

glossaries-extra.sty v1.44: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2020-03-23

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	32
1.3	Modifications to Commands Provided by glossaries	47
1.3.1	Existence Checks	52
1.3.2	Document Definitions	62
1.3.3	Existing Glossary Style Modifications	68
1.3.4	Entry Formatting, Hyperlinks and Indexing	72
1.3.5	Entry Counting	111
1.3.6	Acronym Modifications	127
1.3.7	Indexing and Displaying Glossaries	131
1.4	Link Counting	174
1.5	Integration with glossaries-accsupp	176
1.6	Categories	193
1.7	Abbreviations	220
1.7.1	Abbreviation Styles Setup	241
1.7.2	Predefined Styles (Default Font)	244
1.7.3	Predefined Styles (Small Capitals)	264
1.7.4	Predefined Styles (Fake Small Capitals)	280
1.7.5	Predefined Styles (Emphasized)	297
1.7.6	Predefined Styles (User Parentheses Hook)	322
1.7.7	Predefined Styles (Hyphen)	332
1.7.8	Predefined Styles (No Short on First Use)	347
1.8	Using Entries in Headings	350
1.9	Multi-Lingual Support	373
1.10	glossaries-extra-bib2gls.sty	374
2	Style Adjustments (glossaries-extra-stylemods.sty)	420
2.1	Package Initialisation	420
2.2	List-Like Styles	421
2.3	Longtable Styles	424
2.4	Long Ragged Styles	426
2.5	Supertabular Styles	428
2.6	Super Ragged Styles	430
2.7	Inline Style	432
2.8	Tree Styles	432
2.9	Multicolumn Styles	452

3	bookindex style (glossary-bookindex.sty)	458
3.1	Package Initialisation and Options	458
4	longextra styles (glossary-longextra.sty)	465
4.1	Package Initialisation and Options	465
5	topic styles (glossary-topic.sty)	488
5.1	Package Initialisation and Options	488
	Glossary	494
	Change History	495
	Index	520

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2020/03/23 v1.44 (NLCT)]

Requires xkeyval to define package options.
3 \RequirePackage{xkeyval}

Requires etoolbox package.
4 \RequirePackage{etoolbox}

Has glossaries already been loaded?
5 \@ifpackageloaded{glossaries}
6 {%

Already loaded so pass any options to \setupglossaries. This means that the options that
can only be set when glossaries is loaded can't be used.
7   \newcommand{\glstr@dooption}[1]{\setupglossaries{#1}}%
8   \let\@glstr@declareoption\@gls@declareoption
9 }
10 {%

Not already loaded, so pass options to glossaries.
11   \newcommand{\glstr@dooption}[1]{%
12     \PassOptionsToPackage{#1}{glossaries}%
13   }%

Set the defaults.
14   \PassOptionsToPackage{toc}{glossaries}
15   \PassOptionsToPackage{nopostdot}{glossaries}
16   \PassOptionsToPackage{noredefwarn}{glossaries}
17   \@ifpackageloaded{polyglossia}%
18   {%
19   {%
20     \@ifpackageloaded{babel}%
21     {\PassOptionsToPackage{translate=babel}{glossaries}}%
22     {%
23   }%
24   \newcommand*{\@glstr@declareoption}[2]{%
25     \DeclareOptionX{#1}{#2}%
26     \DeclareOption{#1}{#2}%
27   }
28 }
```

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

```

undefaction

```

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

```

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

\glstr@record Does nothing by default.

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

```

\glstr@recordsee Does nothing by default.

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

```

\glstr@defaultnumberformat

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

```

\GlsXtrSetDefaultNumberFormat

```

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

```

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

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

```

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

```

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

```

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

```

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

The first argument is the option list (as passed in the first optional argument to commands like `\gls`). This allows the `noindex` setting to be picked up. The second argument is the entry's

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

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

Save the label in case it's needed. This needs to be outside the existence check to allow the post-link hook to reference it.

```
121 \edef\@gls@label{\glsdetoklabel{#2}}%
122 \let\glslabel\@gls@label
123 \ifglsentryexists{#2}{}%
124 {%
125   \@@glsxtrwrglossmark
126   \begingroup
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\@glxtr@org@theHvalue\@glxtr@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}

```

lslink@postkeys

```

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

```

lossadd@prekeys

```

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

```

ossadd@postkeys

```

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

```

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

```

173 \newcommand*\@glxtr@dorecord{%

```

```

174 \global\let\@glsrecordlocref\theglentrycounter
175 \let\@glsxtr@orgprefix\@glo@counterprefix
176 \ifx\theglentrycounter\theHglentrycounter
177   \def\@glo@counterprefix{}%
178 \else
  Protect against non-expandable commands occurring in the location.
179   \protected@edef\@glsxtr@theentrycounter{\theglentrycounter}%
180   \protected@edef\@glsxtr@theHentrycounter{\theHglentrycounter}%
181   \@onelevel@sanitize\@glsxtr@theentrycounter
182   \@onelevel@sanitize\@glsxtr@theHentrycounter
183   \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
184     {\@glsxtr@theentrycounter}{\@glsxtr@theHentrycounter}%
185   }%
186   \@do@gls@getcounterprefix
187 \fi

```

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

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

```

188 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
189   \@glsxtr@do@nameref@record
190   {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
191   {\@glsrecordlocref}%
192 \else
193   \protected@write\@auxout{}{\string\@glsxtr@record
194     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
195     {\@glsrecordlocref}}%
196 \fi
197 \@glsxtr@counterrecordhook
198 \let\@glo@counterprefix\@glsxtr@orgprefix
199 }

```

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

```

200 \newcommand*\@glsxtr@dorecordnodefer{%
201   \ifx\theglentrycounter\theHglentrycounter
202     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
203       \@glsxtr@do@nameref@record
204       {\@gls@label}{\@gls@counter}{\@glsnumberformat}%
205       {\theglentrycounter}%
206     \else
207       \protected@write\@auxout{}{\string\@glsxtr@record
208         {\@gls@label}{\@gls@counter}{\@glsnumberformat}%
209         {\theglentrycounter}}%
210     \fi
211   \else
212     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix

```

```

213     {\theglentrycounter}{\theHglentrycounter}%
214 }%
215 \@do@gl@getcounterprefix
216 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
217   \@glxtr@do@nameref@record
218     {\@gl@label}{\@glo@counterprefix}{\@gl@counter}%
219     {\@gl@numberformat}{\theglentrycounter}%
220 \else
221   \protected@write\@auxout{}\string\glxtr@record
222     {\@gl@label}{\@glo@counterprefix}{\@gl@counter}{\@gl@numberformat}%
223     {\theglentrycounter}}%
224 \fi
225 \fi
226 \@glxtr@counterrecordhook
227 }

```

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

```

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

```

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

```

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

```

`\glxtr@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 `\theglentrycounter`.

```

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

```

```

245     {\theHglentrycounter}}%
246   }%
247 }

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

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

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

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

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

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

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

addloclistfield
274 \newcommand*{\glxtr@addloclistfield}{%
275   \key@ifundefined{glossentry}{loclist}%
276   {%

```

```

277 \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
278 \appto\@gls@keymap{,{loclist}{loclist}}%
279 \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
280 \appto\@newglossaryentryposthook{%
281   \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
282 }%
283 \glssetnoexpandfield{loclist}%
284 }%
285 {}%

```

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

```

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

```

Add a key to store the group heading.

```

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

```

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

```

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

```

alsoindex

```

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

```

only

```

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

```

nameref

```

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

```

@if@record@only

```
313 \newcommand*{\@glsxtr@if@record@only}[2]{%
314   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
315     #1%
316   \else
317     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
318       #1%
319     \else
320       #2%
321     \fi
322   \fi
323 }
```

ord@setting@off

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

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

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

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

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

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

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

Warn about using \printglossary:

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

Load glossaries-extra-bib2gls:

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

record Now define the record package option.

```
344 \define@choicekey{glossaries-extra.sty}{record}
345 [ \@glsxtr@record@setting\glsxtr@record@nr ]%
```

```

346 {off,only,alsoindex,nameref}%
347 [only]%
348 {%
349 \ifcase\glxtr@record@nr\relax

```

Don't record.

```

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

```

Only record (don't index).

```

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

```

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

```

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

```

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

```

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

```

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

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

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

Need to provide conditional commands that are backward compatible:

`if@glxtrdocdef`

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

`lsxtrdocdeftrue`

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

`sxtrdocdeffalse`

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

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

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

`ocdefrestricted`

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

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

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

`indexcrossrefs` Automatically index cross references at the end of the document

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

Switch off since this can increase the build time.

```
406 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
407 \newcommand*{\@glsxtr@autoindexcrossrefs}{\@glsxtr@indexcrossrefstrue}
```

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

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

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

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

\glsxtr@float

```
414 \let\glsxtr@float\@float
```

glsxtr@dblfloat

```
415 \let\glsxtr@dblfloat\@dblfloat
```

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

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

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

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

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

postdot Shortcut for nopostdot=false

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

nopostdot Needs to redefine \@glxtr@defpostpunc

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

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

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

```

473 \renewcommand*{\@glsxtr@defpostpunc}{%
474 \renewcommand*{\glspostdescription}{#1}%
475 }%
476 }%
477 }%
478 }%
479 }

```

`glsxtrabbrvtype` Glossary type for abbreviations.

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

`bbreviationsdef` Set by abbreviations option.

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

```

`bbreviationsdef`

```

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

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

```

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

```

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

```

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

```

500 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
501 \newcommand*{\ab}{\cgl}s}%
502 \newcommand*{\abp}{\cgl}spl}%
503 \newcommand*{\as}{\glsxtrshort}%
504 \newcommand*{\asp}{\glsxtrshortpl}%
505 \newcommand*{\al}{\glsxtrlong}%
506 \newcommand*{\alp}{\glsxtrlongpl}%
507 \newcommand*{\af}{\glsxtrfull}%

```

```

508 \newcommand*{\afp}{\glxstrfullpl}%
509 \newcommand*{\Ab}{\cGls}%
510 \newcommand*{\Abp}{\cGlspl}%
511 \newcommand*{\As}{\Glsxtrshort}%
512 \newcommand*{\Asp}{\Glsxtrshortpl}%
513 \newcommand*{\Al}{\Glsxtrlong}%
514 \newcommand*{\Alp}{\Glsxtrlongpl}%
515 \newcommand*{\Af}{\Glsxtrfull}%
516 \newcommand*{\Afp}{\Glsxtrfullpl}%
517 \newcommand*{\AB}{\cGLS}%
518 \newcommand*{\ABP}{\cGLSpl}%
519 \newcommand*{\AS}{\GLSxtrshort}%
520 \newcommand*{\ASP}{\GLSxtrshortpl}%
521 \newcommand*{\AL}{\GLSxtrlong}%
522 \newcommand*{\ALP}{\GLSxtrlongpl}%
523 \newcommand*{\AF}{\GLSxtrfull}%
524 \newcommand*{\AFP}{\GLSxtrfullpl}%

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

```

Disable this command after it's been used.

```

526 \let\GlsXtrDefineAbbreviationShortcuts\relax
527 }

```

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

```

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

```

```

552 \newcommand*{\ACFP}{\GLSxtrfullpl}%
553 \providecommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

554 \let\GLSxtrDefineAcShortcuts\relax
555 }

```

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

```

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

```

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

@setupshortcuts Command used to set the shortcuts option.

```

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

```

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

```

569 \newcommand*{\@glxtr@shortcutsval}{\ifglacrshortcuts acro\else none\fi}%

```

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

```

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

```

```

581     \DefineAcronymSynonyms
582 }%
583 \or % abbreviations
584   \renewcommand*{\@glxtr@setupshortcuts}{%
585     \GlsXtrDefineAbbreviationShortcuts
586   }%
587 \or % abbr
588   \renewcommand*{\@glxtr@setupshortcuts}{%
589     \GlsXtrDefineAbbreviationShortcuts
590   }%
591 \or % other
592   \renewcommand*{\@glxtr@setupshortcuts}{%
593     \GlsXtrDefineOtherShortcuts
594   }%
595 \or % all
596   \renewcommand*{\@glxtr@setupshortcuts}{%
597     \glsacrshortcutstrue
598     \GlsXtrDefineAcShortcuts
599     \GlsXtrDefineAbbreviationShortcuts
600     \GlsXtrDefineOtherShortcuts
601   }%
602 \or % true
603   \renewcommand*{\@glxtr@setupshortcuts}{%
604     \glsacrshortcutstrue
605     \GlsXtrDefineAcShortcuts
606     \GlsXtrDefineAbbreviationShortcuts
607     \GlsXtrDefineOtherShortcuts
608   }%
609 \or % ac
610   \renewcommand*{\@glxtr@setupshortcuts}{%
611     \glsacrshortcutstrue
612     \GlsXtrDefineAcShortcuts
613   }%
614 \else % none, false
615   \renewcommand*{\@glxtr@setupshortcuts}{}%
616 \fi
617 }

```

Leave none and false as last option.

```

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

```

`\lsxtr@doaccsupp`

```

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

```

glossaries-accsupp can't be loaded after glossaries-extra. glossaries-accsupp v4.29+ checks `\@glxtr@doaccsupp` to determine if it's been loaded too late.

accsupp If accsupp, load glossaries-accsupp package.

```

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

```

tr@doloadprefix

```

621 \newcommand*{\@glsxtr@doloadprefix}{}

```

prefix If prefix, load glossaries-prefix package.

```

622 \@glsxtr@declareoption{prefix}{%
623 \renewcommand*{\@glsxtr@doloadprefix}{\RequirePackage{glossaries-prefix}}}

```

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

```

624 \newcommand{\glsxtrNoGlossaryWarning}[1]{%
625 \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
626 \@glsxtr@defaultnoglossarywarning{#1}%
627 }

```

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

```

628 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
629 [ \@glsxtr@nomissingglstextval \@glsxtr@nomissingglstextnr ]%
630 {true,false}[true]{%
631 \ifcase \@glsxtr@nomissingglstextnr \relax % true
632 \renewcommand{\glsxtrNoGlossaryWarning}[1]{\null}%
633 \else % false
634 \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
635 \@glsxtr@defaultnoglossarywarning{#1}%
636 }%
637 \fi
638 }

```

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

xtr@redefstyles

```

639 \newcommand*{\@glsxtr@redefstyles}{}

```

stylemods

```

640 \define@key{glossaries-extra.sty}{stylemods}[default]{%
641 \ifstrequal{#1}{default}%
642 {%
643 \renewcommand*{\@glsxtr@redefstyles}{%
644 \RequirePackage{glossaries-extra-stylemods}}%
645 }%
646 {%
647 \ifstrequal{#1}{all}%
648 {%
649 \renewcommand*{\@glsxtr@redefstyles}{%
650 \PassOptionsToPackage{all}{glossaries-extra-stylemods}%

```



```

651     \RequirePackage{glossaries-extra-stylemods}%
652 }%
653 }%
654 {%
655     \renewcommand*{\@glsxtr@redefstyles}{}%
656     \@for\@glsxtr@tmp:=#1\do{%
657         \IfFileExists{glossary-\@glsxtr@tmp.sty}%
658         {%
659             \eappto\@glsxtr@redefstyles{%
660                 \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
661             }%
662         {%
663             \PackageError{glossaries-extra}%
664                 {Glossaries style package ‘glossary-\@glsxtr@tmp.sty’
665                     doesn’t exist (did you mean to use the ‘style’ key?)}%
666                 {The list of values (#1) in the ‘stylemods’ key should
667                     match the glossary-xxx.sty files provided with
668                     glossaries.sty}%
669             }%
670         }%
671         \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
672     }
673 }%
674 }

```

glsxtr@do@style

```

675 \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.

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

679 \setglossarystyle{#1}%

```

```

680 }%

```

```

681 }

```

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.

```

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

```

cationHyperlink

$\backslash\mathrm{glxtrinternallocationhyperlink}\{\langle counter\rangle\}\{\langle prefix\rangle\}\{\langle location\rangle\}$

The first two arguments are always control sequences.

```
683 \newcommand*\GlsXtrInternalLocationHyperlink}[3]{%
684   \glxtrhyperlink{#1#2#3}{#3}%
685 }
```

locationhyperlink

```
686 \newcommand*\@glxtr@wrglossary@locationhyperlink}[3]{%
687   \pageref{wrglossary.#3}%
688 }
```

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.

```
689 \@glxtr@declareoption{indexcounter}{%
690   \glxtr@doption{counter=wrglossary}%
691   \ifundef\c@wrglossary
692   {%
693     \newcounter{wrglossary}%
694     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
695   }%
696   {}%
697   \renewcommand*\@glxtr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is wrglossary.

```
698   \ifdefstring\@gl@counter{wrglossary}%
699   {%
700     \refstepcounter{wrglossary}%
701     \label{wrglossary.\thewrglossary}%
702   }%
703   {}%
704 }%
705 \renewcommand*\GlsXtrInternalLocationHyperlink}[3]{%
706   \ifdefstring\gl@entrycounter{wrglossary}%
707   {%
708     \@glxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
709   }%
710   {\glxtrhyperlink{##1##2##3}{##3}}%
711 }%
712 }
```

glxtrwrglossmark

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

```
713 \newcommand*\@glxtrwrglossmark{}\}
```

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

```
714 \newcommand*{\@@glsxtrwrglossmark}{}
715 \AtBeginDocument{\renewcommand*{\@@glsxtrwrglossmark}{\@@glsxtrwrglossmark}}
```

sxtrwrglossmark Does nothing by default.

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

debug Provide extra debug options.

