glslongextraNameDescTabularFooter:	\glsxtraddlabelprefix: new 347
new 427	\GlsXtrAutoAddOnFormat: new 71
\glslongextraNameDescTabularHeader:	\glsxtrclearlabelprefixes: new .. 347
new 427	\glsxtrdisplaylocnameref: new ... 341
\glslongextraNameFmt: new 425	\glsxtrfmtexternalnameref: new .. 344
\glslongextraNameSymDescHeader:	\glsxtrfmtinternalnameref: new .. 344
new 437	\GLSXTRhiername: new 51
\glslongextraNameSymDescLocationHeader:	\GLSXtrhiername: new 51
new 439	\GlsXtrhiername: new 50
\glslongextraNameSymDescLocationTabularFooter:	\GLSXtrhiername: new 50
new 439	\glsxtrhiername: new 50

<code>\glxtrhiernamesep: new</code>	51	<code>floats: new</code>	18
<code>\glxtridentifyglsl-like: new</code>	152	<code>long-desc-name: new</code>	432
<code>\glxtrifinlabelprefixlist: new</code> ..	348	<code>long-desc-sym-name: new</code>	444
<code>\GlsXtrLocationField: new</code>	151	<code>long-loc-desc-name: new</code>	433
<code>\glxtrnameloclink: new</code>	343	<code>long-loc-desc-sym-name: new</code>	446
<code>\glxtrnamerefink: new</code>	342	<code>long-loc-sym-desc-name: new</code>	443
<code>\glxtrprependlabelprefix: new</code> ..	347	<code>long-name-desc: new</code>	429
<code>\GlsXtrSetAltModifier: write modifier</code>		<code>long-name-desc-loc: new</code>	430
to aux	93	<code>long-name-desc-sym: new</code>	435
<code>\glxtrSetWidest: new</code>	344	<code>long-name-desc-sym-loc: new</code>	436
<code>\glxtrSetWidestFallback: new</code> ...	346	<code>long-name-sym-desc: new</code>	438
<code>\GlsXtrStandaloneEntryName: new</code> ..	145	<code>long-name-sym-desc-loc: new</code>	439
<code>\GlsXtrStandaloneEntryOther: new</code>	147	<code>long-sym-desc-name: new</code>	441
<code>\GLSxtrusefield: new</code>	37	<code>equations: new</code>	18
<code>\Glsxtrusefield: fixed internal</code>			
command and added check for		1.38 (2018-12-01)	
<code>\texorpdfstring</code>	37	<code>\glslongextraNameFmt: bug fix:</code>	
<code>\ifGlsLongExtraUseTabular: new</code> ..	429	removed double param	425

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\\$	339
\,	51
\.	18, 19, 198, 395
\@	57, 141
\@cGLS@	109, 117
\@cGLSpl@	109, 117
\@cGls@	109, 117
\@cGlspl@	109, 117
\@cgls@	109, 117
\@cglspl@	109, 117
\@do@wrglossary	8, 10, 124
\@do@wrglossary	13, 15, 16, 29, 75, 91
\@glo@assign@sortkey	129
\@glo@list	6
\@glo@type	147
\@glossarysec	420
\@gls@expand@field	32
\@glslocalreset	107
\@glslocalunset	106
\@glsreset	106
\@glsunset	105
\@glsxtr@autoindex@escspch	192, 193
\@glsxtr@checkspch	191–194
\@glsxtr@disabledflycommand	62
\@glsxtr@org@postdescription	128
\@glsxtr@record	15, 16
\@glsxtr@recordcounter	15, 16, 144
\@glsxtrfmt	33
\@glsxtrp	100, 101
\@glsxtrpostloctag	65
\@glsxtrpreloctag	65
\@glsxtrwrglossmark	8, 9, 13, 28, 29, 53, 58, 124
\@newglossaryentry@defcounters	108
\@newglossaryentry@defunitcounters	116
\@par	401
\@ACRlong	96
\@ACRlongpl	96
\@ACRshort	96
\@ACRshortpl	96
\@Acrlong	96
\@Acrlongpl	96
\@Acrshort	96
\@Acrshortpl	96
\@GLS@	96, 112, 159, 349
\@GLSdesc@	80
\@GLSpl@	96, 112, 113, 159, 349
\@GLSplural@	97
\@GLSSymbol@	81
\@GLStext@	97
\@GLSxtr@full	210
\@GLSxtr@fullpl	211
\@GLSxtr@p@acrlong@	96
\@GLSxtr@p@acrlongpl@	96
\@GLSxtr@p@acrshort@	96
\@GLSxtr@p@acrshortpl@	96
\@GLSxtr@p@long@	96
\@GLSxtr@p@longpl@	96
\@GLSxtr@p@plural@	96
\@GLSxtr@p@short@	96
\@GLSxtr@p@shortpl@	96
\@GLSxtr@p@text@	96
\@GLSxtrlong	96, 215
\@GLSxtrlongpl	96, 219
\@GLSxtrp	104
\@GLSxtrshort	96, 213
\@GLSxtrshortpl	96, 217
\@Gls@	96, 111, 112, 158, 349
\@Gls@acrentryname	120
\@Gls@entry@field	37, 87, 102, 103, 188
\@Gls@entryname	120
\@GlsXtrEnableOnTheFly	59
\@GlsXtrStartUnsetBuffering	105

\@GlsXtrStopUnsetBuffering	106	\@dglspl	348
\@Glspl@	96, 111, 112, 158, 349	\@disable@onlypremakeg	123
\@Glsplural@	97	\@do@auxoutstuff	136, 137
\@Glstext@	97	\@do@glsg@getcounterprefix	11
\@Glsxtr	60, 62	\@do@glsssee	54, 55
\@Glsxtr@full	209	\@do@newglossaryentry	120, 206
\@Glsxtr@fullpl	211	\@do@seeglossary	15, 16, 27, 58, 124
\@Glsxtr@p@acrlong@	96	\@do@wrglossary	73, 74, 156
\@Glsxtr@p@acrlongpl@	96	\@empty	
\@Glsxtr@p@acrshort@	96		30, 75, 83–87, 128, 135, 191, 192, 209–219
\@Glsxtr@p@acrshortpl@	96	\@end@glsxtr@addunused	56
\@Glsxtr@p@long@	96	\@end@glsxtr@gettype	126, 129
\@Glsxtr@p@longpl@	96	\@end@glsxtr@usesee	49
\@Glsxtr@p@plural@	96	\@end@glsxtr@trifhyphenstart	301, 302
\@Glsxtr@p@short@	96	\@endfortrue	35, 187, 222, 348
\@Glsxtr@p@shortpl@	96	\@firstofone	76, 148, 182, 189, 195, 342, 343
\@Glsxtr@p@text@	96	\@firstofthree	69, 75,
\@Glsxtrlong	96, 214		83–86, 92, 93, 209, 210, 212, 214, 216, 218
\@Glsxtrlongpl	96, 218	\@firstoftwo	76–
\@Glsxtrp	103		78, 80, 81, 84–87, 89, 92, 93, 122, 187,
\@Glsxtrpl	61, 62		188, 200, 201, 209–211, 216–219, 320, 321
\@Glsxtrshort	96, 212	\@float	18
\@Glsxtrshortpl	96, 216	\@for	6, 24, 35, 56, 75, 107, 108, 119, 123,
\@acrlong	96		126, 133, 148, 155, 161, 180, 187, 195, 348
\@acrlongpl	96	\@glo@alias	53, 54
\@acrshort	96	\@glo@assign@sortkey	126
\@acrshortpl	96	\@glo@autosee	28
\@addtoreset	160	\@glo@autoseehook	54
\@afterheading		\@glo@category	114
	386, 387, 397, 399–401, 413–416, 423	\@glo@check@sortallowed	126
\@alt@glshyp@opt	93	\@glo@counterprefix	10, 11, 29, 30, 135, 343
\@auxout	11–	\@glo@countunit	114
	13, 58, 66, 93, 109, 110, 118, 123, 124,	\@glo@default@sorttype	126
	136, 137, 141, 143, 145, 152, 153, 348, 424	\@glo@desc	43, 44
\@bibgl@restoreat	141	\@glo@descplural	43, 44
\@cGLS	112	\@glo@group	14
\@cGLS@	109, 112, 117	\@glo@label	
\@cGLSpl	113		13, 14, 32, 49, 53–55, 87, 95, 404–410
\@cGLSpl@	109, 113, 117	\@glo@location	14
\@cGls@	109, 117	\@glo@loclist	13
\@cGlspl@	109, 117	\@glo@name	190
\@cgl@	109, 117	\@glo@no@assign@sortkey	129
\@cglspl@	109, 117	\@glo@parent	405, 406
\@dGLS	349	\@glo@see	49, 52, 54–56
\@dGLSpl	349	\@glo@seealso	53, 54
\@dGls	349	\@glo@sort	190
\@dGlspl	349	\@glo@sorttype	126, 133
\@dblfloat	18	\@glo@text	69
\@dgl@	348	\@glo@thislettergrp	151

<code>\@glo@thisvalue</code>	292	<code>\@gls@label</code>	8, 9, 11, 58, 92, 124, 145, 222
<code>\@glo@tmp</code>	30–32, 52, 87	<code>\@gls@levelchar</code>	191
<code>\@glo@type</code>	55, 93, 120, 123, 126, 127, 129, 132, 133, 136, 137, 139, 140, 147, 148	<code>\@gls@link</code>	33, 68, 69, 83–87, 209–219
<code>\@glo@types</code>	178, 179, 403–410	<code>\@gls@link@checkfirsthyper</code>	69, 122
<code>\@glossary@default@style</code> .	62, 63, 126, 418	<code>\@gls@link@label</code>	72, 156
<code>\@glossarystyle</code>	126, 127	<code>\@gls@link@nocheckfirsthyper</code>	67, 83–87, 209–219
<code>\@gls@</code>	96, 111, 112, 157, 348	<code>\@gls@link@opts</code>	72
<code>\@gls@@link</code>	69	<code>\@gls@list</code>	133
<code>\@gls@ReturnAfterFi</code>	135, 343	<code>\@gls@local@increment@currcount</code> ...	109
<code>\@gls@actualchar</code>	191	<code>\@gls@local@increment@currunitcount</code>	117
<code>\@gls@adjustmode</code>	74	<code>\@gls@location</code>	151, 152
<code>\@gls@alt@hyp@opt</code>	93	<code>\@gls@loclist</code>	130, 131, 151, 152
<code>\@gls@alt@hyp@opt@char</code>	93	<code>\@gls@long</code>	204
<code>\@gls@alt@hyp@opt@keys</code>	93	<code>\@gls@longpl</code>	202, 204, 205
<code>\@gls@automake</code>	126	<code>\@gls@map</code>	187
<code>\@gls@between</code>	132, 133	<code>\@gls@nameaccess</code>	169, 171
<code>\@gls@checkedmkidx</code>	191–194	<code>\@gls@nohyperlist</code>	44–46
<code>\@gls@checkedmkidxchars</code>	53, 190	<code>\@gls@noidx@do</code>	133
<code>\@gls@codepage</code>	137	<code>\@gls@noidx@getgrouptitle</code>	148
<code>\@gls@counter</code> 9, 11, 26, 30, 71, 72, 74, 91, 156		<code>\@gls@noidx@nosanitizesort</code>	125
<code>\@gls@currentlettergroup</code> ...	133, 148, 151	<code>\@gls@noidx@sanitizesort</code>	125
<code>\@gls@declareoption</code>	5	<code>\@gls@noidx@loclist@finalsep</code>	130
<code>\@gls@default@longpl</code>	204, 205	<code>\@gls@noidx@loclist@prev</code>	130
<code>\@gls@deffile</code>	57	<code>\@gls@noidx@loclist@sep</code>	130
<code>\@gls@doautomake</code>	126, 144	<code>\@gls@noref@warn</code>	124, 133
<code>\@gls@doautomake@err</code>	144	<code>\@gls@org@glsnoidx@displayloc</code> ..	130, 131
<code>\@gls@enablesavenonumberlist</code>	57	<code>\@gls@org@glssee@format</code>	130, 131
<code>\@gls@encapchar</code>	191	<code>\@gls@preglossaryhook</code>	127, 195
<code>\@gls@entry@count</code>	109, 110	<code>\@gls@prevlevel</code>	411–413, 417, 418
<code>\@gls@entry@field</code>	32, 37, 54, 87, 101, 103, 104, 108, 146, 147	<code>\@gls@quotechar</code>	191
<code>\@gls@entry@unitcount</code>	117, 118	<code>\@gls@reference</code>	58, 123, 124
<code>\@gls@field@font</code>	75–83	<code>\@gls@restoreat</code>	57
<code>\@gls@field@link</code>	76–83, 88, 89	<code>\@gls@saveentrycounter</code> 15, 16, 29, 73, 75, 156	
<code>\@gls@firstaccess</code>	169, 172	<code>\@gls@see@noindex</code>	28, 141
<code>\@gls@get@counterprefix</code>	30	<code>\@gls@setdefault@glslink@opts</code>	9, 33, 72, 91
<code>\@gls@getcounterprefix</code>	11	<code>\@gls@setsort</code>	73, 75
<code>\@gls@getgrouptitle</code>	132, 148	<code>\@gls@setup@default@short@access</code> ..	205
<code>\@gls@grptitle</code>	93, 133	<code>\@gls@setupsort@none</code>	15
<code>\@gls@hyp@opt</code>	88, 89, 93, 112, 113, 153, 157–159, 208–219, 348, 349	<code>\@gls@short</code>	204, 205
<code>\@gls@hyp@opt@cs</code>	92, 93	<code>\@gls@shortaccess</code>	169–172
<code>\@gls@ifaccessattribute@set</code>	170	<code>\@gls@shortaccesspl</code>	169–171
<code>\@gls@ifinlist</code>	149	<code>\@gls@shortpl</code>	202, 205
<code>\@gls@increment@currcount</code>	109	<code>\@gls@sort</code>	151
<code>\@gls@increment@currunitcount</code>	117	<code>\@gls@textaccess</code>	169, 171
<code>\@gls@initaccesskeys</code>	204	<code>\@gls@thisHloc</code>	29
<code>\@gls@keymap</code> ..	13, 14, 31, 49, 53, 87, 143, 187	<code>\@gls@thislabel</code>	75, 107, 348–350
		<code>\@gls@thisloc</code>	29