```
717 \define@choicekey{glossaries-extra.sty}{debug}
718 [ \@@glsxtr@debugval \@@glsxtr@debugnr ] %
719 {true,false,showtargets,showwrgloss,all,showaccsupp}[true] {%
720   \ifcase \@@glsxtr@debugnr \relax % true
721     \glsxtr@doption{debug=true} %
722     \renewcommand*{\@@glsxtrwrglossmark}{} %
723   \or % false
724     \glsxtr@doption{debug=false} %
725     \renewcommand*{\@@glsxtrwrglossmark}{} %
726   \or % showtargets
727     \glsxtr@doption{debug=showtargets} %
728   \or % showwrgloss
729     \glsxtr@doption{debug=true} %
730     \renewcommand*{\@@glsxtrwrglossmark}{\glsxtrwrglossmark} %
731   \or % all
732     \glsxtr@doption{debug=showtargets,debug=showaccsupp} %
733     \renewcommand*{\@@glsxtrwrglossmark}{\glsxtrwrglossmark} %
734   \or % showaccsupp
735     \glsxtr@doption{debug=showaccsupp} %
736   \fi
737 }
```

Pass all other options to glossaries.

```
738 \DeclareOptionX*{%
739   \expandafter \glsxtr@doption \expandafter {\CurrentOption}}
```

Process options.

```
740 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
741 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
742 \@@glsxtr@doaccsupp
```

Load the glossaries-prefix package if required.

```
743 \@@glsxtr@doloadprefix
```

Redefine \glspostdescription if required.

```
744 \@@glsxtr@defpostpunc
```

`\glsshowtarget` This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using `\def`. `\glsshowtargetouter` was introduced in glossaries v4.45, so that also may not be defined.

```

745 \ifdef\glsshowtargetouter
746 {
747   \renewcommand*{\glsshowtarget}[1]{%
748     \glstrtitleorpdforheading
749     {%
750       \ifmmode
751         \nfss@text{\glsshowtargetfont [#1]}%
752       \else
753         \ifinner
754           {\glsshowtargetfont [#1]}%
755         \else
756           \glsshowtargetouter{#1}%
757         \fi
758       \fi
759     }%
760     {[#1]}%
761     {\protect\glsshowtargetfont [#1]}%
762   }
763 }
764 {

```

Old definition.

```

765 \def\glsshowtarget#1{%
766   \glstrtitleorpdforheading
767   {%
768     \ifmmode
769       \texttt{\small [#1]}%
770     \else
771       \ifinner
772         \texttt{\small [#1]}%
773       \else
774         \marginpar{\texttt{\small #1}}%
775       \fi
776     \fi
777   }%
778   {[#1]}%
779   {\texttt{\small [#1]}%
780 }
781 }

```

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

```

782 \let\@glstr@org@doseeglossary\@do@seeglossary

```

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

```

783 \newcommand*{\@glstr@doseeglossary}[2]{%
784   \glstoifexists{#1}%

```

```

785  {%
786    \@glsxtrwrglossmark
787    \@glsxtr@org@doseeglossary{#1}{#2}%
788  }%
789 }

oindex@glossary
790 \newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
791   \@glsxtr@recordsee{#1}{#2}%
792   \@glsxtr@doseeglossary{#1}{#2}%
793 }

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

                  Check if user tried autoseeindex=false when it can't be supported.
795 \if@glsxtr@autoseeindex
796 \else
797   \ifdef\@glsxtr@org@gloautosee
798   {%
799     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
800       option requires at least v4.30 of glossaries.sty}%
801     {You need to update the glossaries.sty package}%
802   }
803 \fi

\@glo@autosee  If \@glo@autosee has been defined (glossaries v4.30 onwards), redefine it to test the au-
               toseeindex option.
804 \ifdef\@glo@autosee
805 {%
806   \renewcommand*{\@glo@autosee}{%
807     \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
808 }%
809 {}

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.
810 \renewcommand*{\gls@checkseeallowed}{%
811   \if@glsxtr@autoseeindex\@gls@see@noindex\fi
812 }

                  Define abbreviations glossaries if required.
813 \@glsxtr@abbreviationsdef
814 \let\@glsxtr@abbreviationsdef\relax

                  Setup shortcuts if required.
815 \@glsxtr@setupshortcuts

```

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

```

816 \@glxtr@redef@forglentries

```

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

```

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

```

Disable options that can only be used when the package is loaded:

```

818 \disable@keys{glossaries-extra.sty}{accsupp}

```

Now define the user command:

```

819 \newcommand*{\glossariesextrasetup}[1]{%
820   \let\glxtr@setup@record\relax
821   \let\glxtr@setupshortcuts\relax
822   \let\glxtr@redef@forglentries\relax
823   \let\glxtr@doloadprefix\relax
824   \setkeys{glossaries-extra.sty}{#1}%
825   \@glxtr@abbreviationsdef
826   \let\glxtr@abbreviationsdef\relax
827   \@glxtr@setupshortcuts
828   \glxtr@setup@record
829   \@glxtr@redef@forglentries
830   \glxtr@doloadprefix
831 }

```

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

```

832 \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.

```

833 \newcommand*{\glxtr@@do@wrglossary}[1]{%
834   \@glxtrwrglossmark
835   \glxtr@inc@wrglossaryctr{#1}%
836   \glxtr@org@@do@wrglossary{#1}%
837 }

```

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

```

838 \let\glxtr@saveentrycounter\@gls@saveentrycounter

```

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

```

839 \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter

```

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

```

840 \renewcommand*\@gls@getcounterprefix[2]{%
841   \protected@edef\@gls@thisloc{#1}\protected@edef\@gls@thisHloc{#2}%
842   \ifx\@gls@thisloc\@gls@thisHloc
843     \def\@glo@counterprefix{}%
844   \else
845     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
846       \def\@glo@tmp{##2}%
847       \ifx\@glo@tmp\@empty
848         \def\@glo@counterprefix{}%
849       \else
850         \def\@glo@counterprefix{##1}%
851       \fi
852     }%
853   \@gls@get@counterprefix#2.#1\end@getprefix

```

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

```

854   \ifx\@glo@counterprefix\@empty
855     \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
856     \else
857       \GlossariesExtraWarning{Hyper target ‘#2’ can’t be formed by
858       prefixing^^Jlocation ‘#1’. You need to modify the
859       definition of \string\theH\@gls@counter^^Jotherwise you
860       will get the warning: “name{\@gls@counter.#1}’ has been^^J
861       referenced but does not exist”%
862       \ifx\@glxtr@record@setting\@glxtr@record@setting@only
863       . You may want to consider using record=nameref instead%
864     \fi}%
865   \fi
866 \fi
867 \fi
868 }

```

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

sxtrdialecthook

```

869 \newcommand*\@glxtrdialecthook{}

```

Set up record option if required.

```

870 \glxtr@setup@record

```

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

```

871 \AtBeginDocument{%
872   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
873   \def\@glxtrundeftag{\glxtrundeftag}%
874 }

```

1.2 Extra Utilities

usedOrUndefined

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

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

```
875 \newcommand*\GlsXtrIfUnusedOrUndefined}[3]{%
876   \ifglentryexists{#1}%
877   {\ifbool{glo@glsetoklabel{#1}@flag}{#3}{#2}}%
878   {#2}%
879 }
```

Starred form of `\ifglossaryexists` was only introduced to glossaries v4.46 so provide it if it hasn't been defined.

```
880 \ifdef\s@ifglossaryexists
881 {}
882 {
```

`\ifglossaryexists`

```
883   \renewcommand{\ifglossaryexists}{%
884     \@ifstar\s@ifglossaryexists\@ifglossaryexists
885   }
```

`\ifglossaryexists`

```
886   \newcommand{\@ifglossaryexists}[3]{%
887     \ifcsundef{glotype@#1@out}{#3}{#2}%
888   }
```

`\ifglossaryexists`

```
889   \newcommand{\s@ifglossaryexists}[3]{%
890     \ifcsundef{glolist@#1}{#3}{#2}%
891   }
892 }
```

ifemptyglossary

```
\glxtrifemptyglossary{<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.)

```

893 \newcommand{\glxtrifemptyglossary}[3]{%
894   \ifcsdef{glolist@#1}%
895   {%
896     \ifcsstring{glolist@#1}{,}{#2}{#3}%
897   }%
898   {%
899     \glxtrundefaction{Glossary type '#1' doesn't exist}{}%
900     #2%
901   }%
902 }
```

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

```

903 \newcommand*{\glxtrifkeydefined}[3]{%
904   \key@ifundefined{glossentry}{#1}{#3}{#2}%
905 }
```

ovidestoragekey Like `\gl saddstoragekey` but does nothing if the key has already been defined.

```

906 \newcommand*{\glxtrprovidestoragekey}{%
907   \@ifstar\@sglsxtr@provide@storagekey\@glxtr@provide@storagekey
908 }
```

vide@storagekey Unstarred version.

```

909 \newcommand*{\@glxtr@provide@storagekey}[3]{%
910   \key@ifundefined{glossentry}{#1}%
911   {%
912     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
913     \appto\@gls@keymap{,}{#1}{#1}}%
914     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
915     \appto\@newglossaryentryposthook{%
916       \letcs{\@glo@tmp}{@glo@#1}%
917       \gl s@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
918     }%

```

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

```

919   \ifblank{#3}
920   {%
921   {%
922     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
923   }%
924   }%
925   {%

```

Provide the no-link command if not already defined.

```

926   \ifblank{#3}
927   {%
928   {%

```

```

929     \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
930   }%
931 }%
932 }

```

vide@storagekey Starred version.

```

933 \newcommand*{\s@glstr@provide@storagekey}[1]{%
934   \key@ifundefined{glossentry}{#1}%
935   {%
936     \expandafter\newcommand\expandafter*\expandafter
937     {\csname gls@assign@#1@field\endcsname}[2]{%
938       \@gls@expand@field{##1}{#1}{##2}%
939     }%
940   }%
941   }%
942   \glstr@provide@addstoragekey{#1}%
943 }

```

The name of a text-block control sequence can be stored in a field (given by `\GlsXtrFmtField`). This command can then be used with `\glstrfmt[<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

```

944 \newcommand{\GlsXtrFmtField}{useri}

```

tDefaultOptions

```

945 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

946 \newrobustcmd*{\glstrfmt}{\@ifstar\s@glstrfmt\@glstrfmt}

```

`\@glstrfmt` Unstarred form.

```

947 \newcommand*{\@glstrfmt}[3][\@glstrfmt{#1}{#2}{#3}{}

```

`\s@glstrfmt` Starred form.

```

948 \newcommand*{\s@glstrfmt}[3][\@glstrfmt{#1}{#2}{#3}{}
949 \new@ifnextchar[\s@glstrfmt{#1}{#2}{#3}{}%
950 {\@glstrfmt{#1}{#2}{#3}{}%
951 }

```

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

```

952 \def\s@glstrfmt#1#2#3[#4]{\@glstrfmt{#1}{#2}{#3}{#4}}

```

`\@glstrfmt` Actual inner working.

```

953 \newcommand*{\@glstrfmt}[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).

```

954 \begingroup
955   \def\glslabel{#2}%
956   \glsdoifexistsordo{#2}%
957   {%
958     \ifglshasfield{\GlsXtrFmtField}{#2}%
959     {%
960       \let\do@gl@link@checkfirsthyper\relax
961       \expandafter\@gl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
962       {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
963     }%
964     {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
965   }%
966   {%

```

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

```

967     \begingroup
968       \@gl@setdefault@gl@link@opts
969       \setkeys{gl@link}{\GlsXtrFmtDefaultOptions,#1}%
970       \ifKV@gl@link@noindex\else\glsadd{#2}\fi
971     \endgroup
972     \glstrfmtdisplay{@firstofone}{#3}{#4}%
973   }%
974 \endgroup
975 }

```

glstrfmtdisplay The command used internally by \glstrfmt 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).

```

976 \newcommand{\glstrfmtdisplay}[3]{\csuse{#1}{#2}#3}

```

glxtrententryfmt No link or indexing.

```

977 \ifdef\texorpdfstring
978 {
979   \newcommand*{\glxtrententryfmt}[2]{%
980     \texorpdfstring{\@glxtrententryfmt{#1}{#2}}{\glstrpdfentryfmt{#1}{#2}}%
981   }
982 }
983 {
984   \newcommand*{\glxtrententryfmt}{\@glxtrententryfmt}
985 }

```

sxtrpdfentryfmt Use for the PDF bookmarks.

```

986 \newcommand*{\glxtrpdfentryfmt}[2]{#2}

```

@glxtrententryfmt

```

987 \newrobustcmd*{\@glxtrententryfmt}[2]{%

```

Locally define \glslabel in case the helper command needs to access the label.

```

988 {%
989   \edef\glslabel{#1}%
990   \glsdoidexistsordo{#1}%
991   {%
992     \ifglshasfield{\GlsXtrFmtField}{#1}%
993     {%
994       \csuse{\glscurrentfieldvalue}{#2}%
995     }%
996     {#2}%
997   }%
998   {#2}%
999 }%
1000 }

```

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

```

1001 \newcommand*\glsxtrfieldlistadd}[3]{%
1002   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
1003 }

```

trfieldlistgadd Similarly but uses \listcsgadd.

```

1004 \newcommand*\glsxtrfieldlistgadd}[3]{%
1005   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
1006 }

```

trfieldlistead Similarly but uses \listcseadd.

```

1007 \newcommand*\glsxtrfieldlistead}[3]{%
1008   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
1009 }

```

trfieldlistxadd Similarly but uses \listcsxadd.

```

1010 \newcommand*\glsxtrfieldlistxadd}[3]{%
1011   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
1012 }

```

Now provide commands to iterate over these lists.

fielddolistloop

```

1013 \newcommand*\glsxtrfielddolistloop}[2]{%
1014   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
1015 }

```

fieldforlistloop

```

1016 \newcommand*\glsxtrfieldforlistloop}[3]{%
1017   \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
1018 }

```

fieldformatlist

```
1019 \newrobustcmd*{\glxtrfieldformatlist}[2]{%
1020   \begingroup
1021   \def\@dtl@formatlist@itemsep{}%
1022   \def\@dtl@formatlist@lastitem{}%
1023   \def\@dtl@formatlist@prelastitem{}%
1024   \def\@dtl@formatlist@prelastitemsep{}%
1025   \forlistcsloop{\@dtl@formatlist@handler}{glo@\glsdetoklabel{#1}@#2}%
1026   \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
1027   \endgroup
1028 }
```

List element tests:

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

```
1029 \newcommand*{\glxtrfieldifinlist}[5]{%
1030   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
1031 }
```

rfieldxifinlist Expands item.

```
1032 \newcommand*{\glxtrfieldxifinlist}[5]{%
1033   \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
1034 }
```

sxtrforcsvfield

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

```
1035 \newcommand*{\glxtrforcsvfield}[3]{%
1036   \@glxtrifhasfield{#2}{#1}%
1037   {%
1038     \let\glxxtrendfor\@endfortrue
1039     \@for\@glxtr@label:=\glscurrentfieldvalue\do
1040     {\expandafter#3\expandafter{\@glxtr@label}}}%
1041   }%
1042 }
```

ldformatcsvlist

```
1043 \newrobustcmd*{\glxtrfieldformatcsvlist}[2]{%
1044   \@glxtrifhasfield{#2}{#1}%
1045   {\@dtlformatlist\glscurrentfieldvalue}%
1046   }%
1047 }
```

dValueInCsvList

$\backslash\text{GlsXtrIfFieldValueInCsvList}\{\langle label \rangle\}\{\langle field \rangle\}\{\langle list \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

```

1048 \newcommand*\GlsXtrIfFieldValueInCsvList}{%
1049   \@ifstar\s@GlsXtrIfFieldValueInCsvList\@GlsXtrIfFieldValueInCsvList
1050 }

```

Note `\DTLifinlist` performs one level on the list but not the element.

`dValueInCsvList` Unstarred version.

```

1051 \newcommand*\@GlsXtrIfFieldValueInCsvList}[5]{%
1052   \@glstrifhasfield{#2}{#1}%
1053   {%
1054     \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
1055     {#3}{#4}{#5}%
1056   }%
1057   {#5}%
1058 }

```

`dValueInCsvList` Starred version.

```

1059 \newcommand*\s@GlsXtrIfFieldValueInCsvList}[5]{%
1060   \s@glstrifhasfield{#2}{#1}%
1061   {%
1062     \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
1063     {#3}{#4}{#5}%
1064   }%
1065   {#5}%
1066 }

```

`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.

```

1067 \newrobustcmd{\glstrifhasfield}{%
1068   \@ifstar{\s@glstrifhasfield}\@glstrifhasfield}%
1069 }

```

`lsxtrifhasfield` Unstarred version adds grouping.

```

1070 \newcommand*\@glstrifhasfield}[4]{%
1071   {\s@glstrifhasfield{#1}{#2}{#3}{#4}}%
1072 }

```

`lsxtrifhasfield` Starred version omits grouping.

```

1073 \newcommand*\s@glstrifhasfield}[4]{%
1074   \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
1075   \ifundef\glscurrentfieldvalue
1076     {#4}%
1077     {%
1078       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
1079     }%
1080 }

```

rIfFieldNonZero Designed for numeric fields.

```
1081 \newcommand{\GlsXtrIfFieldNonZero}{%
1082   \@ifstar\s@GlsXtrIfFieldNonZero\@GlsXtrIfFieldNonZero
1083 }
```

rIfFieldNonZero

```
1084 \newcommand{\@GlsXtrIfFieldNonZero}[4]{%
1085   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
1086 }
```

XtrIfFieldEqNum

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

Designed for numeric fields.

```
1087 \newcommand{\GlsXtrIfFieldEqNum}{%
1088   \@ifstar\s@GlsXtrIfFieldEqNum\@GlsXtrIfFieldEqNum
1089 }
```

XtrIfFieldEqNum

```
1090 \newcommand{\@GlsXtrIfFieldEqNum}[5]{%
1091   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1092 }
```

XtrIfFieldEqNum

```
1093 \newcommand{\s@GlsXtrIfFieldEqNum}[5]{%
1094   \s@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1095 }
```

trIfFieldCmpNum

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

Designed for numeric fields.

```
1096 \newcommand{\GlsXtrIfFieldCmpNum}{%
1097   \@ifstar\s@GlsXtrIfFieldCmpNum\@GlsXtrIfFieldCmpNum
1098 }
```

trIfFieldCmpNum

```
1099 \newcommand{\@GlsXtrIfFieldCmpNum}[6]{%
1100   {%
1101     \letcs{\glscurrentfieldvalue}{glo@glstdetoklabel{#2}@#1}%
1102     \ifundef\glscurrentfieldvalue
1103     {\def\glscurrentfieldvalue{0}}%

```

```

1104   {%
1105     \ifdefempty\glscurrentfieldvalue
1106     {\def\glscurrentfieldvalue{0}}%
1107     {}%
1108   }%
1109   \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1110 }%
1111 }

```

trIfFieldCmpNum

```

1112 \newcommand{\s@GlsXtrIfFieldCmpNum}[6]{%
1113   \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
1114   \ifundef\glscurrentfieldvalue
1115   {\def\glscurrentfieldvalue{0}}%
1116   {%
1117     \ifdefempty\glscurrentfieldvalue
1118     {\def\glscurrentfieldvalue{0}}%
1119     {}%
1120   }%
1121   \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1122 }

```

XtrIfFieldUndef

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

Just uses \ifcsundef.

```

1123 \newcommand{\GlsXtrIfFieldUndef}[2]{%
1124   \ifcsundef{glo@\glsetoklabel{#2}@#1}%
1125 }

```

$\backslash\text{glxtrusefield}$ Provide a user-level alternative to $\backslash\text{@gls@entry@field}$. The first argument is the entry label. The second argument is the field label.

```

1126 \newcommand*{\glxtrusefield}[2]{%
1127   \@gls@entry@field{#1}{#2}%
1128 }

```

$\backslash\text{Glsxtrusefield}$ Provide a user-level alternative to $\backslash\text{@Gls@entry@field}$.

```

1129 \ifdef\texorpdfstring
1130 {
1131   \newcommand*{\Glsxtrusefield}[2]{%
1132     \texorpdfstring
1133     {\@Gls@entry@field{#1}{#2}}
1134     {\@gls@entry@field{#1}{#2}}%
1135   }
1136 }
1137 {

```



```

1138 \newcommand*{\GLSxtrusefield}[2]{%
1139   \@GLS@entry@field{#1}{#2}%
1140 }
1141 }

```

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

```

1142 \ifdef\teorpdfstring
1143 {
1144   \newcommand*{\GLSxtrusefield}[2]{%
1145     \teorpdfstring
1146       {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@GLS@entry@field{#1}{#2}}}}%
1147       {\@GLS@entry@field{#1}{#2}}%
1148   }
1149 }
1150 {
1151   \newcommand*{\GLSxtrusefield}[2]{%
1152     \glsdoifexists{#1}{\mfirstucMakeUppercase{\@GLS@entry@field{#1}{#2}}}%
1153   }
1154 }

```

`entryparentname`

```

1155 \newcommand*{\glstrentryparentname}[1]{%
1156   \ifcsdef{glo@\glsdetoklabel{#1}@parent}%
1157   {\csuse{glo@\csuse{glo@\glsdetoklabel{#1}@parent}@name}}%
1158   {}%
1159 }

```

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

```

1160 \newcommand*{\glsxtrdeffield}[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}

```

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

```

1161 \newcommand*{\glstxreffield}[2]{\protected@csdef{glo@\glsdetoklabel{#1}@#2}}

```

`etfieldifexists`

```

1162 \newcommand*{\glstxrsetfieldifexists}[3]{\glsdoifexists{#1}{#3}}

```

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

```

1163 \newrobustcmd*{\GlsXtrSetField}[3]{%
1164   \glstxrsetfieldifexists{#1}{#2}%
1165   {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
1166 }

```

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

```

1167 \newrobustcmd*{\GlsXtrLetField}[3]{%
1168   \glstxrsetfieldifexists{#1}{#2}%
1169   {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
1170 }

```

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

```
1171 \newrobustcmd*{\csGlsXtrLetField}[3]{%
1172   \glstrsetfieldifexists{#1}{#2}%
1173   {\csletcs{glo@glstdetoklabel{#1}@#2}{#3}}%
1174 }
```

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.

```
1175 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
1176   \glstrsetfieldifexists{#1}{#2}%
1177   {\csletcs{glo@glstdetoklabel{#1}@#2}{glo@glstdetoklabel{#3}@#4}}%
1178 }
```

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

```
1179 \newrobustcmd*{\gGlsXtrSetField}[3]{%
1180   \glstrsetfieldifexists{#1}{#2}%
1181   {\csgdef{glo@glstdetoklabel{#1}@#2}{#3}}%
1182 }
```

xGlsXtrSetField

```
1183 \newrobustcmd*{\xGlsXtrSetField}[3]{%
1184   \glstrsetfieldifexists{#1}{#2}%
1185   {\protected@csxdef{glo@glstdetoklabel{#1}@#2}{#3}}%
1186 }
```

eGlsXtrSetField

```
1187 \newrobustcmd*{\eGlsXtrSetField}[3]{%
1188   \glstrsetfieldifexists{#1}{#2}%
1189   {\protected@csedef{glo@glstdetoklabel{#1}@#2}{#3}}%
1190 }
```

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

```
1191 \newcommand*{\GlsXtrIfFieldEqStr}{%
1192   \@ifstar\sGlsXtrIfFieldEqStr\@GlsXtrIfFieldEqStr
1193 }
```

XtrIfFieldEqStr

```
1194 \newrobustcmd*{\@GlsXtrIfFieldEqStr}[5]{%
1195   \@glstrifhasfield{#1}{#2}%
1196   {%
1197     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1198   }%
1199   {#5}%
1200 }
```

XtrIfFieldEqStr

```
1201 \newrobustcmd*{\sGlsXtrIfFieldEqStr}[5]{%
```

```

1202 \s@glxtrifhasfield{#1}{#2}%
1203 {%
1204 \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1205 }%
1206 {#5}%
1207 }

```

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

```

1208 \newcommand*{\GlsXtrIfFieldEqXpStr}{%
1209 \ifstar\s@GlsXtrIfFieldEqXpStr\@GlsXtrIfFieldEqXpStr
1210 }

```

`rIfFieldEqXpStr`

```

1211 \newrobustcmd*{\@GlsXtrIfFieldEqXpStr}[5]{%
1212 \@glxtrifhasfield{#1}{#2}%
1213 {%
1214 \protected@edef\@glstmp{#3}%
1215 \ifdequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1216 }%
1217 {#5}%
1218 }

```

`rIfFieldEqXpStr`

```

1219 \newrobustcmd*{\s@GlsXtrIfFieldEqXpStr}[5]{%
1220 \s@glxtrifhasfield{#1}{#2}%
1221 {%
1222 \protected@edef\@glstmp{#3}%
1223 \ifdequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1224 }%
1225 {#5}%
1226 }

```

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

```

1227 \newcommand*{\GlsXtrIfXpFieldEqXpStr}{%
1228 \ifstar\s@GlsXtrIfXpFieldEqXpStr\@GlsXtrIfXpFieldEqXpStr
1229 }

```

`fXpFieldEqXpStr`

```

1230 \newrobustcmd*{\@GlsXtrIfXpFieldEqXpStr}[5]{%
1231 \@glxtrifhasfield{#1}{#2}%
1232 {%
1233 \protected@edef\@glstmp{\glscurrentfieldvalue}%
1234 \let\glscurrentfieldvalue\@glstmp
1235 \protected@edef\@glstmp{#3}%
1236 \ifdequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1237 }%

```

```

1238   {#5}%
1239 }

```

fXpFieldEqXpStr

```

1240 \newrobustcmd*{\s@GlsXtrIfXpFieldEqXpStr}[5]{%
1241   \s@glxtrifhasfield{#1}{#2}%
1242   {%
1243     \protected@edef\@glstmp{\glscurrentfieldvalue}%
1244     \let\glscurrentfieldvalue\@glstmp
1245     \protected@edef\@glstmp{#3}%
1246     \ifdefequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
1247   }%
1248   {#5}%
1249 }

```

sXtrForeignText

`\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`.

```

1250 \ifdef\foreignlanguage
1251 {
1252   \ifdef\GetTrackedDialectFromLanguageTag
1253   {
1254     \newcommand{\GlsXtrForeignText}[2]{%

```

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

```

1255     \let\@glxtr@org@currentfieldvalue\glscurrentfieldvalue
1256     \glxtrifhasfield{\GlsXtrForeignTextField}{#1}%
1257     {%
1258       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
1259       {\glscurrentfieldvalue}{\@glxtr@dialect}%
1260       \let\@glxtr@locale\glscurrentfieldvalue
1261       \let\glscurrentfieldvalue\@glxtr@org@currentfieldvalue
1262       \ifdefempty\@glxtr@dialect
1263       {%

```

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

```

1264         \ifundef\TrackedDialectClosestSubMatch
1265         {%
1266           \GlossariesExtraWarning{Can't obtain dialect label
1267             (tracklang v1.3.6+ required)}%
1268         }%
1269         {\let\@glxtr@dialect\TrackedDialectClosestSubMatch}%
1270       }%

```

```

1271      {}%
1272      \ifdefempty\@glstr@dialect
1273      {%
    No tracked dialect found for the root language.
1274      }%
1275      {%
    Check if there's a caption hook for the given dialect label.
1276      \ifcsundef{captions\@glstr@dialect}{}%
1277      {%
    Dialect label not recognised. Check if there's a known mapping.
1278      \IfTrackedDialectHasMapping{\@glstr@dialect}%
1279      {%
1280      \edef\@glstr@dialect{%
1281      \GetTrackedDialectToMapping{\@glstr@dialect}}%
    Does a caption hook exist for this?
1282      \ifcsundef{captions\@glstr@dialect}{}%
1283      {%
    No mapping. Try root language label instead.
1284      \ifcsundef{captions\@tracklang@lang}{}%
1285      {%
1286      \let\@glstr@dialect\@tracklang@lang
1287      }%
1288      }%
1289      }%
1290      {%
    No mapping. Try root language label instead.
1291      \ifcsundef{captions\@tracklang@lang}{}%
1292      {%
1293      \let\@glstr@dialect\@tracklang@lang
1294      }%
1295      }%
1296      }%
1297      }%
1298      \ifdefempty\@glstr@dialect
1299      {%
1300      \GlsXtrUnknownDialectWarning{\@glstr@locale}{\@tracklang@lang}%
1301      #2%
1302      }%
1303      {\foreignlanguage{\@glstr@dialect}{#2}}%
1304      }%
1305      {#2}% key not set
1306      }
1307      }
1308      {
1309      \newcommand{\GlsXtrForeignText}[2]{%
1310      \GlossariesExtraWarning{Can't encapsulate foreign text:

```

```

1311         tracklang v1.3.6+ required}%
1312     #2%
1313 }
1314 }
1315 }
1316 {
    \foreignlanguage isn't defined so just do <text>.
1317 \newcommand{\GlsXtrForeignText}[2]{#2}
1318 }

```

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

```

1319 \newcommand*{\GlsXtrForeignTextField}{userii}

```

`\nDialectWarning`

```

1320 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
1321 \GlossariesExtraWarning{Can't determine valid dialect label
1322 for locale '#1' (root language: #2)}}%
1323 }

```

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

```

1324 \ifdef\GlsEntryCounterLabelPrefix
1325 {%
1326 \newcommand*{\glsxtrpageref}[1]{%
1327 \ifglssentrycounter
1328 \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1329 \else
1330 \ifglssubentrycounter
1331 \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1332 \else
1333 \gls{#1}%
1334 \fi
1335 \fi
1336 }
1337 }%
1338 {%
1339 \newcommand*{\glsxtrpageref}[1]{%
1340 \ifglssentrycounter
1341 \pageref{glsentry-\glsdetoklabel{#1}}%
1342 \else
1343 \ifglssubentrycounter
1344 \pageref{glsentry-\glsdetoklabel{#1}}%
1345 \else
1346 \gls{#1}%
1347 \fi
1348 \fi
1349 }
1350 }%

```

lossarypreamble

```
1351 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1352   \ifcsdef{glolist@#1}%
1353   {%
1354     \ifcsundef{@glossarypreamble@#1}%
1355     {\csdef{@glossarypreamble@#1}{}}%
1356     {}%
1357     \csappto{@glossarypreamble@#1}{#2}%
1358   }%
1359   {%
1360     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1361   }%
1362 }
```

lossarypreamble

```
1363 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1364   \ifcsdef{glolist@#1}%
1365   {%
1366     \ifcsundef{@glossarypreamble@#1}%
1367     {\csdef{@glossarypreamble@#1}{}}%
1368     {}%
1369     \cspreto{@glossarypreamble@#1}{#2}%
1370   }%
1371   {%
1372     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1373   }%
1374 }
```

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.

The original `\@gls@entry@field` causes a problem for undefined entries when used in section headings or captions. Since entries must be defined with just the base package this isn't a significant issue, but it will cause a problem with `bib2gls` where no entries are defined on the first `LTEX` call, so redefine `\@gls@entry@field` to use `\csuse` instead of `\csname`.

`\@gls@entry@field`

`\@gls@entry@field{\langle label \rangle}{\langle field \rangle}`

This command was introduced to glossaries version 4.03 but older versions are likely to be incompatible with glossaries-extra.

```
1375 \ifdef\@gls@entry@field
1376 {
1377   \renewcommand*{\@gls@entry@field}[2]{\csuse{glo@\glsdetoklabel{#1}@#2}}
```

```
1378 }
1379 {}
```

`\ifglsused`

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

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

```
1380 \renewcommand*{\ifglsused}[3]{%
1381   \glsdoifexists{#1}{\ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}}%
1382 }
```

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 `\glsxtrpostlongdescription` instead.