<code>\@gls@thisval</code>	187	<code>\@glsstyle@mcolttreegroup</code>	414
<code>\@gls@tmp</code>	39, 133	<code>\@glsstyle@mcolttreehypergroup</code>	415
<code>\@gls@tmpb</code>	193, 194	<code>\@glsstyle@mcolttreenonamegroup</code>	415
<code>\@gls@type</code>	124, 126, 222, 404–410	<code>\@glsstyle@mcolttreenonamehypergroup</code>	416
<code>\@gls@write@entrycounts</code>	109	<code>\@glsstyle@mcolttreenonamespannav</code> ..	416
<code>\@gls@write@entryunitcounts</code>	117	<code>\@glsstyle@mcolttreespannav</code>	415
<code>\@gls@write@entryunitcounts@do</code>	118	<code>\@glsstyle@tree</code>	397
<code>\@gls@writedef</code>	57	<code>\@glsstyle@treegroup</code>	398
<code>\@gls@xref</code>	13, 53	<code>\@glsstyle@treehypergroup</code>	399
<code>\@glsabbrv@current@abbreviation</code>	204, 219	<code>\@glsstyle@treenoname</code>	399
<code>\@glsacronymlists</code>	120	<code>\@glsstyle@treenonamegroup</code>	400
<code>\@glsdoifexistsorwarn</code> ...	17, 183–186, 188	<code>\@glsstyle@treenonamehypergroup</code> ...	400
<code>\@glsentry</code>	57, 110, 118	<code>\@glstarget</code>	95, 129
<code>\@glslink</code>	73, 94, 95	<code>\@glstext@</code>	97
<code>\@glslocalreset</code>	107	<code>\@glsunset</code>	105, 106
<code>\@glslocalunset</code>	106, 107	<code>\@glswidestname</code>	403, 411
<code>\@glslongextra@begintab</code>	429–446	<code>\@glsxtr</code>	60, 62
<code>\@glslongextrawidestname</code>	427, 428	<code>\@glsxtr@do@wrglossary</code>	124
<code>\@glsnextpages</code>	127	<code>\@glsxtr@abbreviationsdef</code>	20, 28, 29
<code>\@glsnonextpages</code>	127	<code>\@glsxtr@accessdisplay</code>	187–189
<code>\@glsnumberformat</code>	<code>\@glsxtr@activate@initialtagging</code>
...	9, 11, 71, 72, 74, 91, 156, 187, 190, 343	...	195, 196
<code>\@glsorder</code>	123	<code>\@glsxtr@addunitcounter</code>	114
<code>\@glspl@</code>	96, 111, 112, 157, 349	<code>\@glsxtr@addunused</code>	56
<code>\@glsplural@</code>	97	<code>\@glsxtr@addunusedxrefs</code>	55, 56
<code>\@gls punc@token</code>	200, 201	<code>\@glsxtr@altmodifier</code>	93
<code>\@glsrecordlocref</code>	10, 11	<code>\@glsxtr@attrval</code>	73, 181–186, 188, 190
<code>\@glsshowtarget</code>	95	<code>\@glsxtr@autoindex@at</code>	190–192
<code>\@glsstyle@altlist</code>	385	<code>\@glsxtr@autoindex@doextra@esc</code>	190
<code>\@glsstyle@altlistgroup</code>	386	<code>\@glsxtr@autoindex@encap</code>	190–192
<code>\@glsstyle@altlisthypergroup</code>	387	<code>\@glsxtr@autoindex@esc</code>	191, 193, 194
<code>\@glsstyle@alttree</code>	401	<code>\@glsxtr@autoindex@escat</code>	191, 192
<code>\@glsstyle@alttreegroup</code>	412	<code>\@glsxtr@autoindex@escencap</code> ...	191, 192
<code>\@glsstyle@alttreehypergroup</code>	413	<code>\@glsxtr@autoindex@esclevel</code> ...	191, 192
<code>\@glsstyle@index</code>	396	<code>\@glsxtr@autoindex@escquote</code> ...	191, 193
<code>\@glsstyle@indexgroup</code>	397	<code>\@glsxtr@autoindex@level</code>	191, 192
<code>\@glsstyle@indexhypergroup</code>	397	<code>\@glsxtr@autoindex@setname</code>	190
<code>\@glsstyle@inline</code>	395	<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 49, 54
<code>\@glsstyle@list</code>	385	<code>\@glsxtr@autoseeindexfalse</code>	15
<code>\@glsstyle@listdotted</code>	384	<code>\@glsxtr@autoseeindextrue</code>	17
<code>\@glsstyle@listgroup</code>	386	<code>\@glsxtr@bibgls@removespaces</code>	343
<code>\@glsstyle@listhypergroup</code>	386	<code>\@glsxtr@bookindex@atendgroup</code> ..	421–423
<code>\@glsstyle@mcolalttree</code>	416	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	422
<code>\@glsstyle@mcolalttreegroup</code>	417	<code>\@glsxtr@bookindex@atsubsubendgroup</code> ..	422
<code>\@glsstyle@mcolalttreehypergroup</code> ..	417	<code>\@glsxtr@bookindex@between</code>	421, 423
<code>\@glsstyle@mcolalttreespannav</code>	418	<code>\@glsxtr@bookindex@sep</code>	421, 422
<code>\@glsstyle@mcolindexgroup</code>	413	<code>\@glsxtr@bookindex@subatendgroup</code>
<code>\@glsstyle@mcolindexhypergroup</code>	413	...	421–423
<code>\@glsstyle@mcolindexspannav</code>	414	<code>\@glsxtr@bookindex@subbetween</code> ..	421, 422

\@glsxtr@bookindex@subsep	421, 422	\@glsxtr@dostylewarn	222
\@glsxtr@bookindex@subsubatendgroup	421–423	\@glsxtr@enabletagging	194
\@glsxtr@bookindex@subsubbetween ..	421, 422	\@glsxtr@end@	59
\@glsxtr@bookindexgroupskip ...	421, 423	\@glsxtr@endescspch	191–194
\@glsxtr@cat	108, 119, 155, 195	\@glsxtr@entrycount@org@localreset	109
\@glsxtr@check@bibgls@nameref	141	\@glsxtr@entrycount@org@localunset	109
\@glsxtr@checkgroup	148	\@glsxtr@entrycount@org@reset	109
\@glsxtr@counterrecordhook	11–13	\@glsxtr@entrycount@org@unset	109
\@glsxtr@csname	115, 117	\@glsxtr@entryunitcount@org@localreset	117
\@glsxtr@current@style	63, 418	\@glsxtr@entryunitcount@org@localunset	117
\@glsxtr@currentunitcount	115, 117	\@glsxtr@entryunitcount@org@reset .	117
\@glsxtr@currunitcount	116, 118	\@glsxtr@entryunitcount@org@unset .	117
\@glsxtr@debugnr	26	\@glsxtr@equationsfalse	18
\@glsxtr@debugval	26	\@glsxtr@err@undefaction	7, 16
\@glsxtr@declareoption ...	5, 18, 20, 23, 25	\@glsxtr@field@linkdefs	67, 69
\@glsxtr@defaultnoglossarywarning	23, 24	\@glsxtr@floatsfalse	18
\@glsxtr@defaultnumberformat	7, 9, 72, 74, 91, 187, 190	\@glsxtr@format@overridefalse	189
\@glsxtr@defpostpunc	18, 19, 27	\@glsxtr@format@overridetrue	189
\@glsxtr@deprecated@abbrstyle	251, 253, 254, 256, 265, 267, 268, 270, 283, 286, 290, 292	\@glsxtr@foundinlist	201
\@glsxtr@dialect	39–41	\@glsxtr@full	208
\@glsxtr@disabledflycommand	62	\@glsxtr@fullpl	210
\@glsxtr@display@loc	134	\@glsxtr@get@prefixedlabel	348–350
\@glsxtr@do@windex	92	\@glsxtr@gettype	126
\@glsxtr@do@autoadd	71	\@glsxtr@glossdescfont	181–183
\@glsxtr@do@glsdisablehyperinlist ..	90	\@glsxtr@glossnamefont .	183–186, 188, 189
\@glsxtr@do@inc@linkcount	161	\@glsxtr@gobbleto@endescspch	193
\@glsxtr@do@nameref@record	11	\@glsxtr@groupheading	148, 150, 151
\@glsxtr@do@record@wrglossary	8, 15	\@glsxtr@idx@displaynumberlist	125
\@glsxtr@do@redef@forglsentries	7	\@glsxtr@idx@entrynumberlist	125
\@glsxtr@do@style	25, 338	\@glsxtr@if@record@only	122
\@glsxtr@do@titlecaps@warn	182–185, 188, 195, 196	\@glsxtr@ifcsstart	59
\@glsxtr@doabbreviationsdef	20	\@glsxtr@ifnum@mmode	71
\@glsxtr@doaccsupp	23, 27	\@glsxtr@ifpunctoken	201
\@glsxtr@docdefsetting	17, 57	\@glsxtr@ifunitcounter	114
\@glsxtr@docdefval	16, 17, 57, 58	\@glsxtr@insert@dots	202
\@glsxtr@docounterrecord	13	\@glsxtr@insert@dots@next	203
\@glsxtr@doglossary	148	\@glsxtr@insertdots	170, 204
\@glsxtr@doiflabelinlist	149	\@glsxtr@label	35, 56, 161, 180, 181
\@glsxtr@doloctag	65	\@glsxtr@labelprefixes	347, 348
\@glsxtr@dorecord	8, 10	\@glsxtr@loadstyles	383, 384
\@glsxtr@dorecordnodefer	8, 10	\@glsxtr@local@textformat	72, 73
\@glsxtr@dosee@alsoindex@glossary ..	16	\@glsxtr@locale	40, 41
\@glsxtr@doseeglossary	15, 28	\@glsxtr@longnewglossaryentry	43
		\@glsxtr@mark@wordseps	203
		\@glsxtr@mark@wordseps@next	203
		\@glsxtr@markwordseps	204, 205
		\@glsxtr@mixed@assign@sortkey	126

\@glxstr@newglslake	152, 153	\@glxstr@org@glxstrtitlename ..	320, 321
\@glxstr@noidx@displaynumberlist ..	125	\@glxstr@org@glxstrtitleorpdforheading	320, 321
\@glxstr@noidx@do	150	\@glxstr@org@glxstrtitleplural ..	320, 321
\@glxstr@noidx@entrynumberlist	125	\@glxstr@org@glxstrtitleshort ..	320, 321
\@glxstr@noidx@numberlistloop	125	\@glxstr@org@glxstrtitleshortpl ..	320, 321
\@glxstr@nomissingglstextnr	23, 24	\@glxstr@org@glxstrtitletext ..	320, 321
\@glxstr@nomissingglstextval	23	\@glxstr@org@makeglossaries ...	122, 123
\@glxstr@noop@recordcounter	12, 16	\@glxstr@org@markboth	319
\@glxstr@nopostpunc	127	\@glxstr@org@markright	318, 319
\@glxstr@nopostpunc@postdesc	128	\@glxstr@org@newacronymstyle ..	121, 122
\@glxstr@notfoundinlist	201	\@glxstr@org@postdescription ..	128, 196
\@glxstr@op@recordcounter	15, 16	\@glxstr@org@see@noindex	141
\@glxstr@optlist	62	\@glxstr@org@setacronymstyle ..	121, 122
\@glxstr@org@GLS@	68	\@glxstr@org@theHvalue	8, 9
\@glxstr@org@GLSpl@	68	\@glxstr@org@unset@buffer	105, 106
\@glxstr@org@Gls@	68	\@glxstr@org@prefix	10, 11
\@glxstr@org@Glspl@	68	\@glxstr@org@printglossary	62, 128
\@glxstr@org@Glsxtrtitlefirst ..	320, 321	\@glxstr@org@warndep	202
\@glxstr@org@Glsxtrtitlefirstplural	320, 322	\@glxstr@p@acrlong@	96
\@glxstr@org@Glsxtrtitlefull ..	320, 322	\@glxstr@p@acrlongpl@	96
\@glxstr@org@Glsxtrtitlefullpl ..	320, 322	\@glxstr@p@acrshort@	96
\@glxstr@org@Glsxtrtitlelong ..	320, 322	\@glxstr@p@acrshortpl@	96
\@glxstr@org@Glsxtrtitlelongpl ..	320, 322	\@glxstr@p@long@	96
\@glxstr@org@Glsxtrtitlename ..	320, 321	\@glxstr@p@longpl@	96
\@glxstr@org@Glsxtrtitleplural ..	320, 321	\@glxstr@p@plural@	96
\@glxstr@org@Glsxtrtitleshort ..	320, 321	\@glxstr@p@short@	96
\@glxstr@org@Glsxtrtitleshortpl ..	320, 321	\@glxstr@p@shortpl@	96
\@glxstr@org@Glsxtrtitletext ..	320, 321	\@glxstr@p@text@	96
\@glxstr@org@MakeUppercase	320, 321	\@glxstr@p@pagetag	65
\@glxstr@org@checkfirsthyper ...	89, 122	\@glxstr@p@pagetag	65
\@glxstr@org@currentfieldvalue ..	39, 40	\@glxstr@prefix	348
\@glxstr@org@delimN	65, 66	\@glxstr@prevunitcount	116
\@glxstr@org@delimR	65, 66	\@glxstr@printglossnr	128
\@glxstr@org@doseeglossary	28, 124	\@glxstr@printglossopts	62, 126, 128
\@glxstr@org@gloautosee	28	\@glxstr@printglossval	128
\@glxstr@org@glolinkprefix	72, 74	\@glxstr@printunsrtglossaryskipentry	148, 149
\@glxstr@org@gls@	68	\@glxstr@provide@addstoragekey	32
\@glxstr@org@glsdohypertarget	129	\@glxstr@provide@storagekey	31
\@glxstr@org@glsignore	65, 66	\@glxstr@providenewgls	153
\@glxstr@org@glspl@	68	\@glxstr@record ..	15, 16, 67–69, 74, 209–219
\@glxstr@org@glsxtrtitlefirst ..	320, 321	\@glxstr@record@noglossarywarning ..	15
\@glxstr@org@glsxtrtitlefirstplural	320, 321	\@glxstr@record@only@setup	16
\@glxstr@org@glsxtrtitlefull ..	320, 322	\@glxstr@record@setting	8, 10, 11, 14, 15, 30, 52, 57, 58, 93, 123, 141, 143, 152
\@glxstr@org@glsxtrtitlefullpl ..	320, 322	\@glxstr@record@setting@alsoindex ..	8, 10, 52, 123
\@glxstr@org@glsxtrtitlelong ..	320, 322		
\@glxstr@org@glsxtrtitlelongpl ..	320, 322		

<code>\@glstr@record@setting@nameref</code> ...	<code>\@glstrifhyphenstart</code>	301
..... 11, 14, 30, 141, 143	<code>\@glstrindexaliased</code>	91
<code>\@glstr@record@setting@off</code> ..	<code>\@glstrindexcrossrefsfalse</code>	17
57, 93, 152	<code>\@glstrindexcrossrefstrue</code>	17
<code>\@glstr@record@setting@only</code>	<code>\@glstrinmark</code>	319
14, 30	<code>\@glstrlong</code>	96, 214
<code>\@glstr@recordsee</code>	<code>\@glstrlongpl</code>	96, 217
15, 28, 52	<code>\@glstrnewgls</code>	154
<code>\@glstr@redef@forglsentries</code>	<code>\@glstrnewgls@inner</code>	152, 153
7, 29	<code>\@glstrnewgls@innercsname</code>	153
<code>\@glstr@redefstyles</code>	<code>\@glstrnotinmark</code>	319
24, 338	<code>\@glstrp</code>	101, 102
<code>\@glstr@reg@glosslist</code>	<code>\@glstrp@opt</code>	99, 100
123–126, 129	<code>\@glstrpl</code>	61, 62
<code>\@glstr@restore@postpunc</code>	<code>\@glstrpostloctag</code>	64–66
128	<code>\@glstrpreloctag</code>	64–66
<code>\@glstr@rglstrtrigger@record</code> ...	<code>\@glstrsetaliasnoindex</code>	90, 91
157–159	<code>\@glstrshort</code>	96, 212
<code>\@glstr@s@longnewglossaryentry</code>	<code>\@glstrshortpl</code>	96, 216
43	<code>\@glstrtrundeftag</code>	6, 30
<code>\@glstr@s@savepreloctag</code>	<code>\@glstrwrglossmark</code>	26, 27
65, 66	<code>\@gobble</code> .. 7, 16, 18, 76, 149, 150, 203, 421–423	
<code>\@glstr@setentrycountunsetattr</code> ...	<code>\@gobbletwo</code>	202
108	<code>\@gtempa</code>	339
<code>\@glstr@setentryunitcountunsetattr</code> 119	<code>\@ifdefinable</code>	339
<code>\@glstr@setupshortcuts</code>	<code>\@ifnextchar</code>	92, 93
22, 23, 28, 29	<code>\@ifpackageloaded</code>	5, 20, 143,
<code>\@glstr@shortcutsnr</code>	162, 181, 183, 185, 187, 189, 338, 351, 382	
22	<code>\@ifstar</code>	31,
<code>\@glstr@shortcutsval</code>	32, 35, 43, 44, 46, 59, 92, 105, 106, 147, 194	
22, 143	<code>\@ifundefined</code>	338, 339
<code>\@glstr@swaptwo</code>	<code>\@ignored@glossaries</code>	44–46
201	<code>\@input</code>	141
<code>\@glstr@tag</code>	<code>\@input@</code>	136
195	<code>\@istfilename</code>	123
<code>\@glstr@taggingcs</code>	<code>\@makeglossary</code>	123
195	<code>\@mfu@domakefirstuc</code>	195, 196
<code>\@glstr@textformat</code>	<code>\@mfu@nocaplist</code>	196
73, 74	<code>\@one</code>	110, 119, 153
<code>\@glstr@theHvalue</code> ...	<code>\@newglossaryentry@defcounters</code> 108, 116	
8–10, 70, 72–75, 156	<code>\@newglossaryentryposthook</code>	13, 14, 31, 53, 87
<code>\@glstr@thevalue</code>	<code>\@newglossaryentryprehook</code>	13, 14, 31, 43, 53, 87
8–10, 70, 72–75, 156	<code>\@nil</code>	135, 151, 343
<code>\@glstr@thisloctag</code>	<code>\@nnil</code>	191, 193, 194, 201–203
65, 66	<code>\@no@glstrindexaliased</code>	91
<code>\@glstr@titelabel</code>	<code>\@no@makeglossaries</code>	140
131, 132, 150	<code>\@nocounterr</code>	161
<code>\@glstr@tmp</code>		
24, 71, 134		
<code>\@glstr@type</code>		
181		
<code>\@glstr@unitcountlist</code>		
114		
<code>\@glstr@unset</code>		
105, 106		
<code>\@glstr@unset@buffer</code>		
105, 106		
<code>\@glstr@unsrt@getgrouptitle</code>		
148		
<code>\@glstr@use@equation@counter</code> .		
9, 71, 72		
<code>\@glstr@usesee</code>		
49		
<code>\@glstr@warn@onexistsordo</code>		
7, 15, 16		
<code>\@glstr@warn@undefaction</code>		
7, 15, 16		
<code>\@glstr@wrglossary@locationhyperlink</code>		
26		
<code>\@glstr@wrglossnr</code>		
70		
<code>\@glstr@wrglossval</code>		
70		
<code>\@glstr@buffer@nodup@unset</code>		
105		
<code>\@glstr@buffer@unset</code>		
105		
<code>\@glstr@dialecthook</code>		
338		
<code>\@glstr@docdeffalse</code>		
58		
<code>\@glstr@entryfmt</code>		
34		
<code>\@glstr@fmt</code>		
32		
<code>\@glstr@glossentry</code>		
145		
<code>\@glstr@glossentryother</code>		
146		
<code>\@glstr@hypernameprefix</code>		
129, 150		
<code>\@glstr@trifhasfield</code>		
35		