`ewglossaryentry`

```
1383 \renewcommand*{\longnewglossaryentry}{%
1384   \@ifstar{\glsxtr@s@longnewglossaryentry\@glsxtr@longnewglossaryentry
1385 }
```

`ewglossaryentry` Starred version.

```
1386 \newcommand{\@glsxtr@s@longnewglossaryentry}[3]{%
1387   \glsdoifnoexists{#1}%
1388   {%
1389     \bgroup
1390     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1391     \long\def\@newglossaryentryprehook{%
1392       \long\def\@glo@desc{#3}%
1393       \@org@newglossaryentryprehook
1394     }%
1395     \renewcommand*{\gls@assign@desc}[1]{%
1396       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1397       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1398     }
1399     \gls@defglossaryentry{#1}{#2}%
1400   \egroup
1401   }%
1402 }
```

`ewglossaryentry` Unstarred version.

```
1403 \newcommand{\@glsxtr@longnewglossaryentry}[3]{%
1404   \glsdoifnoexists{#1}%
1405   {%
```



```

1406 \bgroup
1407 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1408 \long\def\@newglossaryentryprehook{%
1409 \long\def\@glo@desc{#3\glstrpostlongdescription}%
1410 \@org@newglossaryentryprehook
1411 }%
1412 \renewcommand*{\gls@assign@desc}[1]{%
1413 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

The following is different from the base glossaries.sty:

```

1414 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1415 }
1416 \gls@defglossaryentry{#1}{#2}%
1417 \egroup
1418 }%
1419 }

```

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

```

1420 \newcommand*{\glstrpostlongdescription}{\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.

```

1421 \renewcommand{\newignoredglossary}{%
1422 \@ifstar\glstr@s@newignoredglossary\glstr@org@newignoredglossary
1423 }

```

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

```

1424 \newcommand*{\glstr@org@newignoredglossary}[1]{%
1425 \ifcsdef{glolist@#1}
1426 {%
1427 \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1428 }%
1429 {%
1430 \ifdefempty\@ignored@glossaries
1431 {%
1432 \edef\@ignored@glossaries{#1}%
1433 }%
1434 {%
1435 \eappto\@ignored@glossaries{,#1}%
1436 }%
1437 \csgdef{glolist@#1}{,}%
1438 \ifcsundef{gls@#1@entryfmt}%
1439 {%
1440 \defglsentryfmt[#1]{\glsentryfmt}%
1441 }%
1442 }%
1443 \ifdefempty\@gls@nohyperlist

```

```

1444   {%
1445     \renewcommand*{\@gls@nohyperlist}{#1}%
1446   }%
1447   {%
1448     \eappto\@gls@nohyperlist{, #1}%
1449   }%
1450 }%
1451 }

```

`ignoredglossary` Starred form.

```

1452 \newcommand*{\glsxtr@s@newignoredglossary}[1]{%
1453   \ifcsdef{glolist@#1}
1454   {%
1455     \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
1456   }%
1457   {%
1458     \ifdefempty\@ignored@glossaries
1459     {%
1460       \edef\@ignored@glossaries{#1}%
1461     }%
1462     {%
1463       \eappto\@ignored@glossaries{, #1}%
1464     }%
1465     \csgdef{glolist@#1}{,}%
1466     \ifcsundef{gls@#1@entryfmt}%
1467     {%
1468       \defglsentryfmt[#1]{\glsentryfmt}%
1469     }%
1470     {}%
1471   }%
1472 }

```

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

```

1473 \glsifusetranslator
1474 {%
1475   \renewcommand*{\glssettoctitle}[1]{%
1476     \ifcsdef{gls@tr@set@#1@toctitle}%
1477     {%
1478       \csuse{gls@tr@set@#1@toctitle}%
1479     }%
1480     {%
1481       \ifcsdef{@glotype@#1@title}%
1482       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1483       {\def\glossarytoctitle{\glossarytitle}}%
1484     }%
1485   }%
1486 }
1487 {

```

```

1488 \renewcommand*{\glsettoctitle}[1]{%
1489   \ifcsdef{@glotype@#1@title}%
1490     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1491     {\def\glossarytoctitle{\glossarytitle}}%
1492 }
1493 }

```

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

```

1494 \newcommand{\provideignoredglossary}{%
1495   \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1496 }

```

ignoredglossary Unstarred version.

```

1497 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1498   \ifcsdef{glolist@#1}
1499   {}%
1500   {%
1501     \ifdefempty\@ignored@glossaries
1502     {%
1503       \edef\@ignored@glossaries{#1}%
1504     }%
1505     {%
1506       \eappto\@ignored@glossaries{, #1}%
1507     }%
1508     \csgdef{glolist@#1}{,}%
1509     \ifcsundef{gls@#1@entryfmt}%
1510     {%
1511       \def\glsentryfmt[#1]{\glsentryfmt}%
1512     }%
1513     {}%
1514     \ifdefempty\@gls@nohyperlist
1515     {%
1516       \renewcommand*{\@gls@nohyperlist}{#1}%
1517     }%
1518     {%
1519       \eappto\@gls@nohyperlist{, #1}%
1520     }%
1521   }%
1522 }

```

ignoredglossary Starred form.

```

1523 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1524   \ifcsdef{glolist@#1}
1525   {}%
1526   {%
1527     \ifdefempty\@ignored@glossaries
1528     {%
1529       \edef\@ignored@glossaries{#1}%
1530     }%

```

```

1531   {%
1532     \eappto\@ignored@glossaries{,#1}%
1533   }%
1534   \csgdef{glolist@#1}{,}%
1535   \ifcsundef{gls@#1@entryfmt}%
1536   {%
1537     \defglsentryfmt[#1]{\glsentryfmt}%
1538   }%
1539   {}%
1540 }%
1541 }

```

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

```

1542 \newcommand*\glsxtrcopytoglossary[2]{%
1543   \glsdoifexists{#1}%
1544   {%
1545     \ifcsdef{glolist@#2}
1546     {%
1547       \cseappto{glolist@#2}{#1,}%
1548     }%
1549     {%
1550       \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1551     }%
1552   }%
1553 }

```

1.3.1 Existence Checks

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

```

1554 \renewcommand*\glsdoifexists[2]{%
1555   \ifglsentryexists{#1}{#2}%
1556   {%

```

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

```

1557   \edef\glslabel{\glsdetoklabel{#1}}%
1558   \glsxtrundefaction{Glossary entry ‘\glslabel’
1559     has not been defined}{You need to define a glossary entry before
1560     you can reference it.}%
1561   }%
1562 }

```

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

```

1563 \renewcommand*\glsdoifnoexists[2]{%
1564   \ifglsentryexists{#1}{%
1565     \glsxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1566       has already been defined}{}}{#2}%
1567 }

```

`\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.

```

1568 \ifdef\glsdoifexistsordo
1569 {%
1570   \renewcommand{\glsdoifexistsordo}[3]{%
1571     \ifglstryexists{#1}{#2}%
1572     {%
1573       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1574       has not been defined}{You need to define a glossary entry
1575       before you can use it.}%
1576       #3%
1577     }%
1578   }%
1579 }
1580 {%
1581   \glstr@warnonexistsordo\glsdoifexistsordo
1582   \newcommand{\glsdoifexistsordo}[3]{%
1583     \ifglstryexists{#1}{#2}%
1584     {%
1585       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1586       has not been defined}{You need to define a glossary entry
1587       before you can use it.}%
1588       #3%
1589     }%
1590   }%
1591 }

```

`\glsarynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```

1592 \ifdef\doifglossarynoexistsordo
1593 {%
1594   \renewcommand{\doifglossarynoexistsordo}[3]{%
1595     \ifglossaryexists*{#1}%
1596     {%
1597       \glstrundefaction{Glossary type '#1' already exists}{}%
1598       #3%
1599     }%
1600     {#2}%
1601   }%
1602 }
1603 {%
1604   \glstr@warnonexistsordo\doifglossarynoexistsordo
1605   \newcommand{\doifglossarynoexistsordo}[3]{%
1606     \ifglossaryexists*{#1}%
1607     {%
1608       \glstrundefaction{Glossary type '#1' already exists}{}%
1609       #3%
1610     }%
1611     {#2}%
1612   }%

```

```

1613 }
1614

```

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).

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

```

1615 \appto\@newglossaryentryposthook{%
1616   \ifdefvoid\@glo@see
1617     {\csxdef{glo@\@glo@label @see}{}}%
1618     {%
1619       \csxdef{glo@\@glo@label @see}{\@glo@see}%
1620       \ifglstr@autoseeindex
1621         \@glstr@autoindexcrossrefs
1622       \fi
1623     }%
1624 }
1625 \appto\@gls@keymap{,{see}{see}}

```

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

```

1626 \newcommand*{\glstrusee}[1]{%
1627   \glsdoifexists{#1}%
1628   {%
1629     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1630     \ifdefempty\@glo@see
1631       {}%
1632     {%
1633       \expandafter\glstr@usesee\@glo@see\end@glstr@usesee
1634     }%
1635   }%
1636 }

```

`\glstr@usesee`

```

1637 \newcommand*{\glstr@usesee}[1][\seename]{%
1638   \@glstr@usesee{#1}%
1639 }

```

`\@glstr@usesee`

```

1640 \def\@glstr@usesee[#1]#2\end@glstr@usesee{%
1641   \glstruseeformat{#1}{#2}%
1642 }

```

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

```

1643 \newcommand*{\glstruseeformat}[2]{%
1644   \glsseeformat{#1}{#2}{}%
1645 }

```

`\lsseeitemformat` glossaries originally defined `\glseeitemformat` to use `\glentryname` but in v3.0 this was switched to use `\glentrytext` due to problems occurring with the name field being sanitized. Since this is no longer a problem, glossaries-extra restored the original definition as it makes more sense to use the name in the cross-reference list. Unfortunately this doesn't take style changes into account, so as from v1.42, this now uses `\glsfmttext` and `\glsfmtname` instead. (The text field is chosen rather than the short field to allow for the “noshort” styles.)

```
1646 \renewcommand*{\glseeitemformat}[1]{%
1647   \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#1}}%
1648 }
```

`\glxtrhiername`

`\glxtrhiername{<label>}`

Displays the hierarchical name for the given entry. The cross-reference format `\glseeitemformat` may be redefined to use this command to show the hierarchy, if required. This now uses `\glsfmttext` and `\glsfmtname` instead of `\glsaccessshort` and `\glsaccessname` to allow for style formatting.

```
1649 \newcommand*{\glxtrhiername}[1]{%
1650   \glsdoifexists{#1}%
1651   {%
1652     \glxtrifhasfield{parent}{#1}%
1653     {\glxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1654     }%
1655     \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#1}}%
1656   }%
1657 }
```

`\Glsxtrhiername`

`\Glsxtrhiername{<label>}`

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

```
1658 \newcommand*{\Glsxtrhiername}[1]{%
1659   \glsdoifexists{#1}%
1660   {%
1661     \glxtrifhasfield{parent}{#1}%
1662     {%
1663       \Glsxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1664       \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#1}}%
1665     }%
1666     {\ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#1}}}%
1667   }%
1668 }
```

\GlsXtrhiername

`\GlsXtrhiername{<label>}`

As above but converts the first letter of each name to a capital. (Note that this isn't applying title case, just capitalising the start of each hierarchical element.)

```
1669 \newcommand*{\GlsXtrhiername}[1]{%
1670   \glsdoifexists{#1}%
1671   {%
1672     \glstrifhasfield{parent}{#1}%
1673     {\GlsXtrhiername{\glscurrentfieldvalue}\glstrhiernamesep}%
1674     }%
1675     \ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#1}}%
1676   }%
1677 }
```

\GLSxtrhiername

`\GLSxtrhiername{<label>}`

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

```
1678 \newcommand*{\GLSxtrhiername}[1]{%
1679   \glsdoifexists{#1}%
1680   {%
1681     \glstrifhasfield{parent}{#1}%
1682     {%
1683       \GLSxtrhiername{\glscurrentfieldvalue}\glstrhiernamesep
1684       \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#1}}%
1685     }%
1686     {\ifglshasshort{#1}{\GLSfmttext{#1}}{\GLSfmtname{#1}}}%
1687   }%
1688 }
```

\GLSXTRhiername

`\GLSXTRhiername{<label>}`

As above but displays all names in all-caps.

```
1689 \newcommand*{\GLSXTRhiername}[1]{%
1690   \glsdoifexists{#1}%
1691   {%
1692     \glstrifhasfield{parent}{#1}%
1693     {\GLSXTRhiername{\glscurrentfieldvalue}\glstrhiernamesep}%
1694     {}
1695     \ifglshasshort{#1}{\GLSfmttext{#1}}{\GLSfmtname{#1}}%
1696   }
```



```

1696 }%
1697 }

```

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

```

1698 \newcommand*{\glxtrhiernamesep}{\,\{\small$\triangleright$\}\,}

```

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

```

1699 \newcommand*{\glxtruseseealso}[1]{%
1700   \glsdoidexists{#1}%
1701   {%
1702     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
1703     \ifdefempty\@glo@see
1704       {}%
1705       {%
1706         \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
1707       }%
1708     }%
1709 }

```

`\glxtrusealias` Apply `\glssseeformat` to the `alias` key if not empty. There's no optional tag to worry about here. The value also isn't a comma-separated list, but use the same interface.

```

1710 \newcommand*{\glxtrusealias}[1]{%
1711   \glsdoidexists{#1}%
1712   {%
1713     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@alias}%
1714     \ifdefempty\@glo@see
1715       {}%
1716       {%

```

Expansion isn't necessary because the value is a single label not a list.

```

1717     \glxtruseseeformat{\seename}{\@glo@see}%
1718   }%
1719 }%
1720 }

```

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

```

1721 \newcommand*{\glxtruseseealsoformat}[1]{%
1722   \glssseeformat[\seealso]{#1}%
1723 }

```

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

```

1724 \newrobustcmd{\glxtrseelist}[1]{%
1725   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1726 }

```

`\seealso` In case this command hasn't been defined. Languages packages actually provide `\also` so use that if it's defined.

```
1727 \ifdef\also
1728 {\providecommand{\seealso}{\also}}
1729 {\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.

```
1730 \ifdef\@xdycrossrefhook
1731 {
```

Add the cross-reference class definition to the hook.

```
1732 \appto\@xdycrossrefhook{%
1733 \write\glswrite{(define-crossref-class \string"seealso\string"
1734 :unverified )}%
1735 \write\glswrite{(markup-crossref-list
1736 :class \string"seealso\string"^^J\space\space\space
1737 :open \string"\string\glsxtruseealsoformat\glsopenbrace\string"
1738 :close \string"\glsclosebrace\string")}%
1739 }
```

Append to class list.

```
1740 \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

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

```
1741 \newrobustcmd*{\glsxtrindexseealso}[2]{%
1742 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
1743 \@glsxtr@recordsee{#1}{#2}%
1744 \fi
1745 \glsdoifexists{#1}%
1746 {%
1747 \@glsxtrwrglossmark
1748 \def\@gls@xref{#2}%
1749 \@onelevel@sanitize\@gls@xref
1750 \@gls@checkmkidxchars\@gls@xref
1751 \gls@glossary{\csname glo@#1@type\endcsname}{%
1752 (indexentry
1753 :tkey (\csname glo@#1@index\endcsname)
1754 :xref (\string"\@gls@xref\string")
1755 :attr \string"seealso\string"
1756 )
1757 }%
1758 }%
1759 }
1760 }
1761 {
```

`xindy` not in use or `glossaries` version too old to support this.

```

1762 \newrobustcmd*{\glxtrindexseealso}{\glssee[\seealsoname]}
1763 }

```

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`.

```

1764 \ifdef\gls@set@xr@key
1765 {

```

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

```

1766 \define@key{glossentry}{alias}{%
1767   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1768 }
1769 \define@key{glossentry}{seealso}{%
1770   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1771 }

```

Add to the key mappings.

```

1772 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1773 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1774 \appto\@newglossaryentryposthook{%
1775   \ifdefvoid\@glo@seealso
1776     {\csxdef{glo@\@glo@label @seealso}{}}%
1777     {%
1778       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1779       \ifglxtr@autoseeindex
1780         \@glxtr@autoindexcrossrefs
1781       \fi
1782     }%

```

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

```

1783   \ifdefvoid\@glo@alias
1784     {\csxdef{glo@\@glo@label @alias}{}}%
1785     {%
1786       \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1787     }%
1788 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

1789 \newcommand*{\glxtralias}[1]{\@gls@entry@field{#1}{alias}}

```

trseealsolabels

```
1790 \newcommand*{\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}
```

Add to the \@glo@autosee hook.

```
1791 \appto\@glo@autoseehook{%
1792   \ifdefvoid\@glo@alias
1793   {%
1794     \ifdefvoid\@glo@seealso
1795     }%
1796     {%
1797       \edef\@do@glsssee{\noexpand\glxtrindexseealso
1798         {\@glo@label}{\@glo@seealso}}%
1799       \@do@glsssee
1800     }%
1801   }%
1802 }
```

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

```
1803   \ifdefvoid\@glo@see
1804   {%
1805     \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1806     \@do@glsssee
1807   }%
1808   }%
1809 }%
1810 }%
1811 }
1812 {
```

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

\glxtralias

```
1813 \glsaddstoragekey*{alias}{}{\glxtralias}
```

trseealsolabels

```
1814 \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.

```
1815 \appto\@newglossaryentryposthook{%
1816   \ifcsvoid{glo@\@glo@label @alias}%
1817   {%
1818     \ifcsvoid{glo@\@glo@label @seealso}%
1819     }%
1820     {%
1821       \edef\@do@glsssee{\noexpand\glxtrindexseealso
1822         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1823       \@do@glsssee

```

```

1824     }%
1825     }%
1826     {%

```

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

```

1827     \ifdefvoid\@glo@see
1828     {%
1829         \edef\@do@glsssee{\noexpand\glsssee
1830             {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1831         \@do@glsssee
1832     }%
1833     }%
1834 }%
1835 }

1836 }

```

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

```

1837 \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.

```

1838 \newcommand*\@glxtraddallcrossrefs{%
1839     \forallglossaries{\@glo@type}%
1840     {%
1841         \forglsentries[\@glo@type]{\@glo@label}%
1842         {%
1843             \ifglssused{\@glo@label}%
1844             {\expandafter\@glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1845         }%
1846     }%
1847 }

```

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

```

1848 \newcommand*\@glxtr@addunusedxrefs[1]{%
1849     \letcs{\@glo@see}{glo@glstetoklabel{#1}@see}%
1850     \ifdefvoid\@glo@see
1851     {%
1852     {%
1853         \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1854     }%
1855     \letcs{\@glo@see}{glo@glstetoklabel{#1}@seealso}%
1856     \ifdefvoid\@glo@see
1857     {%
1858     {%
1859         \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1860     }%
1861     }%
1862     }%
1863 }

```

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

```
1862 \newcommand*{\glxtr@addunused}[1][]{%
1863   \glxtr@addunused
1864 }
```

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

```
1865 \def\glxtr@addunused#1\endglxtr@addunused{%
1866   \for\glxtr@label:=#1\do
1867   {%
1868     \ifglxsused{\glxtr@label}{}%
1869     {%
1870       \glsadd[format=glxtrunusedformat]{\glxtr@label}%
1871       \glsunset{\glxtr@label}%
1872       \expandafter\glxtr@addunusedxrefs\expandafter{\glxtr@label}%
1873     }%
1874   }%
1875 }
```

xtrunusedformat

```
1876 \newcommand*{\glxtrunusedformat}[1]{\unskip}
```

1.3.2 Document Definitions

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

```
1877 \ifdef\gls@begindocdefs
1878 {%
1879   \renewcommand*{\gls@begindocdefs}{%
1880     \ifnum\glxtr@docdefval=1\relax
1881       \gls@enablesavenonumberlist
1882       \edef\gls@restoreat{%
1883         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1884       \makeatletter
1885       \InputIfFileExists{\jobname.glsdefs}{\relax}%
1886       \gls@restoreat
1887       \undef\gls@restoreat
1888       \gls@defdocnewglossaryentry
1889     \else
1890       \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.

```
1891       \gls@enablesavenonumberlist
1892       \let\gls@checkseeallowed\relax
1893       \let\newglossaryentry\new@atomglossaryentry
1894       \global\newwrite\gls@deffile
1895       \immediate\openout\gls@deffile=\jobname.glsdefs
```

Write all currently defined entries.

```

1896      \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
1897      \fi
1898  \fi
1899  }
1900 }
1901 {%
1902  \ifnum\@glsxtr@docdefval=3\relax
1903    \PackageError{glossaries-extra}{Package option
1904      'docdef=\@glsxtr@docdefsetting' requires at least version 4.37
1905      of the base glossaries.sty package}{}
1906  \fi
1907 }

```

m@glossaryentry

```

1908 \newrobustcmd{\new@atom@glossaryentry}[2]{%
1909   \gls@defglossaryentry{#1}{#2}%
1910   \@gls@writedef{#1}%
1911 }

```

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.

```

1912 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1913 \renewcommand{\makenoidxglossaries}{%
1914   \@domakeglossaries
1915   {%
1916     \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1917     {%
1918       \glsxtr@orgmakenoidxglossaries

```

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

```

1919   \renewcommand{\do@seeglossary}[2]{%
1920     \@glsxtrwrglossmark
1921     \edef\@gls@label{\glsdetoklabel{##1}}%
1922     \protected@write\@auxout{}{%
1923       \string\@gls@reference
1924       {\csname glo@\@gls@label @type\endcsname}%
1925       {\@gls@label}%
1926       {%
1927         \string\glsseeformat##2}%
1928       }%
1929     }%
1930   }%

```

Check for docdefs=restricted:

```

1931   \if@glsxtrdocdefrestricted

```

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

```

1932 \renewcommand*{\@gls@reference}[3]{%
1933 \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
1934 \ifinlistcs{##2}{@glsref@##1}%
1935 {}%
1936 {\listcsgadd{@glsref@##1}{##2}}%
1937 \ifcsundef{glo@glstdetoklabel{##2}@loclist}%
1938 {\csgdef{glo@glstdetoklabel{##2}@loclist}{}}%
1939 {}%
1940 \listcsgadd{glo@glstdetoklabel{##2}@loclist}{##3}%
1941 }%
1942 \else
    Disable document definitions.
1943 \@glxtrdocdeffalse
1944 \fi
1945 \disable@keys{glossaries-extra}{docdef}%
1946 }%
1947 {%
1948 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1949 not permitted\MessageBreak
1950 with record=\@glxtr@record@setting\space package option}%
1951 {You may only use \string\makenoidxglossaries\space with the
1952 record=off option}%
1953 }%
1954 }%
1955 }

```

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

```

1956 \renewcommand*{\gls@defdocnewglossaryentry}{%
1957 \ifcase\@glxtr@docdefval
    docdef=false:
1958 \renewcommand*{\newglossaryentry}[2]{%
1959 \PackageError{glossaries-extra}{Glossary entries must
1960 be \MessageBreak defined in the preamble with \MessageBreak
1961 package option 'docdef=false'\MessageBreak(consider using
1962 'docdef=restricted')}{Move your glossary definitions to
1963 the preamble. You can also put them in a \MessageBreak separate file
1964 and load them with \string\loadglsentries.}%
1965 }%
1966 \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.

```

1967 \let\gls@checkseeallowed\relax
1968 \let\newglossaryentry\newglossaryentry
1969 \else

```

Restricted mode just needs to allow the `see` value.

```

1970 \let\gls@checkseeallowed\relax
1971 \fi

```


1972 }%

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.

`\GlsXtrEnableOnTheFly`

```
1973 \newcommand*{\GlsXtrEnableOnTheFly}{%
1974   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1975 }
```

`\rGlsXtrEnableOnTheFly`

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.)

```
1976 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1977   \renewcommand*{\glsdetoklabel}[1]{%
1978     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1979     {%
1980       \expandafter\detokenize\expandafter{##1}%
1981     }%
1982     {\detokenize{##1}}%
1983   }%
1984   \@GlsXtrEnableOnTheFly
1985 }
1986 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1987   \expandafter\if\glsbackslash#1%
1988     #3%
1989   \else
1990     #4%
1991   \fi
1992 }
```

`\GlsXtrStarflywarn`

```
1993 \newcommand*{\GlsXtrStarflywarn}{%
1994   \GlossariesExtraWarning{Experimental starred version of
1995   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1996   read the warnings in the glossaries-extra user manual)}}%
1997 }
```

`\rGlsXtrEnableOnTheFly`

```
1998 \newcommand*{\@GlsXtrEnableOnTheFly}{%
```

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

These definitions are all assigned the category given by:

`\GlsXtrcat`

```
1999   \newcommand*{\GlsXtrcat}{general}
```

```

\glsxtr
2000 \newcommand*\glsxtr[1] [] {%
2001 \def\glsxtr@keylist{##1}%
2002 \@glsxtr
2003 }

\@glsxtr
2004 \newcommand*\@glsxtr[2] [] {%
2005 \ifglsentryexists{##2}%
2006 {%
2007 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2008 }%
2009 {%
2010 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2011 description={\nopostdesc},##1}%
2012 }%
2013 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
2014 }

\Glsxtr
2015 \newcommand*\Glsxtr[1] [] {%
2016 \def\glsxtr@keylist{##1}%
2017 \@Glsxtr
2018 }

\@Glsxtr
2019 \newcommand*\@Glsxtr[2] [] {%
2020 \ifglsentryexists{##2}%
2021 {%
2022 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2023 }%
2024 {%
2025 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2026 description={\nopostdesc},##1}%
2027 }%
2028 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
2029 }

\glsxtrpl
2030 \newcommand*\glsxtrpl[1] [] {%
2031 \def\glsxtr@keylist{##1}%
2032 \@glsxtrpl
2033 }

\@glsxtrpl
2034 \newcommand*\@glsxtrpl[2] [] {%
2035 \ifglsentryexists{##2}%
2036 {%

```

```

2037     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2038 }%
2039 {%
2040     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2041     description={\nopostdesc},##1}%
2042 }%
2043 \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
2044 }

```

\Glsxtrpl

```

2045 \newcommand*\Glsxtrpl[1][1]{%
2046 \def\glsxtr@keylist{##1}%
2047 \@Glsxtrpl
2048 }

```

\@Glsxtrpl

```

2049 \newcommand*\@Glsxtrpl[2][1]{%
2050 \ifglsentryexists{##2}
2051 {%
2052     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2053 }%
2054 {%
2055     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2056     description={\nopostdesc},##1}%
2057 }%
2058 \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
2059 }

```

\GlsXtrWarning

```

2060 \newcommand*\GlsXtrWarning[2]{%
2061 \def\glsxtr@optlist{##1}%
2062 \@onelevel@sanitize\glsxtr@optlist
2063 \GlossariesExtraWarning{The options '\glsxtr@optlist' have
2064 been ignored for entry '##2' as it has already been defined}%
2065 }

```

Disable commands after the glossary:

```

2066 \renewcommand\@printglossary[2]{%
2067 \def\glsxtr@printglossopts{##1}%
2068 \@glsxtr@orgprintglossary{##1}{##2}%
2069 \def\glsxtr{\glsxtr@disabledflycommand\glsxtr}%
2070 \def\glsxtrpl{\glsxtr@disabledflycommand\glsxtrpl}%
2071 \def\Glsxtr{\glsxtr@disabledflycommand\Glsxtr}%
2072 \def\Glsxtrpl{\glsxtr@disabledflycommand\Glsxtrpl}%
2073 }

```

abledflycommand

```

2074 \newcommand*\@glsxtr@disabledflycommand[1]{%
2075 \PackageError{glossaries-extra}%

```

```

2076   {\string##1\space can't be used after any of the \MessageBreak
2077     glossaries have been displayed}%
2078   {The on-the-fly commands enabled by
2079     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
2080     before the glossaries. If you want to use any entries \MessageBreak
2081     after any of the glossaries, you must use the standard \MessageBreak
2082     method of first defining the entry and then using the \MessageBreak
2083     entry with commands like \string\gls}%
2084     @@glxtr@disabledflycommand
2085   }%
2086   \newcommand*{@@glxtr@disabledflycommand}[2][\{##2}

```

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

```

2087   \let\GlsXtrEnableOnTheFly\relax
2088 }
2089 \@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.

```

2090 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

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

etglossarystyle

```

2091 \renewcommand*{\setglossarystyle}[1]{%
2092   \ifcsundef{@glsstyle@#1}%
2093   {%
2094     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
2095   }%
2096   {%
2097     \csname @glsstyle@#1\endcsname
2098     \protected@edef\@glxtr@current@style{#1}%
2099   }%
2100   \ifx\@glossary@default@style\relax
2101     \protected@edef\@glossary@default@style{#1}%
2102   \fi
2103 }

```

In case we have an old version of glossaries:

```

2104 \ifdef\@glossary@default@style
2105 {}
2106 {%
2107   \let\@glossary@default@style\relax
2108 }

```

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

```
2109 \ifdef\glslistdottedwidth
2110 {%
2111   \ifdim\glslistdottedwidth=.5\hsize
2112     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
2113     \AtBeginDocument{%
2114       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
2115         \setlength{\glslistdottedwidth}{.5\columnwidth}%
2116       \fi
2117     }%
2118   \fi
2119 }
2120 {}%
```

Similarly for \glsdescwidth:

\glsdescwidth

```
2121 \ifdef\glsdescwidth
2122 {%
2123   \ifdim\glsdescwidth=.6\hsize
2124     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
2125     \AtBeginDocument{%
2126       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
2127         \setlength{\glsdescwidth}{.6\columnwidth}%
2128       \fi
2129     }%
2130   \fi
2131 }
2132 {}%
```

and for \glspagelistwidth:

lspagelistwidth

```
2133 \ifdef\glspagelistwidth
2134 {%
2135   \ifdim\glspagelistwidth=.1\hsize
2136     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
2137     \AtBeginDocument{%
2138       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
2139         \setlength{\glspagelistwidth}{.1\columnwidth}%
2140       \fi
2141     }%
2142   \fi
2143 }
2144 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```
2145 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
```

```

2146 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
2147   \glsnonnumberlistfalse
2148   \renewcommand*{\glossaryentrynumbers}[1]{%
2149     \ifglsentryexists{\glscurrententrylabel}%
2150     {%
2151       \@glsxtrpreloctag
2152       \GlsXtrFormatLocationList{#1}%
2153       \@glsxtrpostloctag
2154       \gls@save@numberlist{#1}%
2155     }{}%
2156   }%
2157 \else
2158   \glsnonnumberlisttrue
2159   \renewcommand*{\glossaryentrynumbers}[1]{%
2160     \ifglsentryexists{\glscurrententrylabel}%
2161     {%
2162       \gls@save@numberlist{#1}%
2163     }{}%
2164   }%
2165 \fi

```

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

```

2166 \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

```

2167 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
2168   \let\@glsxtrpreloctag\@glsxtrpreloctag
2169   \let\@glsxtrpostloctag\@glsxtrpostloctag
2170   \renewcommand*{\@glsxtr@pagetag}{#1}%
2171   \renewcommand*{\@glsxtr@pagestag}{#2}%
2172   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
2173     \csgdef{\@glsxtr@preloctag@##1}{##2}%
2174   }%
2175   \renewcommand*{\@glsxtr@doloctag}{%
2176     \ifcsundef{\@glsxtr@preloctag@\glscurrententrylabel}%
2177     {%
2178       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
2179         Rerun required}%
2180     }%
2181     {%
2182       \csuse{\@glsxtr@preloctag@\glscurrententrylabel}%
2183     }%

```

```

2184 }%
2185 }
2186 \@onlypreamble\GlsXtrEnablePreLocationTag

```

glsxtrpreloctag

```

2187 \newcommand*{\@glsxtrpreloctag}{%
2188   \let\@glsxtr@org@delimN\delimN
2189   \let\@glsxtr@org@delimR\delimR
2190   \let\@glsxtr@org@glsignore\glsignore
   \gdef is required as the delimiters may occur inside a scope.
2191   \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
2192   \renewcommand*{\delimN}{%
2193     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
2194     \@glsxtr@org@delimN}%
2195   \renewcommand*{\delimR}{%
2196     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
2197     \@glsxtr@org@delimR}%
2198   \renewcommand*{\glsignore}[1]{%
2199     \gdef\@glsxtr@thisloctag{\relax}%
2200     \@glsxtr@org@glsignore{##1}}%
2201   \@glsxtr@doloctag
2202 }

```

glsxtrpreloctag

```

2203 \newcommand*{\@glsxtrpreloctag}{}

```

@glsxtr@pagetag

```

2204 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

2205 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

2206 \newcommand*{\@glsxtrpostloctag}{%
2207   \let\delimN\@glsxtr@org@delimN
2208   \let\delimR\@glsxtr@org@delimR
2209   \let\glsignore\@glsxtr@org@glsignore
2210   \protected@write\@auxout{%
2211     {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
2212 }

```

lsxtrpostloctag

```

2213 \newcommand*{\@glsxtrpostloctag}{}

```

lsxtr@preloctag

```

2214 \newcommand*{\@glsxtr@savepreloctag}[2]{}
2215 \protected@write\@auxout{%
2216   \string\providecommand\string\@glsxtr@savepreloctag[2]{}

```

glsxtr@doloctag

```
2217 \newcommand*{\@glsxtr@doloctag}{}
```

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

```
2218 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
2219   \XKV@plfalse
2220   \XKV@sttrue
2221   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
2222   {%
2223     \csname glsnonumberlist\XKV@resa\endcsname
2224     \ifglsnonumberlist
2225       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
2226     \else
2227       \def\glossaryentrynumbers##1{%
2228         \@glsxtrpreloctag
2229         \GlsXtrFormatLocationList{##1}%
2230         \@glsxtrpostloctag
2231         \gls@save@numberlist{##1}}%
2232     \fi
2233   }%
2234 }
```

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.

```
2235 \renewcommand*{\glsentryfmt}{%
2236   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}%
2237   \glsifregular{\glslabel}%
2238   {\glsxtrregularfont{\glsgenentryfmt}}%
2239   {%
2240     \ifglshasshort{\glslabel}%
2241     {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
2242     {\glsxtrregularfont{\glsgenentryfmt}}%
2243   }%
2244 }
```

sxtrregularfont Font used for regular entries.

```
2245 \newcommand*{\glsxtrregularfont}[1]{#1}
```

bbreviationfont Font used for abbreviation entries.

```
2246 \newcommand*{\glsxtrabbreviationfont}[1]{#1}
```


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

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

```
2247 \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).

```
2248 \@glxtr@record{#2}{#3}{glslink}%
2249 \glsdoifexists{#3}%
2250 {%
```

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).

```
2251 \let\glxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
2252 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2253 \def\glscustomtext{#4}%
2254 \@glxtr@field@linkdefs
2255 #1%
2256 \@gls@link[#2]{#3}{#4}%
2257 \let\ifKV@glslink@hyper\glxtrorg@ifKV@glslink@hyper
2258 }%
2259 \glspostlinkhook
2260 }
```

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

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

```
2261 \let\@glxtr@org@gls@\@gls@
2262 \def\@gls@#1#2{%
2263 \@glxtr@record{#1}{#2}{glslink}%
2264 \@glxtr@org@gls@{#1}{#2}%
2265 }%
```

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

```
2266 \let\@glxtr@org@glspl@\@glspl@
2267 \def\@glspl@#1#2{%
2268 \@glxtr@record{#1}{#2}{glslink}%
2269 \@glxtr@org@glspl@{#1}{#2}%
2270 }%
```

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

```
2271 \let\@glxtr@org@Gls@\@Gls@
2272 \def\@Gls@#1#2{%
```

```

2273 \@glxtr@record{#1}{#2}{glslink}%
2274 \@glxtr@org@Gls@{#1}{#2}%
2275 }%

```

\@Glspl@ Save the original definition and redefine.

```

2276 \let\@glxtr@org@Glspl@\@Glspl@
2277 \def\@Glspl@#1#2{%
2278 \@glxtr@record{#1}{#2}{glslink}%
2279 \@glxtr@org@Glspl@{#1}{#2}%
2280 }%

```

\@GLS@ Save the original definition and redefine.

```

2281 \let\@glxtr@org@GLS@\@GLS@
2282 \def\@GLS@#1#2{%
2283 \@glxtr@record{#1}{#2}{glslink}%
2284 \@glxtr@org@GLS@{#1}{#2}%
2285 }%

```

\@GLSpl@ Save the original definition and redefine.