<code>\@nopostdesc</code>	127
<code>\@onelevel@sanitize</code>	13, 53, 62, 132, 150
<code>\@onlypreamble</code>	62, 65, 118, 141, 144, 161, 190, 192–194
<code>\@org@glossaryentrynumbers</code>	127
<code>\@org@newglossaryentryprehook</code>	43, 44
<code>\@print@unsrt@glossary</code>	147
<code>\@printgloss@setsort</code>	126, 127
<code>\@printglossary</code>	62, 147
<code>\@printunsrt@glossary@handler</code>	148
<code>\@printunsrtglossary</code>	147
<code>\@rGLS</code>	158
<code>\@rGLS@</code>	158
<code>\@rGLSpl</code>	159
<code>\@rGLSpl@</code>	159
<code>\@rGls</code>	157
<code>\@rGls@</code>	158
<code>\@rGlspl</code>	158
<code>\@rGlspl@</code>	158
<code>\@rc@ifdefinable</code>	339
<code>\@rgls</code>	157
<code>\@rgls@</code>	157
<code>\@rglspl</code>	157
<code>\@rglspl@</code>	157
<code>\@sGlsXtrEnableOnTheFly</code>	59
<code>\@secondofthree</code>	76–78, 83, 85–88, 209, 211, 213, 215, 216, 218
<code>\@secondoftwo</code>	69, 75, 79–87, 89, 95, 122, 129, 187, 201, 209, 210, 212–219, 233, 255, 269, 291, 320, 321
<code>\@sglsxtr@provide@storagekey</code>	31
<code>\@star@or@long</code>	339
<code>\@starttoc</code>	319
<code>\@thirdofthree</code>	76–78, 84–87, 89, 210, 211, 213, 215, 217, 219, 321
<code>\@thirdoftwo</code>	79–83
<code>\@this@key</code>	187
<code>\@tracklang@lang</code>	40, 41
<code>\@warn@nomakeglossaries</code>	137
<code>\@xdy@main@language</code>	136
<code>\@xdycrossrefhook</code>	52
<code>\@xdy@language</code>	136
<code>\@xdy@locationclassorder</code>	52
<code>[package]</code>	342
<code>\\</code>	135, 343
<code>_</code>	58, 138, 139

A

<code>\AA</code>	366
<code>\aa</code>	366
<code>\AB</code>	20
<code>\Ab</code>	20
<code>\ab</code>	20
abbreviation styles:	
long-hyphen-postshort-hyphen	307, 309
long-hyphen-short-hyphen	303, 308
long-postshort-user	296
long-short-user	294
nolong-short	238
short	236
short-hyphen-long-hyphen	311, 313
short-hyphen-postlong-hyphen	312, 315
short-long-user	296
short-nolong	235, 238
short-nolong-desc	237
short-postlong-user	298
<code>\abbreviationsname</code>	20
<code>\abbrvpluralsuffix</code>	143, 170, 171, 205, 226, 228, 231, 233, 234, 236, 239, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316
<code>\ABP</code>	20
<code>\Abp</code>	20
<code>\abp</code>	20
<code>\AC</code>	21
<code>\Ac</code>	21
<code>\ac</code>	21
<code>\ACF</code>	21
<code>\Acf</code>	21
<code>\acf</code>	21
<code>\ACFP</code>	21
<code>\Acfp</code>	21
<code>\acfp</code>	21
<code>\ACL</code>	21
<code>\Acl</code>	21
<code>\acl</code>	21
<code>\ACLP</code>	21
<code>\Aclp</code>	21
<code>\aclp</code>	21
<code>\ACP</code>	21
<code>\Acp</code>	21
<code>\acp</code>	21
<code>\ACRfullfmt</code>	120

\cGls	20, 21, 107, 119	D	
\cglS	20, 21, 107, 119	datatool-base package	12
\cGLSformat	111	\DeclareAcronymList	120
\cGLSformat	111	\DeclareOption	5, 383
\cglSformat	110, 112	\DeclareOptionX	5, 27
\cGLSpl	20, 21, 107, 119	\def 9, 10, 13–17, 27, 29, 30, 33, 36, 43, 45, 49,	
\cGlspl	20, 21, 107, 119	53, 56, 59–62, 64, 66–87, 93, 95–99, 105,	
\cglSpl	20, 21, 107, 119	110–113, 120, 124, 126–130, 132–136,	
\cGLSplformat	112	148, 150, 151, 153, 156–159, 169–171,	
\cGlsplformat	111	191–196, 200–205, 208–219, 222, 302,	
\cglSplformat	111, 113	304, 307, 310, 312, 313, 341, 343, 347,	
\changes	324	348, 382, 396, 411–413, 417, 418, 427, 428	
\char	131	\defglSentryfmt	44–47
\columnwidth	63, 64	\define@boolkey	17, 18, 70, 90
\count@	110, 118, 119	\define@choicekey	
\csappto	42 7, 15, 17, 19, 22, 23, 26, 70, 128	
\csdef	31, 37, 38,	\define@key	13, 14,
42, 87–89, 109, 114, 115, 117, 176, 187,		19, 24, 25, 31, 53, 70, 74, 87, 129, 169, 202	
197, 198, 223, 224, 232, 255, 269, 290,		\DefineAcronymSynonyms	22
294, 296–298, 308, 310, 313, 315, 384, 402		\delimN	65, 66
\cseappto	47	\delimR	65, 66
\csedef	115	\descriptionname	427, 430, 432,
\csgdef	38,	433, 435, 436, 438, 439, 441, 442, 444, 446	
44–46, 58, 65, 109, 115, 117, 118, 402, 424		\detokenize	59
\cslet	38, 43, 44, 127	\dimen@	122, 402–410, 427, 428
\csletcs	38, 224	\dimen@i	405, 406
\csname	6, 32,	\dimen@ii	402, 403, 405, 406, 427
45, 53, 58, 63, 66, 69, 72, 74, 83–89, 91,		\dimexpr	63, 64, 401, 428, 429
100, 115, 124, 133, 136, 139, 140, 148,		\disable@keys	20, 30, 58, 141
153–156, 161, 181, 202, 209–219, 225, 411		\do 6, 24, 35, 56, 75, 107, 108, 119, 123,	
\cspreto	42	126, 133, 148, 155, 161, 180, 187, 195, 348	
\csuse . 9, 12, 33, 34, 45, 55, 65, 88, 89, 101–		\do@glS@link@checkfirsthyper	
103, 114–116, 118, 126, 131, 133, 134,	 33, 67, 69, 72, 83–87, 209–219	
145, 146, 149–151, 176, 187, 197, 198,		\do@glSdisablehyperinlist	72, 90
223, 224, 344, 347, 402, 403, 405, 406, 427		doc package	193
\csxdef	49, 53, 54, 115, 118	\dolistcsloop	35
\currentglossary	127, 145, 146, 423	\DTLifinlist	71, 124, 125, 129, 348
\CurrentOption	27, 383, 384	\DTLifint	131
\CurrentTrackedLanguage	381	E	
\CurrentTrackedLanguageTag	143	\eappto	
\CurrentTrackedScript	381, 382	13, 24, 44–46, 148, 151, 170–172, 190, 383	
\CurrentTrackedTag	338, 382	\edef 6, 8, 9, 11, 40, 44–47, 52, 54, 55,	
\CustomAbbreviationFields		57, 58, 72–74, 89, 91, 93, 95, 114, 115,	
.... 206, 226–230, 232, 234, 236, 239,		117, 123, 124, 129, 131, 132, 134–137,	
241–247, 249, 253, 254, 257–260, 262,		141, 145, 146, 156, 160, 181–184, 186–	
264, 267, 269, 271, 272, 274–279, 281,		188, 191, 193, 194, 202, 343, 348, 381,	
283, 286, 288, 290, 293, 294, 296, 298–		405, 406, 421, 422, 429, 431–438, 440–446	
301, 303–306, 308, 309, 311–313, 315–317		\eglssetwidest	404–410

<code>\egroup</code>	43, 44, 127	<code>\ExtraCustomAbbreviationFields</code>	170–172, 204, 206
<code>\else</code>	8–14, 17–20, 22–24, 27–30, 33, 36, 41, 42, 57–59, 64, 66, 69, 73, 74, 91, 92, 110, 122, 123, 125, 127–129, 132, 134, 135, 138, 140, 142, 156, 157, 189– 191, 193, 194, 201, 203, 205, 212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261–273, 275, 277–292, 294, 295, 297, 298, 300–302, 304, 307–310, 313, 314, 316, 317, 343–347, 385, 388– 398, 400, 411, 412, 418, 420, 422, 429–447		
<code>\emph</code>	270, 271		
<code>\empty</code>	134, 135, 343, 347, 348		
<code>\encapchar</code>	193		
<code>\end</code>	133, 138, 139, 148, 385, 387–394, 414–418, 421, 429–435, 437–443, 445, 446		
<code>\end@getprefix</code>	30		
<code>\end@glstr@display@loc</code>	134		
<code>\endcsname</code>	6, 32, 45, 53, 58, 63, 66, 69, 72, 74, 83–89, 91, 100, 115, 124, 133, 136, 139, 140, 148, 153–156, 161, 181, 202, 209–219, 225, 411		
<code>\endfoot</code>	427, 430, 432–434, 436, 438, 439, 441, 442, 444, 445		
<code>\endgroup</code>	8, 10, 33, 71, 75, 91, 145, 147, 160, 339, 343, 348		
<code>\endhead</code>	427, 430, 432–434, 436, 438, 439, 441, 442, 444, 445		
<code>\ensuremath</code>	26		
entry categories:			
abbreviation	219		
general	175, 177		
index	179		
<code>\entryname</code>	427, 428, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446		
<code>\epreto</code>	190		
<code>\equal</code>	139, 140, 198		
equation (counter)	18, 71		
<code>\escapechar</code>	339		
etoolbox package	5		
<code>\expandafter</code>	27, 32, 33, 35, 39, 49, 52, 55, 56, 59–61, 71, 88, 89, 93, 100, 106, 112–114, 124–126, 129, 133–135, 148, 151, 153, 160, 170, 171, 181, 184, 185, 187, 189, 190, 193, 201, 204, 205, 301, 339, 343, 421		
<code>\expandonce</code>	120, 151, 170–172, 191, 227, 305, 429–435, 437–446		
		F	
<code>\fi</code>	7–20, 22–24, 27, 28, 30, 33, 36, 41, 42, 49, 53–55, 57–60, 63, 64, 66, 69, 70, 72–74, 91, 92, 110, 118, 119, 122, 125– 129, 132, 134, 135, 137, 138, 140, 142– 144, 156, 157, 189–192, 194, 201, 203, 205, 212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261–273, 275, 277–292, 294, 295, 297, 298, 300, 302, 304, 307–310, 313, 314, 316, 317, 343– 347, 385, 388–398, 400, 402–412, 418, 420, 423, 428, 430, 431, 433–437, 439–447		
first use	448		
flag	448		
text	448		
<code>\firstacronymfont</code>	121, 122		
fontspec package	143		
<code>\footnote</code>	230		
<code>\forall glossaries</code>	55, 147, 179, 181, 404–410		
<code>\forall glsentries</code>	57, 110, 118		
<code>\ForEachTrackedDialect</code>	338, 382		
<code>\foreignlanguage</code>	39, 41		
<code>\forall glsentries</code>	6, 55, 179, 181, 404–410		
<code>\forall listcsloop</code>	35, 118, 133		
<code>\forall listloop</code>	106, 130, 196		
<code>\futurelet</code>	200		
		G	
<code>\gdef</code>	65, 192, 193		
<code>\Genacrfullformat</code>	120, 121		
<code>\genacrfullformat</code>	120, 121		
<code>\GenericAcronymFields</code>	120		
<code>\Genplacrfullformat</code>	121		
<code>\genplacrfullformat</code>	121		
<code>\GetTrackedDialectFromLanguageTag</code> ..	39		
<code>\GetTrackedDialectToMapping</code>	40		
<code>\glo@grabfirst</code>	151		
<code>\glo@name</code>	184, 185, 189		
<code>\gloaliaslabel</code>	94, 95		
<code>\global</code>	10, 43, 44, 57, 127, 151		
<code>\glo linkprefix</code>	70, 72–74, 95, 129, 143		
glossaries package	12, 15, 27–29, 41, 49, 52–54, 126, 384		
glossaries-accsupp package	23, 27, 162, 205		
glossaries-extra package	2, 382		
glossaries-extra-bib2gls package	15, 16, 30, 338, 382, 428		

glossaries-extra-stylemods package	24, 197, 338, 345
glossaries-prefix package	348
glossaries-stylemods package	419
glossaries.sty package	44
\GlossariesExtraWarning	6, 16, 18, 23, 30, 40–42, 60, 62, 73, 121, 124, 134, 138, 140, 141, 147, 181–184, 186, 188, 195, 224, 339, 346, 347
\GlossariesExtraWarningNoLine	18, 110, 119
\GlossariesWarning	65, 124, 126, 130, 131, 222
\GlossariesWarningNoLine	124, 137
glossary styles:	
altlist	385
altlistgroup	386
altlisthypergroup	387
alttree	344, 345, 401, 402, 411, 427
alttreegroup	412
alttreehypergroup	413
index	396
indexgroup	397
indexhypergroup	397
inline	395
list	385
listdotted	384
listdottedstyle	385
listgroup	386
listhypergroup	386
longragged-booktabs	429
mcolalttree	416
mcolalttreegroup	417
mcolalttreehypergroup	417
mcolalttreespannav	418
mcolindexgroup	413
mcolindexhypergroup	413
mcolindexspannav	414
mcoltreegroup	414
mcoltreehypergroup	415
mcoltreenonamegroup	415
mcoltreenonamehypergroup	416
mcoltreenonamespannav	416
mcoltreespannav	415
name-desc	432
sublistdotted	385
tree	397
treegroup	398
treehypergroup	399
treenoname	399
treenonamegroup	400
treenonamehypergroup	400
glossary-bookindex package	384
glossary-hypernav package	93
glossary-long package	389
glossary-longbooktabs package	389
glossary-longextra package	344–346
glossary-tree package	345, 346, 396
\glossaryentrynumbers ...	66, 127, 151, 152
\glossaryheader	133, 148, 385–394, 396–400, 411, 413–417, 421, 429–435, 437–443, 445, 446
\glossaryname	126
\glossarypostamble	133, 148, 150
\glossarypreamble	133, 147
\glossarysection ...	133, 139, 141, 147, 150
\glossarytitle	45, 126, 127, 133, 139, 141, 147
\glossarytoctitle	45, 126, 133, 139, 141, 147
\glossentry	127, 152, 384–394, 396, 398, 400, 411, 421, 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glossentrydesc	384, 385, 387–395, 397–399, 401, 425
\glossentryname	145, 384–394, 396, 398, 400, 411, 412, 419, 425
\glossentrynameother	147
\glossentrysymbol	388–395, 398, 399, 401, 425
\glossxtrsetopts	197
\GLS	107, 119, 155
\GLs	61, 107, 119, 155
\glS	41, 42, 60, 62, 107, 119, 125, 138, 155
\glS@assign@desc	43, 44
\glS@assign@field	13, 14, 32, 87
\glS@checkseeallowed	57, 59, 123
\glS@codepage	137
\glS@defdocnewglossaryentry	57, 108, 116
\glS@defglossaryentry ...	43, 44, 57, 60, 61
\glS@dotocitle	127
\glS@glossary	53
\glS@grplabel	93
\glS@ifnotmeasuring	12, 107
\glS@level	151
\glS@noidxglossary	124
\glS@org@glossaryentryfield	127
\glS@org@glossarysubentryfield	127
\glS@orgTrackLangRequireDialectPrefix	382
\glS@save@numberlist	64, 66
\glS@set@xr@key	53
\glS@tmplen	404–410, 412, 428