```

2286 \let\@glxtr@org@GLSpl@\@GLSpl@
2287 \def\@GLSpl@#1#2{%
2288 \@glxtr@record{#1}{#2}{glslink}%
2289 \@glxtr@org@GLSpl@{#1}{#2}%
2290 }%

```

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

```

2291 \renewcommand*{\@glsdisp}[3][{}]{%
2292 \@glxtr@record{#1}{#2}{glslink}%
2293 \glsdoifexists{#2}{%
2294 \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
2295 \let\glsifplural\@secondoftwo
2296 \let\glscapscase\@firstofthree
2297 \def\glscustomtext{#3}%
2298 \def\glsinsert{}%
2299 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
2300 \@gl@link[#1]{#2}{\@glo@text}%
2301 \ifKV@glslink@local
2302 \glslocalunset{#2}%
2303 \else
2304 \glsunset{#2}%
2305 \fi
2306 }%
2307 \glspostlinkhook
2308 }

```

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

```

2309 \renewcommand*{\@gls@link}[3][{}]{%

```

```

2310 \@glxstr@record{#1}{#2}{glslink}%
2311 \glsdoifexistsordo{#2}%
2312 {%
2313 \let\do@gls@link@checkfirsthyper\relax

```

Post-link hook commands need initialising.

```

2314 \def\glscustomtext{#3}%
2315 \@glxstr@field@linkdefs
2316 \@gls@link[#1]{#2}{#3}%
2317 }%
2318 {%
2319 \glstextformat{#3}%
2320 }%
2321 \glspostlinkhook
2322 }

```

glxstrinitwrgloss Set the default if the wrgloss is omitted.

```

2323 \newcommand*{\glxstrinitwrgloss}{%
2324 \glsifattribute{\glslabel}{wrgloss}{after}%
2325 {%
2326 \glxstrinitwrglossbeforefalse
2327 }%
2328 {%
2329 \glxstrinitwrglossbeforetrue
2330 }%
2331 }

```

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

```

2332 \newif\ifglxstrinitwrglossbefore
2333 \glxstrinitwrglossbeforetrue

```

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

```

2334 \define@choicekey{glslink}{wrgloss}%
2335 [\@glxstr@wrglossval\@glxstr@wrglossnr]%
2336 {before,after}%
2337 {%
2338 \ifcase\@glxstr@wrglossnr\relax
2339 \glxstrinitwrglossbeforetrue
2340 \or
2341 \glxstrinitwrglossbeforefalse
2342 \fi
2343 }

2344 \define@key{glslink}{thevalue}{\def\@glxstr@thevalue{#1}}

2345 \define@key{glslink}{theHvalue}{\def\@glxstr@theHvalue{#1}}

```

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

```

2346 \define@boolkey{glslink}[glxstr@]{hyperoutside}[true]{}
2347 \glxstr@hyperoutsidettrue

```

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

```

2348 \define@key{glslink}{textformat}{%
2349   \ifcsdef{#1}
2350   {%
2351     \letcs{\@glsxtr@local@textformat}{#1}%
2352   }%
2353   {%
2354     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2355   }%
2356 }

2357 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

```

nithyperoutside Set the default if the hyperoutside is omitted.

```

2358 \newcommand*{\glsxtrinithyperoutside}{%
2359   \glsifattribute{glslabel}{hyperoutside}{false}%
2360   {%
2361     \glsxtr@hyperoutsidedefalse
2362   }%
2363   {%
2364     \glsxtr@hyperoutsidettrue
2365   }%
2366 }

```

r@inc@linkcount Does nothing by default.

```

2367 \newcommand*{\glsxtr@inc@linkcount}{}

```

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

```

2368 \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.

```

2369 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2370   \protected@edef\@glsxtr@tmp{#2}%
2371   \expandafter#1\expandafter{\@glsxtr@tmp}%
2372 }

```

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

```

2373 \newcommand*{\@glsxtr@use@equation@counter}{%
2374   \@glsxtr@ifnum@mmode{\def\@gls@counter{equation}}{}%
2375 }

```

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

```

2376 \newcommand*{\glsxtr@do@autoadd}[1]{}

```

$\backslash\text{GlsXtrAutoAddOnFormat}[\langle label \rangle][\langle format list \rangle][\langle glsadd options \rangle]$

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

```

2377 \newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
2378   \renewcommand*{\glstr@do@autoadd}[1]{%
2379     \begingroup
2380       \protected@edef\glstr@do@autoadd{%
2381         \noexpand\ifstrequal{##1}{\glslink}%
2382         {%
2383           \noexpand\DTLifinlist{\@glsnumberformat}{#2}{\noexpand\glsadd[format={\@glsnumberfor
2384           }%
2385           }%
2386         }%
2387       \glstr@do@autoadd
2388     \endgroup
2389   }%
2390 }
```

$\backslash\text{gls@link}$ 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.

```

2391 \def\@gls@link[#1]#2#3{%
2392   \leavevmode
2393   \edef\glslabel{\glsdetoklabel{#2}}%
2394   \def\@gls@link@opts{#1}%
2395   \let\@gls@link@label\glslabel
2396   \let\@glsnumberformat\glstr@defaultnumberformat
2397   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2398   \edef\glstype{\csname glo@\glslabel @type\endcsname}%
2399   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Save current value of $\backslash\text{glolinkprefix}$:

```
2400 \let\@glstr@org@glolinkprefix\glolinkprefix
```

Initialise $\backslash\text{glstr@local@textformat}$

```
2401 \let\@glstr@local@textformat\relax
```

Initialise thevalue and theHvalue (v1.19).

```

2402 \def\@glstr@thevalue{%
2403   \def\@glstr@theHvalue{\@glstr@thevalue}%
```

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

```
2404 \glstr@initwrgloss
```

Initialise whether $\backslash\text{hyperlink}$ should be outside $\backslash\text{glstextformat}$ (new to v1.21).

```
2405 \glstr@inithyperoutside
```

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

```
2406 \@gls@setdefault@glslink@opts
```

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

```
2407 \glxtr@inc@linkcount
```

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

```
2408 \if@glxtr@equations
```

```
2409 \@glxtr@use@equation@counter
```

```
2410 \fi
```

As the original definition.

```
2411 \do@gl:disablehyperinlist
```

```
2412 \do@gl@link@checkfirsthyper
```

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

```
2413 \glslinkpresetkeys
```

Set options.

```
2414 \setkeys{glslink}{#1}%
```

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

```
2415 \glxtr@do@autoadd{glslink}%
```

User hook after options are set:

```
2416 \glslinkpostsetkeys
```

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

```
2417 \ifdefempty{\@glxtr@thevalue}%
```

```
2418 {%
```

```
2419 \@gls@saveentrycounter
```

```
2420 }%
```

```
2421 {%
```

```
2422 \let\theglentrycounter\@glxtr@thevalue
```

```
2423 \def\theHglentrycounter{\@glxtr@theHvalue}%
```

```
2424 }%
```

```
2425 \@gls@setsort{glslabel}%
```

Check if the textformat key has been used.

```
2426 \ifx\@glxtr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
2427 \glshasattribute{glslabel}{textformat}%
```

```
2428 {%
```

```
2429 \edef\@glxtr@attrval{\glsetattribute{glslabel}{textformat}}%
```

```
2430 \ifcsdef{\@glxtr@attrval}%
```

```
2431 {%
```

```
2432 \letcs{\@glxtr@textformat}{\@glxtr@attrval}%
```

```
2433 }%
```

```
2434 {%
```

```
2435 \GlossariesExtraWarning{Unknown control sequence name
```

```
2436 '\@glxtr@attrval' supplied in textformat attribute
```

```
2437 for entry 'glslabel'. Reverting to default \string\glstextformat}%
```

```

2438     \let\@glxstr@textformat\glstextformat
2439   }%
2440 }%
2441 {%
2442     \let\@glxstr@textformat\glstextformat
2443   }%
2444 \else
2445     \let\@glxstr@textformat\@glxstr@local@textformat
2446 \fi

Do write if it should occur before the link text:
2447 \ifglxstrinitwrglossbefore
2448     \@do@wrglossary{#2}%
2449 \fi

Do the link text:
2450 \ifKV@glslink@hyper
2451     \ifglxstr@hyperoutside
2452         \@glslink{\glolinkprefix\glslabel}{\@glxstr@textformat{#3}}%
2453     \else
2454         \@glxstr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2455     \fi
2456 \else
2457     \ifglxstr@hyperoutside
2458         \glndonohyperlink{\glolinkprefix\glslabel}{\@glxstr@textformat{#3}}%
2459     \else
2460         \@glxstr@textformat{\glndonohyperlink{\glolinkprefix\glslabel}{#3}}%
2461     \fi
2462 \fi

Do write if it should occur after the link text:
2463 \ifglxstrinitwrglossbefore
2464 \else
2465     \@do@wrglossary{#2}%
2466 \fi

Restore original value of \glolinkprefix:
2467 \let\glolinkprefix\@glxstr@org@glolinkprefix

As the original definition:
2468 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2469 }

2470 \define@key{glossadd}{thevalue}{\def\@glxstr@thevalue{#1}}
2471 \define@key{glossadd}{theHvalue}{\def\@glxstr@theHvalue{#1}}

lsaddpresetkeys
2472 \newcommand*{\gl saddpresetkeys}{}

saddpostsetkeys
2473 \newcommand*{\gl saddpostsetkeys}{}

```

`\glsadd` Redefine to include `\@glsxtr@record` and suppress in headings

```
2474 \renewrobustcmd*{\glsadd}[2][\]{%
2475   \glsxtrifinmark
2476   }{%
2477   {%
2478     \@gls@adjustmode
2479     \begingroup
2480     \glsxtr@record{#1}{#2}{glossadd}%
2481     \glsdoifexists{#2}%
2482     {%
2483       \let\@glsnumberformat\@glsxtr@defaultnumberformat
2484       \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
2485       \def\@glsxtr@thevalue{%
2486         \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Implement any default settings (before options are set)

```
2487   \glsaddpresetkeys
2488   \setkeys{glossadd}{#1}%
```

Implement any default settings (after options are set)

```
2489   \glsaddpostsetkeys
2490   \ifdefempty{\@glsxtr@thevalue}%
2491   {%
2492     \@gls@saveentrycounter
2493   }%
2494   {%
2495     \let\theglsentrycounter\@glsxtr@thevalue
2496     \def\theHglentrycounter{\@glsxtr@theHvalue}%
2497   }%
```

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

```
2498   \@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.

```
2499   \KV@glslink@noindexfalse
2500   \@do@wrglossary{#2}%
2501 }%
2502 \endgroup
2503 }%
2504 }
```

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

```
2505 \newrobustcmd*{\glsaddeach}[2][\]{%
2506   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2507 }
```

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

```
2508 \newcommand*{\@glsxtr@field@linkdefs}{%
```



```

2509 \let\glxtrifwasfirstuse\@secondoftwo
2510 \let\glrifplural\@secondoftwo
2511 \let\glscapscase\@firstofthree
2512 \let\glinsert\@empty
2513 }

```

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

assignfieldfont

```

2514 \newcommand*\glxtrassignfieldfont}[1]{%
2515   \ifglstryexists{#1}%
2516   {%
2517     \ifglshasshort{#1}%
2518     {%
2519       \glsetabbrvfmt{\glscategory{#1}}%
2520       \glrifregular{#1}%
2521       {\let\@gls@field@font\glxtrregularfont}%
2522       {\let\@gls@field@font\@firstofone}%
2523     }%
2524     {%
2525       \glrifnotregular{#1}%
2526       {\let\@gls@field@font\@firstofone}%
2527       {\let\@gls@field@font\glxtrregularfont}%
2528     }%
2529   }%
2530   {%
2531     \let\@gls@field@font\@gobble
2532   }%
2533 }

```

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

```

2534 \def\@glstext@#1#2[#3]{%
2535   \glxtrassignfieldfont{#2}%
2536   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2537 }

```

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

```

2538 \def\@GLStext@#1#2[#3]{%
2539   \glxtrassignfieldfont{#2}%
2540   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2541   {\@gls@field@font{\@GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2542 }

```

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

```

2543 \def\@Glstext@#1#2[#3]{%
2544   \glxtrassignfieldfont{#2}%
2545   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2546   {\@gls@field@font{\@GLSaccesstext{#2}#3}}%
2547 }

```

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`

```
2548 \newcommand*{\glstrchecknohyperfirst}[1]{%
2549   \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2550 }
```

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

```
2551 \def\@glsfirst@#1#2[#3]{%
2552   \glstrassignfieldfont{#2}%
   Ensure that \glsfirst honours the nohyperfirst attribute.
2553   \@gls@field@link
2554   [\let\glstrifwasfirstuse\@firstoftwo
2555     \glstrchecknohyperfirst{#2}%
2556   ]{#1}{#2}%
2557   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2558 }
```

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

```
2559 \def\@Glsfirst@#1#2[#3]{%
2560   \glstrassignfieldfont{#2}%
   Ensure that \@Glsfirst honours the nohyperfirst attribute.
2561   \@gls@field@link
2562   [\let\glstrifwasfirstuse\@firstoftwo
2563     \let\glscapscase\@secondofthree
2564     \glstrchecknohyperfirst{#2}%
2565   ]%
2566   {#1}{#2}{\@gls@field@font{\@Glsaccessfirst{#2}#3}}%
2567 }
```

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

```
2568 \def\@GLSfirst@#1#2[#3]{%
2569   \glstrassignfieldfont{#2}%
   Ensure that \@GLSfirst honours the nohyperfirst attribute.
2570   \@gls@field@link
2571   [\let\glstrifwasfirstuse\@firstoftwo
2572     \let\glscapscase\@thirdofthree
2573     \glstrchecknohyperfirst{#2}%
2574   ]%
2575   {#1}{#2}{\@gls@field@font{\@GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2576 }
```

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

```
2577 \def\@glsplural@#1#2[#3]{%
2578   \glstrassignfieldfont{#2}%
2579   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%

```

```

2580     {\@gls@field@font{\glsaccessplural{#2}#3}}%
2581 }

```

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

```

2582 \def\@Glsplural@#1#2[#3]{%
2583   \glstrassignfieldfont{#2}%
2584   \@gls@field@link
2585   [\let\glsifplural\@firstoftwo
2586    \let\glscapscase\@secondofthree
2587   ]%
2588   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2589 }

```

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

```

2590 \def\@GLSplural@#1#2[#3]{%
2591   \glstrassignfieldfont{#2}%
2592   \@gls@field@link
2593   [\let\glsifplural\@firstoftwo
2594    \let\glscapscase\@thirdofthree
2595   ]%
2596   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2597 }

```

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

```

2598 \def\@glsfirstplural@#1#2[#3]{%
2599   \glstrassignfieldfont{#2}%
2600   \glsfirstplural@honours the nohyperfirst attribute.
2601   \@gls@field@link
2602   [\let\glstrifwasfirstuse\@firstoftwo
2603    \let\glsifplural\@firstoftwo
2604    \glstrchecknohyperfirst{#2}%
2605   ]%
2606   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2607 }

```

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

```

2607 \def\@Glsfirstplural@#1#2[#3]{%
2608   \glstrassignfieldfont{#2}%
2609   \Glsfirstplural@honours the nohyperfirst attribute.
2610   \@gls@field@link
2611   [\let\glstrifwasfirstuse\@firstoftwo
2612    \let\glsifplural\@firstoftwo
2613    \let\glscapscase\@secondofthree
2614    \glstrchecknohyperfirst{#2}%
2615   ]%
2616   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2617 }

```

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

```
2617 \def\@GLSfirstplural@#1#2[#3]{%
2618   \glstrassignfieldfont{#2}%
   Ensure that \glsfirstplural honours the nohyperfirst attribute.
2619   \@gls@field@link
2620   [\let\glstrifwasfirstuse\@firstoftwo
2621    \let\glsifplural\@firstoftwo
2622    \let\glscapscase\@thirdofthree
2623    \glstrchecknohyperfirst{#2}%
2624   ]%
2625   {#1}{#2}%
2626   {\@gls@field@font{\@GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2627 }
```

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

```
2628 \def\@glsname@#1#2[#3]{%
2629   \glstrassignfieldfont{#2}%
2630   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessname{#2}#3}}%
2631 }
```

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

```
2632 \def\@Glsname@#1#2[#3]{%
2633   \glstrassignfieldfont{#2}%
2634   \@gls@field@link
2635   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2636   {\@gls@field@font{\@Glsaccessname{#2}#3}}%
2637 }
```

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

```
2638 \def\@GLSname@#1#2[#3]{%
2639   \glstrassignfieldfont{#2}%
2640   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2641   {#1}{#2}%
2642   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2643 }
```

\@glsdesc@

```
2644 \def\@glsdesc@#1#2[#3]{%
2645   \glstrassignfieldfont{#2}%
2646   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2647 }
```

\@Glsdesc@ First letter uppercase version.

```
2648 \def\@Glsdesc@#1#2[#3]{%
2649   \glstrassignfieldfont{#2}%
2650   \@gls@field@link
2651   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2652   {\@gls@field@font{\@Glsaccessdesc{#2}#3}}%
2653 }
```

```

\@GLSdesc@ All uppercase version.
2654 \def\@GLSdesc@#1#2[#3]{%
2655   \glstrassignfieldfont{#2}%
2656   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2657   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2658 }

@glsglscapscase@ No case-changing version.
2659 \def\@glsglscapscase@#1#2[#3]{%
2660   \glstrassignfieldfont{#2}%
2661   \@gls@field@link
2662   [\let\glscapscase\@secondoftwo
2663   \let\glsifplural\@firstoftwo
2664   ]{#1}{#2}{\@gls@field@font{\glsglscapscase{#2}#3}}%
2665 }

@Glsdescplural@ First letter uppercase version.
2666 \def\@Glsdescplural@#1#2[#3]{%
2667   \glstrassignfieldfont{#2}%
2668   \@gls@field@link
2669   [\let\glscapscase\@secondoftwo
2670   \let\glsifplural\@firstoftwo
2671   ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2672 }

@GLSdescplural@ All uppercase version.
2673 \def\@GLSdesc@#1#2[#3]{%
2674   \glstrassignfieldfont{#2}%
2675   \@gls@field@link
2676   [\let\glscapscase\@thirdoftwo
2677   \let\glsifplural\@firstoftwo
2678   ]%
2679   {#1}{#2}%
2680   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2681 }

\@glsymbol@
2682 \def\@glsymbol@#1#2[#3]{%
2683   \glstrassignfieldfont{#2}%
2684   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2685 }

\@Glsymbol@ First letter uppercase version.
2686 \def\@Glsymbol@#1#2[#3]{%
2687   \glstrassignfieldfont{#2}%
2688   \@gls@field@link
2689   [\let\glscapscase\@secondoftwo]%
2690   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2691 }

```

\@GLSsymbol@ All uppercase version.

```
2692 \def\@GLSsymbol@#1#2[#3]{%
2693   \glstrassignfieldfont{#2}%
2694   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2695   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2696 }
```

lssymbolplural@ No case-changing version.

```
2697 \def\@glssymbolplural@#1#2[#3]{%
2698   \glstrassignfieldfont{#2}%
2699   \@gls@field@link
2700   [\let\glscapscase\@secondoftwo
2701    \let\glsifplural\@firstoftwo
2702   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2703 }
```

lssymbolplural@ First letter uppercase version.

```
2704 \def\@Glssymbolplural@#1#2[#3]{%
2705   \glstrassignfieldfont{#2}%
2706   \@gls@field@link
2707   [\let\glscapscase\@secondoftwo
2708    \let\glsifplural\@firstoftwo
2709   ]{#1}{#2}{\@gls@field@font{\GLSaccesssymbolplural{#2}#3}}%
2710 }
```

LSsymbolplural@ All uppercase version.

```
2711 \def\@GLSsymbol@#1#2[#3]{%
2712   \glstrassignfieldfont{#2}%
2713   \@gls@field@link
2714   [\let\glscapscase\@thirdoftwo
2715    \let\glsifplural\@firstoftwo
2716   ]%
2717   {#1}{#2}%
2718   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2719 }
```

\@Glsuseri@ First letter uppercase version.

```
2720 \def\@Glsuseri@#1#2[#3]{%
2721   \glstrassignfieldfont{#2}%
2722   \@gls@field@link
2723   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2724   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2725 }
```

\@GLSuseri@ All uppercase version.

```
2726 \def\@GLSuseri@#1#2[#3]{%
2727   \glstrassignfieldfont{#2}%
2728   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
```

```

2729     {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
2730 }

```

\@Glsuserii@ First letter uppercase version.

```

2731 \def\@Glsuserii@#1#2[#3]{%
2732   \glstrassignfieldfont{#2}%
2733   \@gls@field@link
2734   [\let\glscapscase\@secondoftwo]%
2735   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}}%
2736 }

```

\@GLSuserii@ All uppercase version.

```

2737 \def\@GLSuserii@#1#2[#3]{%
2738   \glstrassignfieldfont{#2}%
2739   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2740   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2741 }

```

\@Glsuseriii@ First letter uppercase version.

```

2742 \def\@Glsuseriii@#1#2[#3]{%
2743   \glstrassignfieldfont{#2}%
2744   \@gls@field@link
2745   [\let\glscapscase\@secondoftwo]%
2746   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}}%
2747 }

```

\@GLSuseriii@ All uppercase version.

```

2748 \def\@GLSuseriii@#1#2[#3]{%
2749   \glstrassignfieldfont{#2}%
2750   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2751   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2752 }

```

\@Glsuseriv@ First letter uppercase version.

```

2753 \def\@Glsuseriv@#1#2[#3]{%
2754   \glstrassignfieldfont{#2}%
2755   \@gls@field@link
2756   [\let\glscapscase\@secondoftwo]%
2757   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}}%
2758 }

```

\@GLSuseriv@ All uppercase version.

```

2759 \def\@GLSuseriv@#1#2[#3]{%
2760   \glstrassignfieldfont{#2}%
2761   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2762   {#1}{#2}%
2763   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2764 }

```

\@Glsuserv@ First letter uppercase version.

```
2765 \def\@Glsuserv@#1#2[#3]{%
2766   \glxtrassignfieldfont{#2}%
2767   \@gls@field@link
2768   [\let\glscapscase\@secondoftwo]%
2769   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
2770 }
```

\@GLSuserv@ All uppercase version.

```
2771 \def\@GLSuserv@#1#2[#3]{%
2772   \glxtrassignfieldfont{#2}%
2773   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2774   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2775 }
```

\@Glsuservi@ First letter uppercase version.

```
2776 \def\@Glsuservi@#1#2[#3]{%
2777   \glxtrassignfieldfont{#2}%
2778   \@gls@field@link
2779   [\let\glscapscase\@secondoftwo]%
2780   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2781 }
```

\@GLSuservi@ All uppercase version.

```
2782 \def\@GLSuservi@#1#2[#3]{%
2783   \glxtrassignfieldfont{#2}%
2784   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2785   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2786 }
```

Commands like \acrshort already set \glsifplural, but they don't set \glxtrifwasfirstuse so they need adjusting. These commands shouldn't be used with \newabbreviation, but the redefinitions below allow for users reverting \newacronym back to its base definition.

base@acrcmd@warn Warn user that they need to use to new abbreviation commands.

```
2787 \newcommand*{\@@glxtr@base@acrcmd@warn}[2]{%
2788   \GlossariesExtraWarning{Base acronym command \string#1\space
2789     should not be used with new abbreviation definitions. Use
2790     \string#2\space instead}%
2791 }
```

glxtr@base@acrcmd Warn user that they need to use to new abbreviation commands.

```
2792 \let\@glxtr@base@acrcmd\@@glxtr@base@acrcmd@warn
```

\@acrshort No case change.

```
2793 \def\@acrshort#1#2[#3]{%
2794   \@glxtr@base@acrcmd\acrshort\glxtrshort
2795   \glsdofexists{#2}%
2796 }
```



```

2796 {%
2797   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2798   \let\glxtrifwasfirstuse\@secondoftwo
2799   \let\gl@ifplural\@secondoftwo
2800   \let\gl@scapscase\@firstofthree
2801   \let\gl@insert\@empty
2802   \def\glscustomtext{%
2803     \acronymfont{\gl@saccessshort{#2}}#3%
2804   }%
2805   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2806 }%
2807 \glspostlinkhook
2808 }

```

\@Acrshort First letter uppercase.

```

2809 \def\@Acrshort#1#2[#3]{%
2810   \@glxtr@base@acrcmd\Acrshort\Glsxtrshort
2811   \gl@ifexists{#2}%
2812   {%
2813     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2814     \let\glxtrifwasfirstuse\@secondoftwo
2815     \let\gl@ifplural\@secondoftwo
2816     \let\gl@scapscase\@secondofthree
2817     \let\gl@insert\@empty
2818     \def\glscustomtext{%
2819       \acronymfont{\gl@saccessshort{#2}}#3%
2820     }%
2821     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2822   }%
2823   \glspostlinkhook
2824 }

```

\@ACRshort All uppercase.

```

2825 \def\@ACRshort#1#2[#3]{%
2826   \@glxtr@base@acrcmd\ACRshort\GLSxtrshort
2827   \gl@ifexists{#2}%
2828   {%
2829     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2830     \let\glxtrifwasfirstuse\@secondoftwo
2831     \let\gl@ifplural\@secondoftwo
2832     \let\gl@scapscase\@thirdofthree
2833     \let\gl@insert\@empty
2834     \def\glscustomtext{%
2835       \mfirstucMakeUppercase{\acronymfont{\gl@saccessshort{#2}}#3}%
2836     }%
2837     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2838   }%
2839   \glspostlinkhook
2840 }

```

\@acrshortpl No case change.

```
2841 \def\@acrshortpl#1#2[#3]{%
2842   \@glsxtr@base@acrcmd\acrshortpl\glsxtrshortpl
2843   \glsdoifexists{#2}%
2844   {%
2845     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2846     \let\glsxtrifwasfirstuse\@secondoftwo
2847     \let\glsifplural\@firstoftwo
2848     \let\glsapscase\@firstofthree
2849     \let\glsinsert\@empty
2850     \def\glscustomtext{%
2851       \acronymfont{\glsaccessshortpl{#2}}#3%
2852     }%
2853     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2854   }%
2855   \glspostlinkhook
2856 }
```

\@Acrshortpl First letter uppercase.

```
2857 \def\@Acrshortpl#1#2[#3]{%
2858   \@glsxtr@base@acrcmd\Acrshortpl\Glsxtrshortpl
2859   \glsdoifexists{#2}%
2860   {%
2861     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2862     \let\glsxtrifwasfirstuse\@secondoftwo
2863     \let\glsifplural\@firstoftwo
2864     \let\glsapscase\@secondofthree
2865     \let\glsinsert\@empty
2866     \def\glscustomtext{%
2867       \acronymfont{\Glsaccessshortpl{#2}}#3%
2868     }%
2869     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2870   }%
2871   \glspostlinkhook
2872 }
```

\@ACRshortpl All uppercase.

```
2873 \def\@ACRshortpl#1#2[#3]{%
2874   \@glsxtr@base@acrcmd\ACRshortpl\GLSxtrshortpl
2875   \glsdoifexists{#2}%
2876   {%
2877     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2878     \let\glsxtrifwasfirstuse\@secondoftwo
2879     \let\glsifplural\@firstoftwo
2880     \let\glsapscase\@thirdofthree
2881     \let\glsinsert\@empty
2882     \def\glscustomtext{%
2883       \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
2884     }%
2885   }
```

```

2885 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2886 }%
2887 \glspostlinkhook
2888 }

```

\@acrlong No case change.

```

2889 \def\@acrlong#1#2[#3]{%
2890 \@glsxtr@base@acrcmd\acrlong\glsxtrlong
2891 \glsdoifexists{#2}%
2892 {%
2893 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2894 \let\glsxtrifwasfirstuse\@secondoftwo
2895 \let\glsifplural\@secondoftwo
2896 \let\glsapscase\@firstofthree
2897 \let\glsinsert\@empty
2898 \def\glscustomtext{%
2899 \acronymfont{\glsaccesslong{#2}}#3%
2900 }%
2901 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2902 }%
2903 \glspostlinkhook
2904 }

```

\@Acrlong First letter uppercase.

```

2905 \def\@Acrlong#1#2[#3]{%
2906 \@glsxtr@base@acrcmd\Acrlong\Glsxtrlong
2907 \glsdoifexists{#2}%
2908 {%
2909 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2910 \let\glsxtrifwasfirstuse\@secondoftwo
2911 \let\glsifplural\@secondoftwo
2912 \let\glsapscase\@secondofthree
2913 \let\glsinsert\@empty
2914 \def\glscustomtext{%
2915 \acronymfont{\Glsaccesslong{#2}}#3%
2916 }%
2917 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2918 }%
2919 \glspostlinkhook
2920 }

```

\@ACRlong All uppercase.

```

2921 \def\@ACRlong#1#2[#3]{%
2922 \@glsxtr@base@acrcmd\ACRlong\GLSxtrlong
2923 \glsdoifexists{#2}%
2924 {%
2925 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2926 \let\glsxtrifwasfirstuse\@secondoftwo
2927 \let\glsifplural\@secondoftwo

```

```

2928 \let\glscapscase\@thirdofthree
2929 \let\glsinsert\@empty
2930 \def\glscustomtext{%
2931     \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
2932 }%
2933 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2934 }%
2935 \glspostlinkhook
2936 }

```

\@acrlongpl No case change.

```

2937 \def\@acrlongpl#1#2[#3]{%
2938     \@glsxtr@base@acrcmd\acrlongpl\glsxtrlongpl
2939     \glsdoifexists{#2}%
2940     {%
2941         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2942         \let\glsxtrifwasfirstuse\@secondoftwo
2943         \let\glsifplural\@firstoftwo
2944         \let\glscapscase\@firstofthree
2945         \let\glsinsert\@empty
2946         \def\glscustomtext{%
2947             \acronymfont{\glsaccesslongpl{#2}}#3%
2948         }%
2949         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2950     }%
2951     \glspostlinkhook
2952 }

```

\@Acrlongpl First letter uppercase.

```

2953 \def\@Acrlongpl#1#2[#3]{%
2954     \@glsxtr@base@acrcmd\Acrlongpl\Glsxtrlongpl
2955     \glsdoifexists{#2}%
2956     {%
2957         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2958         \let\glsxtrifwasfirstuse\@secondoftwo
2959         \let\glsifplural\@firstoftwo
2960         \let\glscapscase\@secondofthree
2961         \let\glsinsert\@empty
2962         \def\glscustomtext{%
2963             \acronymfont{\Glsaccesslongpl{#2}}#3%
2964         }%
2965         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2966     }%
2967     \glspostlinkhook
2968 }

```

\@ACRlongpl All uppercase.

```

2969 \def\@ACRlongpl#1#2[#3]{%
2970     \@glsxtr@base@acrcmd\ACRlongpl\GLSxtrlongpl

```

```

2971 \glsdoifexists{#2}%
2972 {%
2973   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2974   \let\glsxtrifwasfirstuse\@secondoftwo
2975   \let\glsifplural\@firstoftwo
2976   \let\glsapscase\@thirdofthree
2977   \let\glsinsert\@empty
2978   \def\glscustomtext{%
2979     \mfirstucMakeUppercase{\acronymfont{\glsaccesslongpl{#2}}{#3}}%
2980   }%
2981   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2982 }%
2983 \glspostlinkhook
2984 }

```

The full formats use the internal long and short commands (such as `\@acrshort` and `\@acrlong`). Therefore they don't need adjustments, but they do need clearer warnings. This means three warnings per use (once for the full command and once each for the short and long commands), but at least this way the most important warning (replace `\acrfull` with `\glsxtrfull` etc) is present.

`\@acrfull`

```

2985 \def\@acrfull#1#2[#3]{%
2986   \@glsxtr@base@acrcmd\acrfull\glsxtrfull
2987   \acrfullfmt{#1}{#2}{#3}%
2988 }

```

`\@Acrfull`

```

2989 \def\@Acrfull#1#2[#3]{%
2990   \@glsxtr@base@acrcmd\Acrfull\Glsxtrfull
2991   \Acrfullfmt{#1}{#2}{#3}%
2992 }

```

`\@ACRfull`

```

2993 \def\@ACRfull#1#2[#3]{%
2994   \@glsxtr@base@acrcmd\ACRfull\GLSxtrfull
2995   \ACRfullfmt{#1}{#2}{#3}%
2996 }

```

`\@acrfullpl`

```

2997 \def\@acrfullpl#1#2[#3]{%
2998   \@glsxtr@base@acrcmd\acrfullpl\glsxtrfullpl
2999   \acrfullplfmt{#1}{#2}{#3}%
3000 }

```

`\@Acrfullpl`

```

3001 \def\@Acrfullpl#1#2[#3]{%
3002   \@glsxtr@base@acrcmd\Acrfullpl\Glsxtrfullpl

```

```

3003 \Acrfullplfmt{#1}{#2}{#3}%
3004 }

```

\@ACRfullpl

```

3005 \def\@ACRfullpl#1#2[#3]{%
3006 \@glstr@base@acrcmd\ACRfullpl\GLSxtrfullpl
3007 \ACRfullplfmt{#1}{#2}{#3}%
3008 }

```

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

\@gl saddkey

```

3009 \renewcommand*{\@gl saddkey}[7]{%
3010 \key@ifundefined{glossentry}{#1}%
3011 {%
3012 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
3013 \appto\@gl s@keymap{,{#1}{#1}}%
3014 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
3015 \appto\@newglossaryentryposthook{%
3016 \letcs{@glo@tmp}{@glo@#1}%
3017 \gl s@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
3018 }%
3019 \newcommand*{#3}[1]{\@gl s@entry@field{##1}{#1}}%
3020 \newcommand*{#4}[1]{\@gl s@entry@field{##1}{#1}}%

```

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

```

3021 \ifcsdef{@gl s@user@#1@}%
3022 {%
3023 \PackageError{glossaries}%
3024 {Can't define '\string#5' as helper command
3025 '\expandafter\string\csname @gl s@user@#1@endcsname' already
3026 exists}%
3027 }%
3028 }%
3029 {%
3030 \expandafter\newcommand\expandafter*\expandafter
3031 {\csname @gl s@user@#1@endcsname}[2] [] {%
3032 \new@ifnextchar[%
3033 {\csuse{@gl s@user@#1@}{##1}{##2}}%
3034 {\csuse{@gl s@user@#1@}{##1}{##2} [] }}%
3035 \csdef{@gl s@user@#1@}##1##2[##3]{%
3036 \@gl s@field@link{##1}{##2}{#3{##2}##3}%
3037 }%
3038 \newrobustcmd*{#5}{%
3039 \expandafter\@gl s@hyp@opt\csname @gl s@user@#1@endcsname}%
3040 }%

```

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

```

3041 \ifcsdef{@Gl s@user@#1@}%

```

```

3042   {%
3043     \PackageError{glossaries}%
3044     {Can't define '\string#6' as helper command
3045     '\expandafter\string\csname @Gls@user@#1@\endcsname' already
3046     exists}%
3047   }%
3048 }%
3049 {%
3050   \expandafter\newcommand\expandafter*\expandafter
3051   {\csname @Gls@user@#1@\endcsname}[2] [] {%
3052     \new@ifnextchar[%
3053       {\csuse{@Gls@user@#1@}{##1}{##2}}%
3054       {\csuse{@Gls@user@#1@}{##1}{##2} [] }}%
3055   \csdef{@Gls@user@#1@}##1##2[##3]{%
3056     \@gls@field@link[\let\glscapscase\@secondofthree]%
3057     {##1}{##2}{#4{##2}##3}%
3058   }%
3059   \newrobustcmd*{#6}{%
3060     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
3061   }%

```

Finally the all caps version (modified):

```

3062   \ifcsdef{@GLS@user@#1@}%
3063   {%
3064     \PackageError{glossaries}%
3065     {Can't define '\string#7' as helper command
3066     '\expandafter\string\csname @GLS@user@#1@\endcsname' already
3067     exists}%
3068   }%
3069 }%
3070 {%
3071   \expandafter\newcommand\expandafter*\expandafter
3072   {\csname @GLS@user@#1@\endcsname}[2] [] {%
3073     \new@ifnextchar[%
3074       {\csuse{@GLS@user@#1@}{##1}{##2}}%
3075       {\csuse{@GLS@user@#1@}{##1}{##2} [] }}%
3076   \csdef{@GLS@user@#1@}##1##2[##3]{%
3077     \@gls@field@link[\let\glscapscase\@thirdofthree]%
3078     {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
3079   }%
3080   \newrobustcmd*{#7}{%
3081     \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
3082   }%
3083 }%
3084 {%
3085   \PackageError{glossaries-extra}{Key '#1' already exists}{}%
3086 }%
3087 }

```

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

```
3088 \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.

```
3089 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
3090 \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). Note that if the entry is undefined (as with bib2gls on the first \TeX run), \ifglsused does neither true nor false parts. However, in that case, this macro won't be called anyway (since it's used in the argument of \glsdoifexistsordo).