<code>\gls@type</code>	124	271, 273, 275, 277–292, 294, 295, 298, 300, 301, 303, 305, 306, 309, 311, 316, 317
<code>\glsabbrvdefaultfont</code> ...	208, 226, 228, 231, 233, 234, 236, 239, 292, 302, 305, 315	
<code>\glsabbrvemfont</code>	270–279, 281, 283, 285, 287, 289–291	
<code>\glsabbrvfont</code>	97, 98, 121, 208, 212, 213, 216, 217, 219, 221, 222, 225–234, 236, 239, 241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299–301, 303, 305, 307–309, 311, 313, 314, 316	
<code>\glsabbrvhyphenfont</code>	302–304, 308, 310–313, 315	
<code>\glsabbrvonlyfont</code>	315, 316, 318	
<code>\glsabbrvscfont</code> .	242–248, 250, 251, 253–255	
<code>\glsabbrvsmfont</code>	256–260, 262, 264, 265, 267, 269	
<code>\glsabbrvuserfont</code>	292–298, 300	
<code>\GLSaccessdesc</code>	79	
<code>\Glsaccessdesc</code>	79, 182, 194	
<code>\glsaccessdesc</code>	79, 182, 198, 199	
<code>\GLSaccessdescplural</code>	80	
<code>\Glsaccessdescplural</code>	80	
<code>\glsaccessdescplural</code>	80	
<code>\GLSaccessfirst</code>	77	
<code>\Glsaccessfirst</code>	77	
<code>\glsaccessfirst</code>	77	
<code>\GLSaccessfirstplural</code>	78	
<code>\Glsaccessfirstplural</code>	78	
<code>\glsaccessfirstplural</code>	78	
<code>\Glsaccesslong</code>	86, 207, 215, 226, 235, 239, 240, 243, 249, 250, 252, 257, 263–266, 272, 273, 281–287, 294, 295, 303, 305, 306, 309, 311, 317	
<code>\glsaccesslong</code>	85, 86, 206, 214, 215, 226, 228, 229, 231–234, 236–240, 243, 245–254, 256, 257, 259, 261–268, 270, 271, 273, 275, 277, 278, 280–291, 294, 295, 298, 300, 303, 305, 306, 309, 311, 316, 317	
<code>\Glsaccesslongpl</code>	87, 207, 218, 226, 235, 240, 243, 249, 250, 252, 258, 263–266, 272, 273, 281–287, 294, 295, 303, 305, 306, 309, 311, 317	
<code>\glsaccesslongpl</code>	86, 87, 207, 218, 219, 226, 229, 231–240, 243, 245–254, 256, 257, 259, 261–268, 270, 271, 273, 275, 277, 278, 280–291, 294, 295, 298, 300, 303, 305, 306, 309, 311, 316, 317	
<code>\GLSaccessname</code>	51, 79	
<code>\Glsaccessname</code>	50, 51, 79	
<code>\glsaccessname</code>	50, 51, 79	
<code>\GLSaccessplural</code>	78	
<code>\Glsaccessplural</code>	77	
<code>\glsaccessplural</code>	77	
<code>\GLSaccessshort</code>	51	
<code>\Glsaccessshort</code>	50, 51, 84, 213, 222, 229, 231–233, 237, 238, 245, 247, 248, 254–256, 259, 261, 262, 268, 270, 275, 277, 278, 280, 289–291, 297, 298, 300, 309, 314	
<code>\glsaccessshort</code>	50, 51, 83, 84, 206, 207, 212, 213, 221, 226, 228, 231, 233–238, 240, 243, 245–250, 252–257, 259–273, 275, 277–282, 284–287, 289, 291, 294, 295, 297, 298, 300, 303, 305, 306, 308, 311, 314, 317	
<code>\Glsaccessshortpl</code>	85, 216, 222, 229, 231–234, 237, 238, 245, 247, 248, 254, 256, 259, 261, 262, 268, 270, 275, 277, 278, 280, 289–292, 297, 298, 300, 309, 314, 317	
<code>\glsaccessshortpl</code>	84, 85, 207, 216, 217, 222, 226–228, 231, 233–238, 240, 243, 245–259, 261–263, 265, 266, 268–273, 275, 277–282, 284, 286, 287, 289–291, 294, 295, 297, 298, 300, 303, 306, 308, 311, 314, 317	
<code>\GLSaccesssymbol</code>	80	
<code>\Glsaccesssymbol</code>	80, 194	
<code>\glsaccesssymbol</code>	80, 194, 199	
<code>\GLSaccesssymbolplural</code>	81	
<code>\Glsaccesssymbolplural</code>	81	
<code>\glsaccesssymbolplural</code>	81	
<code>\GLSaccessstext</code>	76	
<code>\Glsaccessstext</code>	76	
<code>\glsaccessstext</code>	50, 76	
<code>\glsacrshortcutstrue</code>	22, 23	
<code>\glsacspacemax</code>	122	
<code>\glsadd</code>	33, 56, 71, 75, 138	
<code>\glsadd options</code>		
<code>theHvalue</code>	9	
<code>thevalue</code>	9, 10	
<code>\glsaddpostsetkeys</code>	10, 75	
<code>\glsaddpresetkeys</code>	10, 74	
<code>\glsaddstoragekey</code>	55, 175, 341	
<code>\glsbackslash</code>	59	
<code>\glsapscase</code>	69, 75–89, 209–221	

<code>\glscategory</code> .. 67, 75, 89, 97, 98, 176–178, 181–188, 194, 197, 198, 209–213, 216, 217	<code>\Glsentryfull</code> 121
<code>\glscategorylabel</code> 89, 170–172, 202, 204, 205, 232, 255, 269, 290, 294, 296–298, 308, 310, 313, 315	<code>\Glsentryfull</code> 121
<code>\glsclosebrace</code> 52, 139, 140	<code>\Glsentryfullpl</code> 121
<code>\glscounter</code> 9, 18	<code>\Glsentryfullpl</code> 121
<code>\glscurrententrylabel</code> 64– 66, 127, 136, 145, 146, 148, 149, 196, 197	<code>\Glsentryitem</code> 145, 146, 384–394, 396, 398, 400, 411, 422, 425
<code>\glscurrentfieldvalue</code> 33–36, 39, 40, 50, 51, 292, 340	<code>\Glsentrylong</code> 98, 99, 113, 159, 168, 175
<code>\glscustomtext</code> .. 67, 69, 83–87, 209–219, 221	<code>\Glsentrylong</code> 98, 99, 113, 159, 168, 175, 232, 255, 269, 290, 297, 298, 313, 335
<code>\glsdefaulttype</code> . 6, 19, 42, 126, 137, 147, 149	<code>\Glsentrylongpl</code> 98, 99, 113, 169, 175
<code>\glsdescriptionaccessdisplay</code> .. 166, 182	<code>\Glsentrylongpl</code> ... 98, 99, 113, 169, 175, 336
<code>\glsdescriptionpluralaccessdisplay</code> 166, 167	<code>\Glsentrylongplural</code> 160
<code>\glsdescwidth</code> 387–394, 426, 428, 429	<code>\Glsentrylongplural</code> 159
<code>\glsdetoklabel</code> 8, 9, 31, 34–38, 41–44, 47–49, 52, 56, 58, 59, 72, 74, 91, 95, 108, 109, 114–118, 124, 127, 130, 131, 145, 146, 151, 154–156, 181, 184, 185, 189, 405, 406	<code>\Glsentryname</code> 162, 172, 183, 185, 186
<code>\glsdisp</code> 350	<code>\Glsentryname</code> 145, 162, 172, 190, 332, 333, 404–410, 424
<code>\glsdisplaynumberlist</code> 125, 130	<code>\Glsentrynumberlist</code> 125, 131, 408–410
<code>\glsdohyperlink</code> 93, 95, 96, 344	<code>\Glsentryplural</code> 163, 173
<code>\glsdohypertarget</code> 95, 129	<code>\Glsentryplural</code> 163, 172, 173, 333, 334
<code>\glsdoifexists</code> 17, 27, 37, 43, 47, 49–51, 53, 67, 69, 74, 83–87, 95, 107, 124, 130, 131, 145, 146, 209–219	<code>\Glsentryprevcount</code> 108, 110, 116
<code>\glsdoifexistsordo</code> 33, 34, 69	<code>\Glsentryprevmaxcount</code> 116
<code>\glsdoifexistsorwarn</code> 17, 181, 182, 194	<code>\Glsentryprevtotalcount</code> 116
<code>\glsdoifnoexists</code> 43	<code>\Glsentryshort</code> 97, 98, 167, 174
<code>\glsdonohyperlink</code> 74, 95, 96	<code>\Glsentryshort</code> 97, 98, 122, 167, 174, 294, 296, 307, 331, 332
<code>\glsdosanitizesort</code> 125	<code>\Glsentryshortpl</code> 98, 99, 168, 174
<code>\glsenableentrycount</code> 107, 110, 118	<code>\Glsentryshortpl</code> 98, 99, 168, 174, 175, 331, 332
<code>\glsenableentryunitcount</code> 109, 119	<code>\Glsentrysymbol</code> 165, 173
<code>\GlsEntryCounterLabelPrefix</code> 41	<code>\Glsentrysymbol</code> 165, 173, 407–409
<code>\glsentrycurrcount</code> 108, 110, 116	<code>\Glsentrysymbolplural</code> 166, 173
<code>\Glsentrydesc</code> 166, 174, 182	<code>\Glsentrysymbolplural</code> .. 165, 166, 173, 174
<code>\Glsentrydesc</code> 166, 174, 183	<code>\Glsentrytext</code> 163, 172
<code>\Glsentrydescplural</code> 167, 174	<code>\Glsentrytext</code> 95, 162, 163, 172, 333
<code>\Glsentrydescplural</code> 166, 167, 174	<code>\Glsentrytype</code> 146
<code>\Glsentryfirst</code> 113, 159, 164, 173	<code>\Glsentryuseri</code> 81
<code>\Glsentryfirst</code> 113, 159, 164, 173, 334	<code>\Glsentryuseri</code> 81
<code>\Glsentryfirstplural</code> ... 113, 160, 164, 173	<code>\Glsentryuserii</code> 81
<code>\Glsentryfirstplural</code> 113, 159, 164, 165, 173, 335	<code>\Glsentryuserii</code> 82
<code>\glsentryfmt</code> 44–47	<code>\Glsentryuseriii</code> 82
	<code>\Glsentryuseriii</code> 82
	<code>\Glsentryuseriv</code> 82
	<code>\Glsentryuseriv</code> 82
	<code>\Glsentryuserv</code> 82
	<code>\Glsentryuserv</code> 83
	<code>\Glsentryuservi</code> 83
	<code>\Glsentryuservi</code> 83
	<code>\glsextrapostnamehook</code> 187
	<code>\Glsfieldfetch</code> 94

<code>\glsfieldxdef</code>	180, 181	<code>\glsgetgrouptitle</code>	386, 387, 397, 399–401, 412–418
<code>\glsFindWidestLevelTwo</code>	347	<code>\glsgetwidestname</code>	402
<code>\glsFindWidestTopLevelName</code>	346	<code>\glsgroupheading</code>	151, 385–394, 396–401, 411–418, 423, 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
<code>\glsfindwidesttoplevelname</code>	403	<code>\glsgroupskip</code>	151, 385, 387–395, 397, 398, 400, 412, 423, 430, 431, 433, 434, 436, 437, 439, 441, 442, 444, 445, 447
<code>\GLSfirst</code>	326	<code>\glshasattribute</code>	73, 94, 110, 114, 115, 117, 118, 136, 156, 160, 181–186, 188, 190, 226–230, 232, 236–238, 241–244, 246, 253, 255, 257–260, 267, 269, 271–274, 276, 277, 285, 288–290, 293, 294, 296, 297, 299– 301, 303–306, 308, 310–313, 315, 316, 318
<code>\Glsfirst</code>	327	<code>\glshascategoryattribute</code>	177
<code>\glsfirst</code>	326	<code>\glshex</code> ..	352–358, 360–366, 368–371, 373–381
<code>\glsfirstabbrvdefaultfont</code>	208, 226, 228, 231, 233, 234, 236, 239, 305	<code>\glshyperlink</code>	95, 340
<code>\glsfirstabbrvfont</code>	271–292	<code>\glshypernavsep</code>	133
<code>\glsfirstabbrvfont</code>	121, 206, 207, 226–240, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 306, 308, 311, 313, 316	<code>\glshypernumber</code>	136, 189, 341–343
<code>\glsfirstabbrvfont</code>	226–240, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 306, 308, 311, 313, 316	<code>\glsifattribute</code>	69, 70, 76, 90, 92, 101, 160, 179, 182–185, 188, 189, 196, 199, 200, 322–331
<code>\glsfirstabbrvhypenfont</code>	302–304, 307, 308, 310–315	<code>\glsifcategory</code>	179
<code>\glsfirstabbrvonlyfont</code>	316, 317	<code>\glsifcategoryattribute</code>	89, 170–172, 177, 178, 204, 205
<code>\glsfirstabbrvscfont</code>	242–256	<code>\glsifnotregular</code>	76
<code>\glsfirstabbrvsmfont</code>	257–270	<code>\glsifnotregularcategory</code>	178
<code>\glsfirstabbrvuserfont</code>	293–301	<code>\glsifplural</code>	69, 75, 77, 78, 80, 81, 83–87, 199, 200, 209–220
<code>\glsfirstaccessdisplay</code>	164	<code>\glsifregular</code>	67, 75, 113, 159, 160
<code>\glsfirstlongdefaultfont</code>	226, 228, 234, 236, 239, 242–252, 257– 267, 271, 272, 274, 275, 278–283, 285, 286	<code>\glsifregularcategory</code>	178
<code>\glsfirstlongfont</code>	273, 274, 276, 277, 283, 284, 286–288	<code>\glsifusetranslator</code>	45
<code>\glsfirstlongfont</code> ...	206, 207, 226–229, 231, 233–241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316	<code>\glsignore</code>	65, 66
<code>\glsfirstlonggemfont</code>	230–234, 253–256, 267–270, 288–292	<code>\glsinlinedescformat</code>	395
<code>\glsfirstlongfont</code> ...	206, 207, 226–229, 231, 233–241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316	<code>\glsinlinesubdescformat</code>	395
<code>\glsfirstlongfontfootnotefont</code>	230–234, 253–256, 267–270, 288–292	<code>\glsinsert</code>	69, 75, 83–87, 209–221, 301, 308, 310, 313, 315
<code>\glsfirstlonghyphenfont</code>	302–313	<code>\glskeylisttok</code>	120, 204, 206
<code>\glsfirstlongonlyfont</code>	316–318	<code>\glslabel</code>	8, 9, 33, 47, 67, 69–74, 89–91, 94, 95, 122, 156, 160, 161, 197–199, 219–221, 232, 255, 269, 290, 294, 296–298, 308, 310, 313, 315
<code>\glsfirstlonguserfont</code>	293–301	<code>\glslabeltok</code> ...	120, 204, 206, 226–232, 234, 236–239, 241–247, 250, 253, 255, 257–260, 262, 264, 267, 269, 271–279, 281, 283, 285, 287–291, 293, 294, 296– 301, 303–306, 308, 310–313, 315, 316, 318
<code>\GLSfirstplural</code>	327		
<code>\Glsfirstplural</code>	327		
<code>\glsfirstplural</code>	327		
<code>\glsfirstpluralaccessdisplay</code> ..	164, 165		
<code>\glsforeachincategory</code>	222		
<code>\glsgenentryfmt</code>	67		
<code>\glsgetattribute</code>	73, 94, 110, 114– 116, 136, 156, 160, 181–184, 186, 188, 190		
<code>\glsgetcategoryattribute</code>	176		

<code>\glsletentryfield</code>	191	<code>\glslongextraLocationDescNameHeader</code>	434
<code>\glslink</code>	120, 121, 349	<code>\glslongextraLocationDescNameTabularFooter</code>	
<code>\glslink options</code>		433, 434
counter	10	<code>\glslongextraLocationDescNameTabularHeader</code>	
format	189	433, 434
hyper	318	<code>\glslongextraLocationDescSymNameHeader</code>	
hyperoutside	70	446
noindex	8, 9, 90, 318	<code>\glslongextraLocationDescSymNameTabularFooter</code>	
textformat	73	445, 446
theHvalue	73	<code>\glslongextraLocationDescSymNameTabularHeader</code>	
thevalue	73, 152	445, 446
wrgloss	8, 69, 70	<code>\glslongextraLocationFmt</code>	
<code>\glslinkcheckfirsthyperhook</code>	90	431, 434, 437, 440, 443, 447
<code>\glslinkpostsetkeys</code>	10, 73, 156	<code>\glslongextraLocationSymDescNameHeader</code>	
<code>\glslinkpresetkeys</code>	10, 72, 156	443
<code>\glslinkvar</code>	92, 93	<code>\glslongextraLocationSymDescNameTabularFooter</code>	
<code>\glslistchildpostlocation</code>	385	442, 443
<code>\glslistchildprelocation</code>	385, 386	<code>\glslongextraLocationSymDescNameTabularHeader</code>	
<code>\glslistdesc</code>	385, 386	442, 443
<code>\glslistdottedwidth</code>	384	<code>\glslongextraLocSetDescWidth</code>	431, 433, 434
<code>\glslistgroupheaderfmt</code>	386, 387	<code>\glslongextraNameAlign</code>	
<code>\glslistnavigationitem</code>	386, 387	429–432, 434, 435, 437, 438, 440–446
<code>\glslistprelocation</code>	385, 386	<code>\glslongextraNameDescHeader</code>	430
<code>\glslocalunset</code>	69, 156	<code>\glslongextraNameDescLocationHeader</code>	431
<code>\glslongaccessdisplay</code>	168	<code>\glslongextraNameDescLocationTabularFooter</code>	
<code>\glslongdefaultfont</code>	208, 226, 228,	430, 431
	230, 234, 236, 239, 243, 245, 246, 248–	<code>\glslongextraNameDescLocationTabularHeader</code>	
	252, 257, 259, 260, 262, 264–266, 271,	430, 431
	275, 278, 279, 281, 282, 285, 293, 302, 315	<code>\glslongextraNameDescSymHeader</code>	435
<code>\glslongemfont</code> ..	271, 273, 276, 283, 286, 287	<code>\glslongextraNameDescSymLocationHeader</code>	
<code>\glslongextraDescAlign</code>	437
.....	429–432, 434, 435, 437–446	<code>\glslongextraNameDescSymLocationTabularFooter</code>	
<code>\glslongextraDescFmt</code>	426, 430, 431, 433,	436, 437
	434, 436, 437, 439, 440, 442, 443, 445, 447	<code>\glslongextraNameDescSymLocationTabularHeader</code>	
<code>\glslongextraDescNameHeader</code>	433	436, 437
<code>\glslongextraDescNameTabularFooter</code>	432	<code>\glslongextraNameDescSymTabularFooter</code>	
<code>\glslongextraDescNameTabularHeader</code>	432	434, 435
<code>\glslongextraDescSymNameHeader</code>	445	<code>\glslongextraNameDescSymTabularHeader</code>	
<code>\glslongextraDescSymNameTabularFooter</code>		434, 435
.....	444, 445	<code>\glslongextraNameDescTabularFooter</code>	
<code>\glslongextraDescSymNameTabularHeader</code>		427, 429
.....	444, 445	<code>\glslongextraNameDescTabularHeader</code>	
<code>\glslongextraGroupHeading</code>	430, 431, 433,	427, 429
	434, 436, 437, 439, 440, 442, 443, 445, 447	<code>\glslongextraNameFmt</code>	430, 431, 433,
<code>\glslongextraHeaderFmt</code>			434, 436, 437, 439, 440, 442, 443, 445, 447
.....	427, 428, 430, 432,	<code>\glslongextraNameSymDescHeader</code>	439
	433, 435, 436, 438, 439, 441, 442, 444–446	<code>\glslongextraNameSymDescLocationHeader</code>	
<code>\glslongextraLocationAlign</code>	440
.....	431, 433, 434, 437, 440, 443, 446		

<code>\glslongextraNameSymDescLocationTabularFooter</code>	244, 245, 249, 253, 257–260, 264, 267, 439, 440	<code>\glslongpluralaccessdisplay</code> 169
<code>\glslongextraNameSymDescLocationTabularHeader</code>	296–301, 303–306, 308, 310–312, 316, 318 439, 440	<code>\glslongtok</code>
<code>\glslongextraNameSymDescTabularFooter</code> 438	<code>\glslongtok</code>	. 120, 204, 206, 226–230, 232, 234, 236, 239, 241–247, 249, 250, 253, 254, 257, 258, 260, 262, 264, 267, 269, 271–279, 281, 283, 286, 288, 290, 293, 294, 296– 301, 303–306, 308, 309, 311–313, 316–318
<code>\glslongextraNameSymDescTabularHeader</code> 438	<code>\glslonguserfont</code> 293, 295–298, 300
<code>\glslongextraSetDescWidth</code>	.. 428, 429, 432	<code>\glsmcols</code> 414–418
<code>\glslongextraSubDescFmt</code>	430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447	<code>\GLSname</code> 324
<code>\glslongextraSubLocationFmt</code> 431, 434, 437, 440, 444, 447	<code>\Glsname</code> 324
<code>\glslongextraSubNameFmt</code>	430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447	<code>\glsname</code> 324
<code>\glslongextraSubSymbolFmt</code> 436, 437, 439, 440, 442, 444, 445, 447	<code>\glsnameaccessdisplay</code>	.. 162, 183, 184, 186
<code>\glslongextraSymbolAlign</code> 435, 437, 438, 440–446	<code>\glsnamefont</code> 183, 185, 186, 188, 428
<code>\glslongextraSymbolFmt</code> 426, 436, 437, 439, 440, 442, 443, 445, 447	<code>\glsnavhyperlink</code> 133
<code>\glslongextraSymDescNameHeader</code> 442	<code>\glsnavhyperlinkname</code> 94
<code>\glslongextraSymDescNameTabularFooter</code> 441	<code>\glsnavhypertarget</code> 386, 387, 397, 399, 401, 413–418
<code>\glslongextraSymDescNameTabularHeader</code> 441	<code>\glsnavigation</code> 386, 387, 397, 399, 401, 413–418
<code>\glslongextraSymLocSetDescWidth</code> 436, 437, 440, 443, 446	<code>\glsnextpages</code> 127
<code>\glslongextraSymSetDescWidth</code> 429, 435, 438, 441, 442, 444, 445	<code>\glsnoidxdisplayloc</code> 130, 131, 342
<code>\glslongextraTabularVAlign</code>	. 429, 431– 433, 435, 436, 438, 440, 441, 443, 444, 446	<code>\glsnoidxdisplaylocclsthandler</code> 130
<code>\glslongextraUpdateWidest</code> 344–346	<code>\glsnoidxloclist</code> 131, 151, 152
<code>\glslongextraUpdateWidestChild</code>	344–347	<code>\glsnoidxnumberlistloophandler</code> 130
<code>\GlsLongExtraUseTabularfalse</code> 429	<code>\glsnonextpages</code> 127
<code>\glslongfont</code> 98, 208, 214, 215, 218, 219, 226–229, 231, 233, 234, 236, 239–241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316, 317	<code>\glsnonumberlistfalse</code> 64
<code>\glslongfootnotefont</code> 230, 231, 233, 253, 255, 267, 269, 289, 291	<code>\glsnonumberlisttrue</code> 64
<code>\glslonghyphenfont</code> 302, 303, 305, 306, 308, 311, 313	<code>\glsnopostdotfalse</code> 128
<code>\glslongonlyfont</code> 316	<code>\glsnopostdottrue</code> 128
<code>\glslongpltok</code> 205, 206, 226–230, 239, 241, 242,	<code>\glsnumberlistloop</code> 125
		<code>\glsnumlistlastsep</code> 130
		<code>\glsnumlistsep</code> 130
		<code>\glsopenbrace</code> 52, 139, 140
		<code>\glsorder</code> 123
		<code>\glspagelistwidth</code>	388, 390, 392, 394, 427, 429
		<code>\glspar</code> 150
		<code>\glspatchLToutput</code> 429, 431, 432, 434, 435, 437, 438, 440, 442, 443, 445, 446
		<code>\glspenaltygroupskip</code>	... 430, 431, 433, 434, 436, 437, 439, 441, 442, 444, 445, 447
		<code>\GLSpl</code> 107, 119, 155
		<code>\Glspl</code> 61, 107, 119, 155
		<code>\glspl</code> 61, 107, 119, 155
		<code>\GLSplural</code> 325, 326
		<code>\Glsplural</code> 326

<code>\glsplural</code>	325	<code>\glstextup</code>	242
<code>\glspluralaccessdisplay</code>	163	<code>\glstreechilddesc</code>	397, 398
<code>\glspluralsuffix</code>	143, 204, 208	<code>\glstreechildpredesc</code>	398
<code>\glspostdescription</code>	18, 19, 128, 196, 384, 385, 387–395, 397–399, 401, 425	<code>\glstreechildprelocation</code> ...	397, 398, 400
<code>\glspostinline</code>	395	<code>\glstreechildsymbol</code>	396, 398
<code>\glspostlinkhook</code>	68, 69, 83–87, 100, 209–219	<code>\glstreedefaultnamefmt</code>	395, 396
<code>\glsprestandardsort</code>	125	<code>\glstreedesc</code>	396, 398
<code>\glsresetentrylist</code>	133, 148	<code>\glstreegroupheaderfmt</code>	
<code>\glssee</code>	53–55	397, 399–401, 412–418, 420
<code>\glsseeformat</code>	49, 52, 58, 124, 130, 131	<code>\glstreeindent</code>	398, 400, 410–412
<code>\glsseelist</code>	52	<code>\glstreeitem</code>	396, 414, 422
<code>\glssetabbrvfmt</code>	67, 75, 97, 98, 181–186, 188, 194, 209–213, 216, 217, 219	<code>\glstreenamebox</code>	411, 412
<code>\glssetattribute</code>	226– 232, 234, 236–239, 241–244, 246, 247, 250, 253, 255, 257–260, 262, 264, 267, 269, 271–274, 276–279, 281, 283, 285, 287–289, 291, 293, 294, 296, 297, 299– 301, 303–306, 308, 310–313, 315, 316, 318	<code>\glstreenamefmt</code>	
<code>\glssetcategoryattribute</code>	108, 119, 122, 155, 161, 176, 177, 179, 180, 195	395, 396, 398, 400, 402, 404–412
<code>\glssetnoexpandfield</code>	13, 14	<code>\glstreenavigationfmt</code>	397, 399, 401, 413–418
<code>\glssettoctitle</code>	127	<code>\glstreenonamechilddesc</code>	400
<code>\glssetwidest</code>	345, 346	<code>\glstreenonamedesc</code>	400
<code>\glsshortaccessdisplay</code>	167	<code>\glstreenonamesymbol</code>	400
<code>\glsshortpltok</code>		<code>\glstreepredesc</code>	397, 399
....	205, 206, 226–230, 232, 234, 236, 243–247, 253, 254, 257–260, 262, 267, 269, 271–279, 288–290, 293, 294, 296– 301, 303, 304, 308, 310–313, 315, 316, 318	<code>\glstreeprelocation</code>	396, 398, 400, 401
<code>\glsshortpluralaccessdisplay</code>	168	<code>\glstreesubitem</code>	396, 422
<code>\glsshorttok</code>	120, 204–206, 225–230, 232, 234, 236, 241, 242, 244–247, 249, 253, 254, 257, 258, 260, 262, 264, 267, 269, 271–279, 281, 283, 288, 290, 293, 294, 296, 298–301, 303, 304, 306, 308, 310–313, 315, 316, 318	<code>\glstreesubsubitem</code>	396, 423
<code>\glssubentryitem</code>		<code>\glstreesymbol</code>	396, 398
....	146, 384–394, 396, 398, 400, 412, 422, 426	<code>\GlsrLetField</code>	38
<code>\glssymbolaccessdisplay</code>	165	<code>\glstype</code>	69, 72, 83–87, 156, 209–219
<code>\glssymbolpluralaccessdisplay</code>	165, 166	<code>\glunset</code>	56, 69, 110–112, 157
<code>\glstarget</code>	145, 147, 384– 396, 398, 400, 411, 412, 422, 423, 425, 426	<code>\glupdatewidest</code>	344, 345
<code>\GLStext</code>	324, 325	<code>\gluserdescription</code>	293, 294, 297, 300
<code>\Glstext</code>	325	<code>\glswrite</code>	52, 123
<code>\glstext</code>	324, 325	<code>\glswriteentry</code>	8, 9
<code>\glstextaccessdisplay</code>	162, 163	<code>\Glsxtr</code>	62
<code>\glstextformat</code>	69, 73	<code>\glxtr</code>	62
		<code>\glxtr@do@wrglossary</code>	8, 10, 13, 15
		<code>\glxtr@addloclistfield</code>	15, 16
		<code>\glxtr@addunused</code>	56
		<code>\glxtr@applyabbrvfmt</code>	219
		<code>\glxtr@applyabbrvstyle</code>	202, 204, 223
		<code>\glxtr@counterrecord</code>	145
		<code>\glxtr@dblfloat</code>	18
		<code>\glxtr@do@alsoindex@wrglossary</code>	16
		<code>\glxtr@do@autoadd</code>	9, 71, 72
		<code>\glxtr@doooption</code>	5, 18, 19, 26, 27, 29
		<code>\glxtr@fields</code>	143
		<code>\glxtr@float</code>	18
		<code>\glxtr@headentry@p</code>	101, 102
		<code>\glxtr@hyperoutsidefalse</code>	70
		<code>\glxtr@hyperoutsidettrue</code>	70
		<code>\glxtr@ifnextpunc</code>	200
		<code>\glxtr@ifpunctoken</code>	201

<code>\glxtr@inc@linkcount</code>	72, 161	<code>\glxtraddallcrossrefs</code>	55
<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 26, 29	<code>\glxtralias</code>	91
<code>\glxtr@indexonly@saveentrycounter</code>	15, 16, 29	<code>\glxtrAltTreeIndent</code>	401, 402
<code>\glxtr@keylist</code>	60, 61	<code>\glxtralttreeInit</code>	411, 417, 418
<code>\glxtr@label</code>	424	<code>\glxtrAltTreePar</code>	401
<code>\glxtr@langtag</code>	143	<code>\glxtrAltTreeSetHangIndent</code> ...	401, 411
<code>\glxtr@linkprefix</code>	143	<code>\glxtrAltTreeSetSubHangIndent</code>	412
<code>\glxtr@loaddialect</code>	338, 382	<code>\glxtralttreeSubSymbolDescLocation</code>	412
<code>\glxtr@locationhypertext</code>	343	<code>\glxtralttreeSymbolDescLocation</code> ..	401, 412
<code>\glxtr@makeglossaries</code>	123	<code>\glxtrassignfieldfont</code>	76–83
<code>\glxtr@newabbreviation</code>	121, 203	<code>\glxtrautoindex</code>	190
<code>\glxtr@next</code>	201	<code>\glxtrautoindexassignsort</code>	190
<code>\glxtr@org@odo@wrglossary</code>	29	<code>\glxtrautoindexentry</code>	190
<code>\glxtr@org@dohyperlink</code>	93	<code>\glxtrautoindexesc</code>	190
<code>\glxtr@org@getgrouptitle</code>	132	<code>\glxtrbibaddress</code>	341
<code>\glxtr@org@newignoredglossary</code>	44	<code>\glxtrbibauthor</code>	341
<code>\glxtr@org@makenoidxglossaries</code>	57	<code>\glxtrbibbooktitle</code>	341
<code>\glxtr@pluralsuffixes</code>	143	<code>\glxtrbibchapter</code>	341
<code>\glxtr@process</code>	148, 149	<code>\glxtrbibedition</code>	341
<code>\glxtr@provideignoredglossary</code>	46	<code>\glxtrbibhowpublished</code>	341
<code>\glxtr@punclist</code>	200, 201	<code>\glxtrbibinstitution</code>	341
<code>\glxtr@record</code>	11, 143	<code>\glxtrbibjournal</code>	341
<code>\glxtr@record@nameref</code>	12, 143	<code>\glxtrbibmonth</code>	341
<code>\glxtr@record@nr</code>	15	<code>\glxtrbibnote</code>	341
<code>\glxtr@recordsee</code>	13	<code>\glxtrbibnumber</code>	341
<code>\glxtr@renewcommand</code>	339	<code>\glxtrbiborganization</code>	341
<code>\glxtr@resource</code>	141, 143	<code>\glxtrbibpages</code>	341
<code>\glxtr@s@newignoredglossary</code>	44	<code>\glxtrbibpublisher</code>	341
<code>\glxtr@s@provideignoredglossary</code> ...	46	<code>\glxtrbibschool</code>	341
<code>\glxtr@saveentrycounter</code>	8, 10, 13, 91	<code>\glxtrbibseries</code>	341
<code>\glxtr@setaccessdisplay</code>	188	<code>\glxtrbibtitle</code>	341
<code>\glxtr@setbookindexmark</code>	424	<code>\glxtrbibtype</code>	341
<code>\glxtr@setup@record</code>	15, 16, 29, 30	<code>\glxtrbibvolume</code>	341
<code>\glxtr@shortcutsval</code>	143	<code>\glxtrbookindexatendgroup</code>	422
<code>\glxtr@texencoding</code>	143	<code>\glxtrbookindexatsubendgroup</code>	422
<code>\glxtr@undefaction@nr</code>	7	<code>\glxtrbookindexatsubsubendgroup</code> ..	422
<code>\glxtr@undefaction@val</code>	7	<code>\glxtrbookindexbetween</code>	422
<code>\glxtr@usesee</code>	49	<code>\glxtrbookindexbookmark</code>	423
<code>\glxtr@warnonexistsordo</code>	7, 15, 16, 48	<code>\glxtrbookindexcols</code>	421
<code>\glxtr@writefields</code>	141	<code>\glxtrbookindexcolspread</code>	421
<code>\glxtrabbreviationfont</code>	67	<code>\glxtrbookindexfirstmarkfmt</code>	424
<code>\glxtrabbrvfootnote</code>	230–232, 253–255, 267–269, 288–290	<code>\glxtrbookindexformatheader</code>	423
<code>\glxtrabbrvpluralsuffix</code>	143, 208, 226, 228, 231, 233, 234, 236, 239, 242, 257, 270, 293, 302, 305, 308, 313, 316	<code>\glxtrbookindexgroupskip</code>	423
<code>\glxtrabbrvtype</code>	20, 206	<code>\glxtrbookindexlastmarkfmt</code>	424
<code>\glxtractivatenopost</code>	127	<code>\glxtrbookindexmulticolenv</code>	421
		<code>\glxtrbookindexname</code>	419, 422
		<code>\glxtrbookindexparentchildsep</code>	419, 421

<code>\glxtrbookindexparentschildsep</code> .	<code>\glxtrfmtdisplay</code>	33
..... 421, 422	<code>\glxtrfmtexternalnameref</code>	343
<code>\glxtrbookindexprelocation</code> ...	<code>\GlsXtrFmtField</code>	33, 34
419, 422	<code>\glxtrfmtinternalnameref</code>	342, 343
<code>\glxtrbookindexsubbetween</code>	<code>\glxtrfootnotename</code>	
422 230, 232, 253, 254, 267, 269, 288, 290	
<code>\glxtrbookindexsubname</code>	<code>\GlsXtrForeignTextField</code>	39
423	<code>\GlsXtrFormatLocationList</code> 64, 66, 408–410	
<code>\glxtrbookindexsubprelocation</code>	<code>\GLSxtrfull</code>	21, 329, 330
423	<code>\Glsxtrfull</code>	20, 21, 330
<code>\glxtrbookindexsubsubbetween</code>	<code>\glxtrfull</code>	20, 21, 329, 330
422	<code>\Glsxtrfullformat</code>	
<code>\glxtrbookindexthepage</code> 207, 221, 223, 224, 226, 229, 231, 233,	
424	235, 237, 240, 243, 245, 247, 248, 251,	
<code>\glxtrcat</code>	252, 254, 255, 257, 259, 261, 263, 265,	
60, 61	266, 268, 270, 271, 273, 275, 277, 279,	
<code>\glxtrchecknohyperfirst</code>	280, 282, 284, 286, 288, 289, 291, 294,	
76–78	295, 297, 300, 303, 306, 309, 311, 314, 316	
<code>\glxtrcombingdiacriticIIrules</code> .	<code>\glxtrfullformat</code>	
354 207, 221, 223, 224, 226, 228,	
<code>\glxtrcombingdiacriticIIrules</code> ..	231, 233, 235, 237, 240, 243, 245, 247,	
354	248, 251–253, 255, 257, 259, 261, 262,	
<code>\glxtrcombingdiacriticIVrules</code> ..	265–267, 269, 271, 273, 275, 277, 279,	
354	280, 282, 284, 286, 288, 289, 291, 294,	
<code>\glxtrComputeTreeIndent</code>	295, 297, 300, 303, 306, 309, 311, 314, 316	
411, 412	<code>\GLSxtrfullpl</code>	21, 330, 331
<code>\glxtrComputeTreeSubIndent</code>	<code>\Glsxtrfullpl</code>	20, 21, 331
412	<code>\glxtrfullpl</code>	20, 21, 330
<code>\glxtrcounterprefix</code>	<code>\Glsxtrfullplformat</code>	
135	. 207, 221, 223, 224, 226, 229, 231, 233,	
<code>\glxtrcurrencyrules</code>	235, 237, 240, 243, 245, 247, 248, 251,	
356	252, 254, 255, 257, 259, 261, 263, 265,	
<code>\Glsxtrdefaultsubsequentfmt</code> ...	266, 268, 270, 272, 273, 275, 277, 279,	
222, 223	280, 283, 284, 286, 288, 289, 291, 294,	
<code>\glxtrdefaultsubsequentfmt</code> ...	295, 297, 300, 303, 306, 309, 311, 314, 317	
221, 223	<code>\glxtrfullplformat</code>	
<code>\Glsxtrdefaultsubsequentplfmt</code> 221, 223, 224, 226, 228,	
222, 223	231, 233, 235, 237, 240, 243, 245, 247,	
<code>\glxtrdefaultsubsequentplfmt</code> .	248, 251–253, 255, 257, 259, 261, 262,	
222, 223	265–267, 269, 271, 273, 275, 277, 279,	
<code>\GlsXtrDefineAbbreviationShortcuts</code>	280, 282, 284, 286, 288, 289, 291, 294,	
..... 22, 23	295, 297, 300, 303, 306, 309, 311, 314, 316	
<code>\GlsXtrDefineAcShortcuts</code>	<code>\GLSxtrfullpl</code>	21, 330, 331
23	<code>\Glsxtrfullpl</code>	20, 21, 331
<code>\GlsXtrDefineOtherShortcuts</code>	<code>\glxtrfullpl</code>	20, 21, 330
23	<code>\Glsxtrfullplformat</code>	
<code>\glxtrdetoklocation</code> 207, 221, 223, 224, 226, 229, 231, 233,	
155	235, 237, 240, 243, 245, 247, 248, 251,	
<code>\glxtrdiscardperiod</code>	252, 254, 255, 258, 259, 261, 263, 265,	
197	267, 268, 270, 272, 273, 275, 277, 279,	
<code>\glxtrdisplayendloc</code>	280, 283, 284, 286, 288, 289, 291, 294,	
134	295, 297, 300, 303, 306, 309, 311, 314, 317	
<code>\glxtrdisplayendlochook</code>	<code>\glxtrfullplformat</code>	
134 221, 223, 224, 226, 228, 231,	
<code>\glxtrdisplaysingleloc</code>	233, 235, 237, 240, 243, 245, 247, 248,	
134	251–253, 255, 257, 259, 261, 262, 265,	
<code>\glxtrdisplaystartloc</code>	266, 268, 269, 271, 273, 275, 277, 279,	
134	280, 282, 284, 286, 288, 289, 291, 294,	
<code>\glxtrdoautoindexname</code>	295, 297, 300, 303, 306, 309, 311, 314, 316	
92, 187	<code>\glxtrfullsep</code>	206, 207,
<code>\glxtrdopostpunc</code>	226–229, 231–238, 240, 242–250, 252,	
232, 255, 269, 290	254, 256–266, 268, 270–278, 280–282,	
<code>\glxtrdowrglossaryhook</code>	284–287, 290–292, 302–307, 310–313, 317	
92	<code>\glxtrngenabbrvfmt</code>	67
<code>\GlsXtrDualField</code>	<code>\glxtrgeneralpuncIIrules</code>	356
340	<code>\glxtrgeneralpuncIrules</code>	356
<code>\glxtremsuffix</code>		
271, 273, 275,		
276, 278, 279, 281, 283, 285, 287, 289, 291		
<code>\GlsXtrEnableEntryCounting</code>		
119		
<code>\GlsXtrEnableEntryUnitCounting</code>		
108		
<code>\GlsXtrEnableOnTheFly</code>		
60, 62		
<code>\glxtrendfor</code>		
35		
<code>\glxtrfieldlistgadd</code>		
144		
<code>\glxtrfieldtitlecase</code>		
182–185, 188		
<code>\glxtrfieldtitlecasecs</code>		
181		
<code>\glxtrfieldxifinlist</code>		
149		
<code>\glxtrfirstscfont</code>		
242		
<code>\glxtrfirstsmfont</code>		
256		
<code>\GlsXtrFmtDefaultOptions</code>		
33		

<code>\glstrgetgrouptitle</code>	133, 423	<code>\glstrtrinithyperoutside</code>	72
<code>\glstrgroupfield</code>	151	<code>\glstrtrinitwrgloss</code>	72, 156
<code>\Glsxtrheadfirst</code>	321	<code>\glstrtrinitwrglossbeforefalse</code> ...	69, 70
<code>\glstrheadfirst</code>	321	<code>\glstrtrinitwrglossbeforetrue</code> ...	69, 70
<code>\Glsxtrheadfirstplural</code>	321	<code>\Glsxtrinlinefullformat</code> 207, 209, 223,	
<code>\glstrheadfirstplural</code>	321	224, 232, 233, 235, 237, 238, 240, 247–	
<code>\Glsxtrheadfull</code>	321	250, 252, 254, 256, 261–263, 265, 266,	
<code>\glstrheadfull</code>	321	268, 270, 278, 280–282, 284, 286, 287,	
<code>\Glsxtrheadfullpl</code>	321	290, 291, 295, 298, 306, 309, 314, 317, 337	
<code>\glstrheadfullpl</code>	321	<code>\glstrinlinefullformat</code>	
<code>\Glsxtrheadlong</code>	321	207, 209, 210, 223, 224, 231,	
<code>\glstrheadlong</code>	321	233, 234, 236, 238, 240, 246, 248–250,	
<code>\Glsxtrheadlongpl</code>	321	252, 254, 256, 260, 262–264, 266, 268,	
<code>\glstrheadlongpl</code>	321	270, 278, 279, 281, 282, 284, 285, 287,	
<code>\Glsxtrheadname</code>	321	289, 291, 295, 298, 305, 309, 314, 317, 336	
<code>\glstrheadname</code>	145, 321	<code>\Glsxtrinlinefullplformat</code> 208, 211, 223,	
<code>\Glsxtrheadplural</code>	321	224, 232, 233, 235, 237, 238, 240, 247–	
<code>\glstrheadplural</code>	321	250, 252, 254, 256, 261–263, 265, 266,	
<code>\Glsxtrheadshort</code>	321	268, 270, 278, 280–282, 284, 286, 287,	
<code>\glstrheadshort</code>	321	290, 292, 295, 298, 306, 309, 314, 317, 337	
<code>\Glsxtrheadshortpl</code>	321	<code>\glstrinlinefullplformat</code>	
<code>\glstrheadshortpl</code>	321	207, 210, 211, 223,	
<code>\Glsxtrheadtext</code>	321	224, 231, 233, 234, 236, 238, 240, 246,	
<code>\glstrheadtext</code>	321	248–250, 252, 254, 256, 261–264, 266,	
<code>\glstrhiernamesep</code>	50, 51	268, 270, 278, 280–282, 284, 285, 287,	
<code>\glstrhyperlink</code>	25, 26, 94	290, 291, 295, 298, 305, 309, 314, 317, 337	
<code>\glstrhyphensuffix</code>	303, 311	<code>\glstrinsertinsidefalse</code>	225
<code>\glstridentifyglslike</code>	153	<code>\GlsXtrInternalLocationHyperlink</code> 26, 135	
<code>\glstrifcounttrigger</code>	110–112	<code>\glstrLatinA</code>	358–363
<code>\glstrifcustomdiscardperiod</code>	198	<code>\glstrLatinAELigature</code>	360, 362, 363
<code>\glstrifemptyglossary</code>	133, 139, 147	<code>\glstrLatinE</code>	358–363
<code>\GlsXtrIfFieldCmpNum</code>	36	<code>\glstrLatinEszettSs</code>	359–361, 363
<code>\GlsXtrIfFieldEqNum</code>	146	<code>\glstrLatinEszettSz</code>	360, 362
<code>\GlsXtrIfFieldNonZero</code>	339	<code>\glstrLatinEth</code>	359–362
<code>\glstrifhasfield</code> ...	39, 50, 51, 91, 340, 419	<code>\glstrLatinH</code>	358–363
<code>\glstrifhyphenstart</code>		<code>\glstrLatinI</code>	358–363
.....	302, 304, 307, 310, 312, 313	<code>\glstrLatinInsularG</code>	362
<code>\glstrifindexing</code>	92	<code>\glstrLatinK</code>	358–363
<code>\glstrifinmark</code>	74, 101–104, 320, 321	<code>\glstrLatinL</code>	358–363
<code>\glstrifnextpunc</code>	200, 201	<code>\glstrLatinM</code>	358–363
<code>\glstrifperiod</code>	198–200	<code>\glstrLatinN</code>	358–363
<code>\glstrifrecordtrigger</code>	157–159	<code>\glstrLatinO</code>	358–363
<code>\glstrifwasfirstuse</code>		<code>\glstrLatinOELigature</code>	360, 362, 363
.....	75–78, 83–87, 89, 122, 198,	<code>\glstrLatinP</code>	358–363
.....	199, 209, 212–219, 232, 233, 255, 269,	<code>\glstrLatinS</code>	358–363
.....	290, 291, 294, 296–298, 308, 310, 313, 315	<code>\glstrLatinT</code>	358–363
<code>\glstrinclinckcounter</code>	161	<code>\glstrLatinThorn</code>	362
<code>\glstrindexaliased</code>	91	<code>\glstrLatinX</code>	358–363
<code>\glstrindexseealso</code>	54, 55	<code>\GlsXtrLocationField</code>	151

<code>\glxtrlocationhyperlink</code>	135	<code>\glxtrnewnumber</code>	22
<code>\glxtrlocrangefmt</code>	134	<code>\glxtrnewsymbol</code>	22
<code>\GLSxtrlong</code>	21, 328	<code>\glxtrNoGlossaryWarning</code>	15, 24, 136
<code>\Glsxtrlong</code>	20, 21, 329	<code>\GlsXtrNoGlsWarningAutoMake</code>	140
<code>\glxtrlong</code>	20, 21, 328	<code>\GlsXtrNoGlsWarningBuildInfo</code>	140
<code>\glxtrlonghyphen</code>	309	<code>\GlsXtrNoGlsWarningCheckFile</code>	140
<code>\glxtrlonghyphennoshort</code>	305, 306	<code>\GlsXtrNoGlsWarningEmptyMain</code> ..	139, 140
<code>\glxtrlonghyphenshort</code>	303	<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	140
<code>\glxtrlongnoshortdescname</code> .	239, 286, 305	<code>\GlsXtrNoGlsWarningEmptyStart</code>	139
<code>\glxtrlongnoshortname</code>		<code>\GlsXtrNoGlsWarningHead</code>	139
.....	241, 249, 264, 281, 283, 306	<code>\GlsXtrNoGlsWarningMisMatch</code>	140
<code>\GLSxtrlongpl</code>	21, 328, 329	<code>\GlsXtrNoGlsWarningNoOut</code>	140
<code>\Glsxtrlongpl</code>	20, 21, 329	<code>\GlsXtrNoGlsWarningTail</code>	140, 141
<code>\glxtrlongpl</code>	20, 21, 328	<code>\glxtrnopostpunc</code>	127
<code>\glxtrlongshortdescname</code>		<code>\glxtronlydescname</code>	317
.....	227, 244, 258, 272, 274, 304, 309	<code>\glxtronlydescsort</code>	317
<code>\glxtrlongshortdescsort</code>		<code>\glxtronlyname</code>	316
....	227, 244, 258, 272, 274, 299, 304, 309	<code>\glxtronlysuffix</code>	316
<code>\glxtrlongshortname</code>		<code>\glxtrorg@ifKV@glslink@hyper</code> ...	67, 68
	226, 242, 257, 271, 272, 293, 294, 303, 308	<code>\glxtrorglong</code>	204, 227, 305
<code>\glxtrlongshortuserdescname</code> ..	296, 299	<code>\glxtrorgshort</code>	204, 227
<code>\glxtrmarkhook</code>	319	<code>\GLSxtrp</code>	101
<code>\glxtrMathItalicAlpha</code> .	367, 368, 371, 372	<code>\Glsxtrp</code>	100
<code>\glxtrMathItalicBeta</code> ..	367, 368, 371, 372	<code>\glxtrp</code>	100, 102
<code>\glxtrMathItalicChi</code>	368, 372, 373	<code>\glxtrparen</code>	198,
<code>\glxtrMathItalicDelta</code> .	367, 368, 371, 372		199, 206, 207, 226–229, 231–238, 240,
<code>\glxtrMathItalicEpsilon</code> ..	367, 368, 371, 372		242–252, 254, 256–266, 268, 270–282,
<code>\glxtrMathItalicEta</code> ...	367, 368, 371, 372		284–287, 290–292, 302–307, 310–313, 317
<code>\glxtrMathItalicGamma</code> .	367, 368, 371, 372	<code>\Glsxtrpl</code>	62
<code>\glxtrMathItalicIota</code> ..	367, 368, 371, 373	<code>\glxtrpl</code>	62
<code>\glxtrMathItalicKappa</code>	368, 372, 373	<code>\glxtrpostdescription</code> .	128, 179, 196, 395
<code>\glxtrMathItalicLambda</code>	368, 372, 373	<code>\glxtrposthyphenlong</code>	313, 315
<code>\glxtrMathItalicMu</code>	368, 372, 373	<code>\glxtrposthyphenshort</code>	308, 310
<code>\glxtrMathItalicNu</code>	368, 372, 373	<code>\glxtrposthyphensubsequent</code>	
<code>\glxtrMathItalicOmega</code>	368, 372, 373		308, 310, 313, 315
<code>\glxtrMathItalicOmicron</code> ...	368, 372, 373	<code>\glxtrpostlink</code>	198
<code>\glxtrMathItalicPhi</code>	368, 372, 373	<code>\glxtrpostlinkendsentence</code>	198
<code>\glxtrMathItalicPi</code>	368, 372, 373	<code>\glxtrpostlinkhook</code>	197
<code>\glxtrMathItalicPsi</code>	368, 372, 373	<code>\glxtrpostlocalreset</code>	107, 109, 117
<code>\glxtrMathItalicRho</code>	368, 372, 373	<code>\glxtrpostlocalunset</code>	106, 109, 117
<code>\glxtrMathItalicSigma</code>	368, 372, 373	<code>\glxtrpostlongdescription</code>	43
<code>\glxtrMathItalicTau</code>	368, 372, 373	<code>\glxtrpostnamehook</code>	184–186, 189
<code>\glxtrMathItalicTheta</code> .	367, 368, 371, 372	<code>\GlsXtrPostNewAbbreviation</code>	206,
<code>\glxtrMathItalicUpsilon</code> ...	368, 372, 373		223, 224, 226–230, 232, 234–239, 241–
<code>\glxtrMathItalicXi</code>	368, 372, 373		244, 246, 247, 250, 253, 255, 257–260,
<code>\glxtrMathItalicZeta</code> ..	367, 368, 371, 372		262, 264, 267, 269, 271–274, 276–279,
<code>\glxtrmultisupplocation</code>	341		281, 283, 285, 287–290, 293, 294, 296–
<code>\glxtrnamereflink</code>	342		301, 303–306, 308, 310–313, 315, 316, 318
<code>\glxtrnewabbrevpresetkeyhook</code>	205	<code>\glxtrpostreset</code>	106, 109, 117

<code>\glxtrpostunset</code>	105, 109, 117	<code>\Glsxtrsubsequentfmt</code>	220, 223, 239, 250, 252,
<code>\glxtrprelocation</code>	385, 387, 389, 391, 393, 396, 419		264, 266, 282, 283, 285, 287, 305, 308, 314
<code>\glxtrprotectlinks</code>	94–96	<code>\glxtrsubsequentfmt</code>	220, 223, 239, 250, 251,
<code>\GlsXtrRecordCounter</code>	13		264, 266, 282, 283, 285, 287, 305, 308, 314
<code>\glxtrrecordtriggervalue</code>	156	<code>\Glsxtrsubsequentplfmt</code>	220, 223, 240, 250, 252,
<code>\GlsXtrRecordWarning</code>	141		264, 266, 282, 283, 285, 287, 305, 309, 314
<code>\glxtrregularfont</code>	67, 75, 76	<code>\glxtrsubsequentplfmt</code>	220, 223, 239, 250, 251,
<code>\glxtrresourcecount</code>	142		264, 266, 282, 283, 285, 287, 305, 308, 314
<code>\glxtrresourcefile</code>	142	<code>\glxtrsupplocationurl</code> .	135, 136, 341, 343
<code>\glxtrresourceinit</code>	141	<code>\glxtrtagfont</code>	196
<code>\glxtrrestoremarkhook</code>	319	<code>\Glsxtrtitlefirst</code>	320, 321, 334
<code>\glxtrrestorepostpunc</code>	128	<code>\glxtrtitlefirst</code>	320, 321, 334
<code>\glxtrscfont</code>	242	<code>\Glsxtrtitlefirstplural</code>	320–322, 335
<code>\glxtrscsuffix</code>	243, 245, 246, 248, 250, 251, 253, 255	<code>\glxtrtitlefirstplural</code>	320, 321, 335
<code>\GlsXtrSetActualChar</code>	193	<code>\Glsxtrtitlefull</code>	320–322, 337
<code>\glxtrsetaliasnoindex</code>	15, 16, 91	<code>\glxtrtitlefull</code>	320–322, 336
<code>\GlsXtrSetEncapChar</code>	193	<code>\Glsxtrtitlefullpl</code>	320–322, 337
<code>\GlsXtrSetEscChar</code>	193	<code>\glxtrtitlefullpl</code>	320–322, 337
<code>\glxtrsetfieldifexists</code>	38	<code>\Glsxtrtitlelong</code>	320–322, 335, 336
<code>\GlsXtrSetLevelChar</code>	193	<code>\glxtrtitlelong</code>	320–322, 335
<code>\glxtrsetpopts</code>	99	<code>\Glsxtrtitlelongpl</code>	320–322, 336
<code>\glxtrsetupfulldefs</code>	209–211, 233, 255, 269, 291	<code>\glxtrtitlelongpl</code>	320–322, 336
<code>\GLSxtrshort</code>	21, 103, 104, 322, 323	<code>\Glsxtrtitlename</code>	320, 321, 333
<code>\Glsxtrshort</code>	20, 21, 323	<code>\glxtrtitlename</code>	320, 321, 332
<code>\glxtrshort</code>	20, 21, 322	<code>\glxtrtitleorpdforheading</code>	27, 145, 146, 320, 321
<code>\glxtrshortdescname</code> ...	236, 247, 262, 279	<code>\Glsxtrtitleplural</code>	320, 321, 334
<code>\glxtrshorthyphen</code>	314	<code>\glxtrtitleplural</code>	320, 321, 333, 334
<code>\glxtrshorthyphenlong</code>	311	<code>\Glsxtrtitleshort</code>	320, 321, 332
<code>\glxtrshortlongdescname</code>	229, 245, 259, 275, 277, 312, 315	<code>\glxtrtitleshort</code>	320, 321, 331
<code>\glxtrshortlongdescsort</code>	229, 245, 260, 275, 277, 301, 312, 315	<code>\Glsxtrtitleshortpl</code>	320, 321, 332
<code>\glxtrshortlongname</code>	228, 244, 258, 274, 276, 296, 300, 311, 313	<code>\glxtrtitleshortpl</code>	320, 321, 331, 332
<code>\glxtrshortlonguserdescname</code> ..	298, 301	<code>\Glsxtrtitletext</code>	320, 321, 333
<code>\glxtrshortnolongname</code> .	234, 246, 260, 278	<code>\glxtrtitletext</code>	320, 321, 333
<code>\GLSxtrshorttpl</code>	21, 322, 323	<code>\GlsXtrTotalRecordCount</code>	155
<code>\Glsxtrshorttpl</code>	20, 21, 323	<code>\glxtrtreetopindent</code>	402, 410
<code>\glxtrshorttpl</code>	20, 21, 322, 323	<code>\glxtrundefaction</code> 7, 15, 16, 31, 44, 45, 47, 48	
<code>\glxtrsmfont</code>	256	<code>\glxtrundeftag</code>	30, 130, 131
<code>\glxtrsmsuffix</code>	257, 259, 260, 262, 264, 265, 267, 269	<code>\GlsXtrUnknownDialectWarning</code>	41
<code>\GlsXtrStandaloneEntryName</code>	145	<code>\glxtrunsrtdo</code>	149
<code>\GlsXtrStandaloneEntryOther</code>	146	<code>\glxtrUpAlpha</code>	366, 367, 371, 372
<code>\GlsXtrStandaloneGlossaryType</code> .	145, 146	<code>\glxtrUpBeta</code>	366, 367, 371, 372
<code>\GlsXtrStandaloneSubEntryItem</code> .	145, 146	<code>\glxtrUpChi</code>	367, 372, 373
		<code>\glxtrUpDelta</code>	366, 367, 371, 372
		<code>\glxtrUpDigamma</code>	366, 367, 371

<code>\glxtrUpEpsilon</code>	366, 367, 371, 372	<code>hyperref</code> package	95, 189, 318, 331, 342
<code>\glxtrUpEta</code>	366, 367, 371, 372		
<code>\glxtrUpGamma</code>	366, 367, 371, 372		
<code>\glxtrUpIota</code>	366, 367, 371, 373		
<code>\glxtrUpKappa</code>	366, 367, 372, 373		
<code>\glxtrUpLambda</code>	366, 367, 372, 373		
<code>\glxtrUpMu</code>	366, 367, 372, 373		
<code>\glxtrUpNu</code>	366, 367, 372, 373		
<code>\glxtrUpOmega</code>	367, 372, 373		
<code>\glxtrUpOmicron</code>	366, 367, 372, 373		
<code>\glxtrUpPhi</code>	367, 372, 373		
<code>\glxtrUpPi</code>	366, 367, 372, 373		
<code>\glxtrUpPsi</code>	367, 372, 373		
<code>\glxtrUpRho</code>	366, 367, 372, 373		
<code>\glxtrUpSigma</code>	366, 367, 372, 373		
<code>\glxtrUpTau</code>	367, 372, 373		
<code>\glxtrUpTheta</code>	366, 367, 371, 373		
<code>\glxtrUpUpsilon</code>	367, 372, 373		
<code>\glxtrUpXi</code>	366, 367, 372, 373		
<code>\glxtrUpZeta</code>	366, 367, 371, 372		
<code>\GlsXtrUseAbbrStyleFmts</code>	227, 230, 236, 238, 239, 241, 242, 244, 246, 249, 258, 260, 263, 272, 274, 276, 278, 280, 285, 288, 296, 299, 301, 304, 305, 307, 310, 312, 315, 318		
<code>\GlsXtrUseAbbrStyleSetup</code>	235, 237, 238, 241, 242, 249, 251, 263, 265, 280, 284, 285, 288		
<code>\glxtruserfield</code>	292		
<code>\glxtruserparen</code>	293–301		
<code>\glxtrusersuffix</code>	293, 295, 297, 300		
<code>\glxtruseseealsoformat</code>	52		
<code>\glxtruseseeeformat</code>	49		
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	202, 224		
<code>\GlsXtrWarning</code>	60, 61		
<code>\glxtrword</code>	203		
<code>\glxtrwordsep</code>	203, 302, 304, 307, 310, 312, 313		
<code>\glxtrwrglossmark</code>	27		
H			
<code>\hangindent</code> ..	398, 400, 401, 411–413, 417, 418		
<code>\hbox</code>	384		
<code>\hfill</code>	384		
<code>\href</code>	94		
<code>\hsize</code>	63, 64		
<code>\hss</code>	384		
<code>\hyperlink</code>	16, 95, 342		
<code>\hyperpage</code>	189		
<code>\hyperref</code>	94, 135, 339, 340, 344		
I			
<code>\if</code>	59		
<code>\if@display</code>	12		
<code>\if@glxtr@autoseeindex</code>	28, 49, 54		
<code>\if@glxtr@equations</code>	9, 72		
<code>\if@glxtr@floats</code>	18		
<code>\if@glxtr@format@override</code>	190		
<code>\if@glxtr@docdefrestricted</code>	58		
<code>\if@glxtr@indexcrossrefs</code>	17, 55		
<code>\ifblank</code>	32, 60, 61, 123		
<code>\ifbool</code>	31, 43		
<code>\ifcase</code> ..	7, 15, 22, 24, 26, 58, 70, 128, 396, 422		
<code>\ifcsdef</code>	31, 42, 44–47, 70, 73, 87, 88, 100–104, 114, 126, 132, 133, 146, 150, 153–155, 160, 181– 184, 186–188, 198, 202, 219, 223, 387–394		
<code>\ifcsstring</code>	31, 177, 222		
<code>\ifcsundef</code>	37, 40, 42, 44–47, 58, 63, 65, 96, 108, 114–118, 132, 136, 150, 161, 177, 222, 224, 225, 384, 402, 403, 411, 424		
<code>\ifcsvoid</code>	55, 176		
<code>\ifdef</code> ..	15, 22, 28, 34, 37, 39, 41, 48, 52, 53, 56, 63, 64, 90, 93, 95, 101–103, 126, 130, 131, 141, 143, 153, 179, 180, 193, 196, 292, 320, 331–337, 339, 344–347, 382, 384–387, 395–401, 412–418, 420, 423, 424		
<code>\ifdefempty</code>	7–9, 36, 40, 41, 44–46, 49, 52, 73, 75, 108, 119, 120, 123, 126, 134, 148, 151, 155, 156, 170–172, 195, 203, 219, 347, 421		
<code>\ifdefequal</code>	39, 57, 93, 140, 151, 152, 187		
<code>\ifdefstring</code>	6, 26, 39, 190, 195, 420		
<code>\ifdefvoid</code>	49, 53–56, 94, 114, 131, 135, 151, 152, 343		
<code>\ifdim</code>	63, 64, 122, 402–410, 427, 428		
<code>\IfFileExists</code> ..	24, 136, 140, 141, 144, 382, 383		
<code>\ifglossaryexists</code>	48		
<code>\ifglssacronym</code>	20, 140		
<code>\ifglssacrshortcuts</code>	22		
<code>\ifglssautomake</code>	126, 140, 144		
<code>\ifglssentrycounter</code>	41, 42		
<code>\ifglssentryexists</code>	9, 31, 47, 48, 60, 61, 64, 75, 151, 177, 196, 197, 348		
<code>\ifglssfieldeq</code>	175		
<code>\ifglsshasfield</code>	33, 34, 292		
<code>\ifglsshaslong</code>	113, 159, 160		
<code>\ifglsshasparent</code> ..	145, 146, 148, 151, 404–406		

<code>\ifglshasshort</code>	50, 51, 67, 75	<code>\istfilename</code>	123
<code>\ifglshassymbol</code>	199, 398, 399, 401	<code>\item</code>	138, 139, 384–387, 396, 397, 413, 414
<code>\ifglindexonlyfirst</code>	92		
<code>\ifGlsLongExtraUseTabular</code>	429, 430, 432, 433, 435, 436, 438, 440, 441, 443, 444, 446	J	
<code>\ifglsnogroupskip</code>	385, 387–395, 397, 398, 400, 412, 420, 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447	<code>\jobname</code>	57, 136, 138–140, 142, 144
<code>\ifglsnonumberlist</code>	66	K	
<code>\ifglsnopostdot</code>	18, 19, 128	<code>\key@ifundefined</code> .	13, 14, 31, 32, 87, 148, 150
<code>\ifglssanitizesort</code>	125	<code>\KV@glslink@hyperfalse</code> ..	76, 89, 90, 95, 96
<code>\ifglssubentrycounter</code>	41, 42	<code>\KV@glslink@hypertrue</code>	95
<code>\ifglssused</code>	55, 56, 89, 92, 110, 118, 122, 219, 404, 405, 407, 408, 410	<code>\KV@glslink@noindexfalse</code>	75, 90
<code>\ifglxsindy</code>	136, 138	<code>\KV@glslink@noindextrue</code>	90, 96
<code>\ifglsxtr@hyperoutside</code>	73, 74	L	
<code>\ifglsxtrinitwrglossbefore</code>	70, 73, 74, 156	<code>\L</code>	363, 366
<code>\ifglsxtrininsertinside</code>	212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261– 273, 275, 277–292, 294, 295, 297, 298, 300, 302, 304, 307–310, 313, 314, 316, 317	<code>\l</code>	366
<code>\ifHy@hyperindex</code>	189	<code>\label</code>	26
<code>\ifinlist</code>	106	<code>\LaTeX</code>	138, 139
<code>\ifinlistcs</code>	35, 58	<code>\leaders</code>	384
<code>\ifinner</code>	27	<code>\leavevmode</code>	44, 72
<code>\ifKV@glslink@hyper</code>	67, 68, 72–74	<code>\let</code> ..	5, 7–11, 13, 15, 16, 18, 20–22, 27–29, 33, 35, 39, 40, 43, 57, 59, 62, 63, 65–69, 72–93, 95, 96, 100, 105, 106, 108, 109, 117–124, 126–133, 141–144, 148, 149, 151, 152, 156, 161, 171, 182, 183, 185– 192, 195, 196, 201–205, 209–219, 221– 223, 233, 255, 269, 291, 318–322, 339, 342, 343, 382, 396, 401, 403, 414, 421–424
<code>\ifKV@glslink@local</code>	69, 156	<code>\letabbreviationstyle</code> ..	232, 234, 235, 237, 240, 241, 247, 249, 261, 263, 279, 280
<code>\ifKV@glslink@noindex</code>	8, 9, 13, 33, 91	<code>\letcs</code>	31, 36, 49, 52, 56, 70, 73, 87, 130–132, 150, 151, 181–189, 424
<code>\ifmmode</code>	12, 27	<code>\levelchar</code>	193
<code>\ifnum</code>	16, 17, 36, 57, 110, 118, 119, 131, 142, 156, 344–346, 398, 400, 411, 412	<code>\linewidth</code>	428
<code>\ifst@rred</code>	12	<code>\listadd</code>	114
<code>\ifstrempty</code>	146, 153, 161, 342, 344–348	<code>\listbreak</code>	195
<code>\ifstrequal</code>	19, 24, 71, 342	<code>\listcsadd</code>	34
<code>\ifthenelse</code>	139, 140, 198	<code>\listcseadd</code>	34, 115
<code>\IfTrackedDialectHasMapping</code>	40	<code>\listcsgadd</code>	34, 58
<code>\IfTrackedLanguageFileExists</code>	338	<code>\listcsxadd</code>	34, 115
<code>\ifundef</code>	16, 26, 36, 40, 123, 195, 196, 342, 381, 427	<code>\listxadd</code>	105, 106
<code>\ifx</code> ..	8–11, 14, 29, 30, 52, 63, 64, 73, 123, 126, 127, 129, 134, 135, 141, 143, 144, 190, 191, 193, 201, 203, 205, 301, 302, 343, 418	<code>\loadglsentries</code>	59, 137
<code>\immediate</code> ...	57, 109, 110, 118, 136, 137, 144	<code>\long</code>	43
<code>\index</code>	190	M	
<code>\indexspace</code> .	385, 397–401, 412–418, 420, 423	<code>\m@ne</code>	339
<code>\input</code>	338	<code>\MakeAcronymsAbbreviations</code>	121
<code>\inputencodingname</code>	143	<code>\makeatletter</code>	57, 136, 141, 192
<code>\InputIfFileExists</code>	57	<code>\makeatother</code>	192
		<code>\makebox</code>	384, 411, 412

<code>\makefirsttuc</code>	196	<code>\newcommand</code>	5–37, 39, 41–52, 54–56, 59–62, 64–67, 69–71, 74–76, 87– 93, 95, 96, 99, 101–108, 110, 112–116, 118, 119, 121, 122, 126–132, 134–142, 144–170, 172–181, 187–204, 206–219, 221–225, 227–230, 234, 236, 239, 241, 242, 256, 257, 270, 271, 292, 293, 296, 298, 302, 304, 307, 310, 312, 313, 315– 317, 319–344, 346–349, 352–381, 383, 385, 395–399, 401–403, 410, 411, 419, 420, 423–430, 432–436, 438, 439, 441–446
<code>makeglossaries</code>	129	<code>\newcount</code>	142, 152
<code>\makeglossaries</code>	122, 137–140, 144	<code>\newcounter</code>	26, 160
<code>\makeglossary</code>	123	<code>\newentry</code>	22
<code>makeindex</code>	448	<code>\newglossary</code>	20, 123
<code>makeindex</code>	16, 122	<code>\newglossaryentry</code>
<code>\makenoidxglossaries</code>	138	22, 57, 59, 108, 116, 120, 179, 180, 206	
<code>\MakeTextUppercase</code>	321	<code>\newglossaryentry options</code>	
<code>\MakeUppercase</code>	320, 321	access	171
<code>\marginpar</code>	27	alias	17, 49, 53–56
<code>\markboth</code>	319	desc	166, 174
<code>\markright</code>	319	descplural	166, 167, 174
<code>\mathit</code>	351	first 94, 163, 164, 173, 225, 326, 327, 334, 448	
<code>\mathrm</code>	350–352	firstaccess	172
<code>\maxdimen</code>	63, 64	firstplural	164, 173, 225, 327, 334, 448
<code>\mbox</code>	386, 387, 411	group	150, 151
<code>\medskip</code>	139, 140, 150	loclist	34
<code>\MessageBreak</code>	58, 59, 62, 110, 119, 123, 125–127, 140, 141, 222	long	168, 175, 335
<code>mfirsttuc package</code>	195	longplural	169, 175, 336
<code>\mfirsttucMakeUppercase</code>	37, 76–87, 89, 97–99, 101, 103, 104, 112, 113, 121, 160, 162–169, 172–175, 184, 185, 189, 210, 211, 213, 215, 217, 219–221	name ... 49, 50, 162, 172, 190, 323, 324, 332	
<code>\mfu@checkword@arg</code>	195, 196	plural	163, 172, 173, 225, 325, 326, 333
<code>\mfu@checkword@do</code>	196	see	17, 28, 49, 53, 56, 59, 123
<code>\midrule</code>	427, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446	seealso	17, 49, 51, 53, 55, 56, 459
N		short	167, 174, 202
<code>\NeedsTeXFormat</code>	5, 339, 383, 419, 425	shortaccess	170–172
<code>\new@atom@glossaryentry</code>	57	shortaccessplural	170
<code>\new@command</code>	339	shortplural	168, 175, 202
<code>\new@glossaryentry</code>	59, 125	symbol	165, 173
<code>\new@ifnextchar</code>	33, 88, 89, 112, 113, 153, 157–159, 200, 208–219, 348, 349	symbolplural	165, 166, 173, 174
<code>\newabbr</code>	21	text 94, 162, 163, 172, 225, 227, 324, 325, 333	
<code>\newabbreviation</code>	21	textaccess	171
<code>\newabbreviationhook</code>	206	user2	41
<code>\newabbreviationstyle</code>	<code>\newglossarystyle</code> ... 421, 429, 430, 432, 433, 435, 436, 438, 439, 441, 443, 444, 446	
225, 227–230, 232, 234–239, 241–247, 249, 251, 253, 254, 257–261, 263–265, 267, 268, 271, 272, 274–281, 283–286, 288, 290, 293, 294, 296, 298, 299, 301, 303, 305, 306, 308, 309, 311, 313, 315–317		<code>\newif</code>	70, 189, 225, 429
<code>\newacronym</code>	120, 121	<code>\newlength</code>	401, 402
<code>\newacronymhook</code>	120	<code>\newnum</code>	22
<code>\newacronymstyle</code>	121, 122	<code>\newrobustcmd</code>	32, 34, 35, 38, 39, 52, 53, 57, 71, 75, 88, 89, 99–101,

112, 113, 127, 132, 145, 146, 149, 150,
153, 154, 157–159, 188, 195, 196, 208–
219, 301, 320, 322–331, 348–350, 403–410

\newsym 22

\newterm 179

\newtoks 202

\newwrite 57, 123

\nobreak 386, 387, 397, 399–401, 413–416

\NoCaseChange 101–104, 146, 322–331

\noexpand 11,
13, 24, 52, 54, 55, 57, 71, 120, 136, 137,
141, 148, 149, 151, 160, 191, 192, 206,
339, 348, 383, 422, 429, 431–438, 440–446

\nofiles 139

\noindent 140, 399–401, 414–416, 418

\nopagebreak 397, 399–401, 412–419, 423

\nopostdesc 44, 60, 61, 127, 179

\ns@GLSxtrfull 210

\ns@Glsxtrfull 209

\ns@glxtrfull 208

\ns@GLSxtrfullpl 211

\ns@Glsxtrfullpl 211

\ns@glxtrfullpl 210

\ns@GLSxtrlong 215

\ns@Glsxtrlong 214

\ns@glxtrlong 214

\ns@GLSxtrlongpl 219

\ns@Glsxtrlongpl 218

\ns@glxtrlongpl 217

\ns@GLSxtrshort 213

\ns@Glsxtrshort 212

\ns@glxtrshort 212

\ns@GLSxtrshortpl 217

\ns@Glsxtrshortpl 216

\ns@glxtrshortpl 215

\null 24

\number 57, 115, 116, 118, 141, 153

\numexpr 115, 118

O

\O 363, 366

\o 366

\openout 57

\or 7, 16, 22, 23, 26, 27, 59, 70, 396, 422

\org@glossaryentrynumbers 64, 127

\org@glossarytitle 126, 127

\org@ifKV@glslink@hyper 72, 74

P

\p@glS@hyp@opt 92

package options:

abbreviations 20

accsupp 23, 162

acronym 20

automake 126, 138, 144

 true 144

autoseeindex 28

 false 28

counter

 wrglossary 25

debug

 showtargets 95

docdef 16, 57, 58, 108, 116

 atom 57

 false 57, 58

 restricted 17

 true 56, 59

docdefs

 restricted 58

equations 9, 72

indexcounter 339

nonumberlist 64

nopostdot 18

 false 18

numbers 21

postdot 18

record 7,
14, 15, 57, 67, 122, 140, 141, 209–219, 457

 alsoindex 8, 10

 nameref 12, 29, 30, 141, 144

 only 8, 139

shortcuts 22

 ac 22

 all 22

 false 22

 none 22

 true 22

sort

 use 75

style 25

stylemods 25

symbols 21, 180

undefaction 47

 error 6

 warn 6

xindy 52, 53

\PackageError 6, 13, 24, 28,
57–59, 62, 63, 70, 88, 89, 91, 100, 101,

108, 109, 116, 118, 119, 121, 123, 125, 133, 144, 148, 150, 153, 198, 222–225, 384	\ProvidesFile 338
\PackageWarning 18	\ProvidesPackage 5, 339, 383, 419, 425
\PackageWarningNoLine 18	
\pagelistname ... 430, 433, 436, 439, 442, 445	Q
\pageref 25, 41, 42	\quotechar 193
\par 139, 140, 386, 387, 396, 398–401, 411–418, 420	R
\parindent .. 396, 398, 400–402, 411–418, 421	\raggedright 389, 390, 393, 394, 421, 426, 427
\parskip 396, 398, 400, 414–416, 421	\refstepcounter 26
\PassOptionsToPackage 5, 24	\relax 7, 15, 17, 21, 22, 24, 26, 28, 29, 33, 36, 57, 59, 62–65, 69, 70, 72, 73, 92, 100, 109, 110, 118, 123, 124, 126, 127, 130, 131, 134, 141–144, 151, 152, 156, 191–193, 196, 198, 203, 205, 301, 302, 339, 344–346, 396, 398, 400, 403–413, 417, 418, 421–424
\pdfbookmark 420	resize package 256
\preglossarypreamble 42	\renewcommand 6, 7, 15– 20, 22–29, 43–48, 50, 57–59, 62, 64–67, 69, 71, 87, 89–91, 93–95, 99, 106–109, 113, 116–129, 131–137, 149, 150, 155, 179, 181–186, 194–197, 207, 208, 223, 224, 226–301, 303–306, 308–319, 347, 381, 384–401, 411–418, 421–423, 429–447
\preto 90, 347	\renewenvironment 385, 387–394, 396, 398, 400, 411, 414–416, 418, 421, 429, 431, 432, 434–438, 440, 441, 443–446
\print@noop@unsrtglossaryunit ... 13, 16	\renewglossarystyle 384–394, 396–400, 411–418
\print@op@unsrtglossaryunit 15, 16	\renewrobustcmd 74, 95
\printabbreviations 20	\RequireGlossariesExtraLang ... 338, 382
\printglossaries 124, 138	\RequirePackage 5, 15, 23, 24, 27, 383, 419, 425
\printglossary 20, 124, 138–140	\reserved@a 200, 201
\printglossary options	\reserved@b 200, 201
nonumberlist 66	\reserved@d 201
type 126	\RestoreAcronyms 121
\printnoidxglossaries 138	\rGLS 155
\printnoidxglossary 124, 138	\rGls 155
\printnumbers 22, 180	\rgls 155
\printsymbols 22, 180	\rGLSformat 158
\printunsrtglossary 139, 141, 147	\rGlsformat 158
\printunsrtglossaryentryprocesshook 148, 149	\rglsformat 157, 160
\printunsrtglossaryhandler 149	\rGLSpl 155
\printunsrtglossarypredoglossary .. 148	\rGlspl 155
\printunsrtglossaryskipentry 148	\rglspl 155
\printunsrtglossaryunit 15, 16, 150	\rGLSplformat 159
\printunsrtglossaryunitsetup 149	\rGlsplformat 158
\ProcessOptions 384	\rglsplformat 157, 160
\ProcessOptionsX 27	\romannumeral 402, 403, 411
\protect 101–104, 172–175, 203, 206, 207, 225–232, 234–236, 238–255, 257– 269, 271–279, 281–291, 293, 294, 296– 301, 303–306, 308–313, 315–318, 322–331	
\protected@csedef 37, 38, 132, 402, 403	
\protected@csxdef 38, 132, 402, 403	
\protected@edef 29, 39, 63, 71, 93, 120, 131, 132, 149–151, 190, 206, 343	
\protected@write 11–13, 58, 66, 93, 123, 124, 141, 143, 145, 152, 153, 348, 424	
\providecommand 20, 21, 32, 52, 66, 89, 90, 93, 109, 118, 123, 136, 137, 143, 152, 339, 348, 350–352, 384, 419	

S	
<code>\s@glstrfmt</code>	33
<code>\s@glshyp@opt</code>	92
<code>\s@glstr@enabletagging</code>	194, 195
<code>\s@glstrfmt</code>	32
<code>\s@glstrifhasfield</code>	35
<code>\s@GlsXtrStartUnsetBuffering</code>	105
<code>\s@GlsXtrStopUnsetBuffering</code>	106
<code>\s@printunsrtglossary</code>	147, 149
<code>\seealsoname</code>	52, 53
<code>\seename</code>	49
<code>\setabbreviationstyle</code>	121, 227, 235
<code>\setacronymstyle</code>	121, 122
<code>\setentrycounter</code>	134, 341, 343
<code>\SetGenericNewAcronym</code>	122
<code>\setglossarystyle</code>	25, 127, 384, 386, 387, 397, 399, 400, 412–418, 421
<code>\setkeys</code>	9, 25, 29, 33, 72, 74, 91, 120, 127, 156, 204, 205
<code>\setlength</code>	63, 64, 396, 398, 400, 401, 412, 414–416, 421, 428, 429
<code>\settowidth</code>	122, 402–411, 427, 428
<code>\setupglossaries</code>	5, 29
<code>\sfcode</code>	18, 19, 198, 395
<code>\small</code>	27, 51
soul package	105
<code>\space</code>	6, 13, 52, 58, 60, 62, 91, 108–110, 116, 118, 119, 121–125, 127, 137, 139, 140, 144, 148, 150, 153, 198, 199, 203, 207, 227, 384, 385, 395, 398, 399, 401, 402, 411, 419
<code>\spacefactor</code>	18, 19, 198, 205, 395
<code>\stepcounter</code>	161
<code>\string</code>	6, 11–13, 30, 52, 53, 58–60, 62, 66, 73, 88, 91, 93, 100, 101, 108–110, 116, 118, 119, 121, 123–127, 136–141, 143–145, 148–150, 152, 153, 183, 185, 186, 188, 190, 198, 339, 348, 352–381, 424
<code>\strut</code>	384–395, 400, 426
<code>\subglossentry</code>	127, 151, 384–394, 396, 398, 400, 412, 422, 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447
<code>\subitem</code>	396
<code>\subsubitem</code>	396
<code>\symbolname</code>	428, 435, 436, 438, 439, 441, 442, 444, 446
T	
<code>\tabcolsep</code>	428, 429
<code>\tablehead</code>	391–394
<code>\tabletail</code>	391–394
<code>\tabularnewline</code>	387–395, 427, 430–447
<code>\TeX</code>	138
<code>\texorpdfstring</code>	34, 37, 101–103, 320, 331–337
<code>\textbf</code>	395, 427
textcase package	318
<code>\textsc</code>	242
<code>\textsmaller</code>	256
<code>\texttt</code>	27, 137–140
<code>\the</code>	120, 135, 142, 193, 194, 206, 225– 232, 234, 236–239, 241–247, 249, 250, 253–255, 257–260, 262, 264, 267, 269, 271–279, 281, 283, 285–291, 293, 294, 296–301, 303–306, 308–313, 315–318, 343
<code>\theglsentrycounter</code> ..	8, 10–12, 73, 75, 156
<code>\theH</code>	30
<code>\theHglentrycounter</code> ..	8, 10–12, 73, 75, 156
<code>\theindex</code>	189
<code>\thewrglossary</code>	26
<code>\this@dialect</code>	338, 382
<code>\thisgrptitle</code>	423
<code>\toks@</code>	135, 193, 194, 343
<code>\toprule</code>	427, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 445
<code>\TrackedDialectClosestSubMatch</code>	40
tracklang package	39, 40, 143, 352
<code>\TrackLangGetDefaultScript</code>	381
<code>\TrackLangIfHasDefaultScript</code>	381
<code>\TrackLangRequireDialectPrefix</code>	382
<code>\triangleright</code>	51
U	
<code>\u</code>	339
<code>\undef</code>	16, 57, 195
<code>\underline</code>	196
<code>\unskip</code>	44, 56, 384
upgreek package	351
<code>\usepackage</code>	139, 140
W	
<code>\warn@nomakeglossaries</code>	124
<code>\warn@noprntglossary</code>	124, 127
wrglossary (counter)	25, 26
<code>\write</code>	52, 109, 110, 118, 123, 136, 137, 144
X	
<code>\x</code>	135, 160, 343, 348
<code>\xcapitalisewords</code>	181
<code>\xdef</code>	127, 149, 339

\xifinlist	114	\XKV@checkchoice	66
\xifinlistcs	35	\XKV@plfalse	66
xindy	448	\XKV@resa	66
xindy	16, 122	\XKV@sttrue	66
xkeyval package	5		