```
3091 \ifglsused{\glslabel}%
3092 {\let\glsxtrifwasfirstuse\@secondoftwo}
3093 {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```
3094 \edef\glscategorylabel{\glscategory{\glslabel}}%
3095 \ifglsused{\glslabel}%
3096 {%
3097   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
3098   {\KV@glslink@hyperfalse}{}%
3099 }%
3100 {%
3101   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
3102   {\KV@glslink@hyperfalse}{}%
3103 }%
3104 \glslinkcheckfirsthyperhook
3105 }

```

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.

```
3106 \ifdef\do@glsglsdisablehyperinlist
3107 {%
3108   \let\@glsxtr@do@glsglsdisablehyperinlist\do@glsglsdisablehyperinlist
3109   \renewcommand*{\do@glsglsdisablehyperinlist}{%
3110     \@glsxtr@do@glsglsdisablehyperinlist
3111     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
3112   }
3113 }
3114 {}

```

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

```
3115 \define@boolkey{glslink}{noindex}[true]{}
3116 \KV@glslink@noindexfalse

```


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

`lt@glslink@opts`

```

3117 \ifdef \@gls@setdefault@glslink@opts
3118 {
3119   \renewcommand*{\@gls@setdefault@glslink@opts}{%
3120     \KV@glslink@noindexfalse
3121     \@glxtrsetaliasnoindex
3122   }
3123 }
3124 {

```

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

```

3125   \newcommand*{\@gls@setdefault@glslink@opts}{%
3126     \KV@glslink@noindexfalse
3127     \@glxtrsetaliasnoindex
3128   }
3129   \pretoto\do@glsglslink@hyperinlist{\@gls@setdefault@glslink@opts}
3130 }

```

`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.)

```

3131 \providecommand*\@glxtrsetaliasnoindex{%
3132   \KV@glslink@noindextrue
3133 }

```

`setaliasindex`

```

3134 \newcommand*\@glxtrsetaliasindex{%
3135   \glxtrifhasfield{alias}{\glslabel}%
3136   {%
3137     \let\glxtrindexaliased\@glxtrindexaliased
3138     \glxtrsetaliasnoindex
3139     \let\glxtrindexaliased\@no@glxtrindexaliased
3140   }%
3141   {%
3142 }

```

`xtrindexaliased`

```

3143 \newcommand*\@glxtrindexaliased{%
3144   \ifKV@glslink@noindex
3145   \else
3146     \begingroup
3147     \let\@glsnumberformat\@glxtr@defaultnumberformat
3148     \edef\@gls@counter{\csname glo@\glsglslink@label{\glslabel}@counter\endcsname}%
3149     \glxtr@saveentrycounter
3150     \@do@wrglossary{\glxtralias{\glslabel}}%
3151   \endgroup

```

```

3152 \fi
3153 }

```

extrindexaliased

```

3154 \newcommand{\@no@glxtrindexaliased}{%
3155   \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
3156   not permitted outside definition of \string\glxtrsetaliasnoindex}%
3157   {}%
3158 }

```

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

```

3159 \let\glxtrindexaliased\@no@glxtrindexaliased

```

DefaultGlsOpts Set the default options for \glslink etc.

```

3160 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
3161   \renewcommand*{\@glslsetdefault@glslink@opts}{%
3162     \setkeys{glslink}{#1}%
3163     \@glxtrsetaliasnoindex
3164   }%
3165 }

```

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

```

3166 \newcommand*{\glxtrifindexing}[2]{%
3167   \ifKV@glslink@noindex #2\else #1\fi
3168 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute. This needs to use \GlsXtrIfUnusedOrUndefined instead of \ifglused to allow it to work with bib2gls.

```

3169 \renewcommand*{\glswriteentry}[2]{%
3170   \glxtrifindexing
3171   {%
3172     \ifglindexonlyfirst
3173       \GlsXtrIfUnusedOrUndefined{#1}
3174       {#2}%
3175       {\glxtrdoautoindexname{#1}{dualindex}}}%
3176   \else
3177     \glslifattribute{#1}{indexonlyfirst}{true}%
3178     {%
3179       \GlsXtrIfUnusedOrUndefined{#1}%
3180       {#2}%
3181       {\glxtrdoautoindexname{#1}{dualindex}}}%
3182     }%
3183     {#2}%
3184   \fi
3185   }%
3186   {}%
3187 }

```

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

```
3188 \appto\@do@@wrglossary{\@glstr@do@@wrindex
3189   \glstrdowrglossaryhook{\@gls@label}}%
3190 }
```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```
3191 \appto\gls@noidxglossary{\@glstr@do@@wrindex
3192   \glstrdowrglossaryhook{\@gls@label}}%
3193 }
```

`xtr@do@@wrindex`

```
3194 \newcommand*{\@glstr@do@@wrindex}{%
3195   \glstrdoautoindexname{\@gls@label}{dualindex}}%
3196 }
```

`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.)

```
3197 \newcommand*{\glstrdowrglossaryhook}[1]{}
```

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

```
3198 \newcommand*{\@gls@alt@hyp@opt}[1]{%
3199   \let\glslinkvar\@firstofthree
3200   \let\@gls@hyp@opt@cs#1\relax
3201   \@ifstar{\s@gls@hyp@opt}%
3202   {\@ifnextchar+%
3203     {\@firstoftwo{\p@gls@hyp@opt}}%
3204     {%
3205       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
3206       {\@firstoftwo{\@alt@gls@hyp@opt}}%
3207       {#1}}%
3208   }%
3209 }%
3210 }
```

`alt@gls@hyp@opt` User version

```
3211 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
3212   \let\glslinkvar\@firstofthree
3213   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
3214 \newcommand*{\@gls@alt@hyp@opt@char}{}
```

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

```
3215 \newcommand*{\@gls@alt@hyp@opt@keys}{}
```

rSetAltModifier

```
3216 \newcommand*{\GlsXtrSetAltModifier}[2]{%
```

```
3217 \let\@gls@hyp@opt\@gls@alt@hyp@opt
```

Check that the supplied character isn't + or *

```
3218 \ifstrequal{#1}{+}%
```

```
3219 {\PackageError{glossaries-extra}%
```

```
3220 {Can't use '#1' as modifier (it's already in use)}}}%
```

```
3221 {%
```

```
3222 \ifstrequal{#1}{*}%
```

```
3223 {\PackageError{glossaries-extra}%
```

```
3224 {Can't use '#1' as modifier (it's already in use)}}}%
```

```
3225 }%
```

```
3226 }%
```

```
3227 \def\@gls@alt@hyp@opt@char{#1}%
```

```
3228 \def\@gls@alt@hyp@opt@keys{#2}%
```

```
3229 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
```

```
3230 {}%
```

```
3231 {%
```

Let **bib2gls** know the modifier.

```
3232 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@altmodifier}[1]{}}%
```

```
3233 \protected@write\@auxout{}{\string\@glsxtr@altmodifier{#1}}%
```

```
3234 }%
```

```
3235 }
```

org@dohyperlink

```
3236 \let\glsxtr@org@dohyperlink\glsdohyperlink
```

glsnavhyperlink Since `\glsnavhyperlink` uses `\glslink`, it's necessary to patch it uses `\glsdohyperlink` instead of `\glsxtrdohyperlink`. The simplest way to achieve this is to locally let `\glsxtrdohyperlink` to `\glsdohyperlink`.

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

```
3237 \ifdef\glsnavhyperlink
```

```
3238 {
```

```
3239 \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
```

```
3240 \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
3241 {%
```

```
3242 \let\glsxtrdohyperlink\glsxtr@org@dohyperlink
```

```
3243 \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

```
3244 }%
```

```
3245 }%
```

```
3246 }
```

```
3247 {}
```

The redefinition of `\glsdohyperlink` has been causing problems so introduce a new command instead.

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

```

3248 \newcommand*{\glsxtrdohyperlink}[2]{%
3249   \glsattribute{\glslabel}{targeturl}%
3250   {%
3251     \glsattribute{\glslabel}{targetname}%
3252     {%
3253       \glsattribute{\glslabel}{targetcategory}%
3254       {%
3255         \hyperref{\glsattribute{\glslabel}{targeturl}}%
3256           {\glsattribute{\glslabel}{targetcategory}}%
3257           {\glsattribute{\glslabel}{targetname}}%
3258           {\glsxtrprotectlinks#2}}%
3259       }%
3260     }%
3261     \hyperref{\glsattribute{\glslabel}{targeturl}}%
3262       {}%
3263       {\glsattribute{\glslabel}{targetname}}%
3264       {\glsxtrprotectlinks#2}}%
3265   }%
3266 }%
3267 {%
3268   \href{\glsattribute{\glslabel}{targeturl}}%
3269     {\glsxtrprotectlinks#2}}%
3270 }%
3271 }%
3272 {%

```

Check for alias.

```

3273   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
3274   \ifvoid\gloaliaslabel
3275   {%
3276     \glsxtrhyperlink{#1}{\glsxtrprotectlinks#2}}%
3277   }%
3278   {%

```

Redirect link to the alias target.

```

3279   \glsxtrhyperlink
3280     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
3281     {\glsxtrprotectlinks#2}}%
3282   }%

```

```

3283 }%
3284 }

```

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

```

3285 \ifdef\@glsshowtarget
3286 {
3287   \newcommand{\glsxtrhyperlink}[2]{%
3288     \@glsshowtarget{#1}%
3289     \hyperlink{#1}{#2}%
3290   }%
3291 }
3292 {
3293   \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
3294 }

```

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

```

3295 \renewrobustcmd*{\glslink}[2][\glstrytext{\@glo@label}]{%
3296   \glsdoifexists{#2}%
3297   {%
3298     \def\@glo@label{#2}%
3299     {\edef\glslabel{#2}%
3300      \@glslink{\glslinkprefix\glslabel}{#1}}%
3301   }%
3302 }

```

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

```

3303 \renewcommand{\glsdisablehyper}{%
3304   \KV@glslink@hyperfalse
3305   \def\@glslink{\glsdohyperlink}%
3306   \let\@glstarget\@secondoftwo
3307 }

```

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

```

3308 \renewcommand{\glsenablehyper}{%
3309   \KV@glslink@hypertrue
3310   \def\@glslink{\glsxtrdohyperlink}%
3311   \def\@glstarget{\glsdohypertarget}%
3312 }

```

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

```
3313 \def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}
```

\@glslink Reset \@glslink with patched versions:

```
3314 \ifcsundef{hyperlink}%
3315 {%
3316   \def\@glslink{\glsdonohyperlink}
3317 }%
3318 {%
3319   \def\@glslink{\glsxtrdohyperlink}
3320 }
```

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

```
3321 \newcommand*{\glsxtrprotectlinks}{%
3322   \KV@glslink@hyperfalse
3323   \KV@glslink@noindextrue
3324   \let\@gls@\@glsxtr@p@text@
3325   \let\@Gls@\@Glsxtr@p@text@
3326   \let\@GLS@\@GLSxtr@p@text@
3327   \let\@glspl@\@glsxtr@p@plural@
3328   \let\@Glspl@\@Glsxtr@p@plural@
3329   \let\@GLSpl@\@GLSxtr@p@plural@
3330   \let\@glsxtrshort@\@glsxtr@p@short@
3331   \let\@Glsxtrshort@\@Glsxtr@p@short@
3332   \let\@GLSxtrshort@\@GLSxtr@p@short@
3333   \let\@glsxtrlong@\@glsxtr@p@long@
3334   \let\@Glsxtrlong@\@Glsxtr@p@long@
3335   \let\@GLSxtrlong@\@GLSxtr@p@long@
3336   \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
3337   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
3338   \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
3339   \let\@glsxtrlongpl@\@glsxtr@p@longpl@
3340   \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
3341   \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
3342   \let\@acrshort@\@glsxtr@p@acrshort@
3343   \let\@Acrshort@\@Glsxtr@p@acrshort@
3344   \let\@ACRshort@\@GLSxtr@p@acrshort@
3345   \let\@acrshortpl@\@glsxtr@p@acrshortpl@
3346   \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
3347   \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
3348   \let\@acrlong@\@glsxtr@p@acrlong@
3349   \let\@Acrlong@\@Glsxtr@p@acrlong@
3350   \let\@ACRlong@\@GLSxtr@p@acrlong@
3351   \let\@acrlongpl@\@glsxtr@p@acrlongpl@
3352   \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
3353   \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
3354 }
```

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

```

@glxtr@p@text@
3355 \def\@glxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}}

@Glsxtr@p@text@
3356 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}}

@GLSxtr@p@text@
3357 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}}

lsxtr@p@plural@
3358 \def\@lsxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}}

lsxtr@p@plural@
3359 \def\@lsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}}

LSxtr@p@plural@
3360 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}}

glxtr@p@short@
3361 \def\@glxtr@p@short@#1#2[#3]{%
3362   {%
3363     \glsetabbrvfmt{\glscategory{#2}}%
3364     \glabbrvfont{\glentryshort{#2}}#3%
3365   }%
3366 }

Glsxtr@p@short@
3367 \def\@Glsxtr@p@short@#1#2[#3]{%
3368   {%
3369     \glsetabbrvfmt{\glscategory{#2}}%
3370     \glabbrvfont{\Glsentryshort{#2}}#3%
3371   }%
3372 }

GLSxtr@p@short@
3373 \def\@GLSxtr@p@short@#1#2[#3]{%
3374   {%
3375     \glsetabbrvfmt{\glscategory{#2}}%
3376     \mfirstucMakeUppercase{\glabbrvfont{\glentryshort{#2}}#3}%
3377   }%
3378 }

sxtr@p@shortpl@
3379 \def\@lsxtr@p@shortpl@#1#2[#3]{%
3380   {%
3381     \glsetabbrvfmt{\glscategory{#2}}%
3382     \glabbrvfont{\glentryshortpl{#2}}#3%
3383   }%
3384 }

```


sxtr@p@shortpl@

```
3385 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
3386   {%
3387     \glsetabbrvfmt{\glscategory{#2}}%
3388     \glsabbrvfont{\Glsentryshortpl{#2}}#3%
3389   }%
3390 }
```

Sxtr@p@shortpl@

```
3391 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
3392   {%
3393     \glsetabbrvfmt{\glscategory{#2}}%
3394     \mfirstucMakeUppercase{\glsabbrvfont{\Glsentryshortpl{#2}}#3}%
3395   }%
3396 }
```

@glxtr@p@long@

```
3397 \def\@glxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}}#3}}
```

@Glsxtr@p@long@

```
3398 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}}#3}}
```

@GLSxtr@p@long@

```
3399 \def\@GLSxtr@p@long@#1#2[#3]{%
3400   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
3401 \def\@lsxtr@p@longpl@#1#2[#3]{\{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
3402 \def\@GLSxtr@p@longpl@#1#2[#3]{\{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
3403 \def\@GLSxtr@p@longpl@#1#2[#3]{%
3404   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylongpl{#2}}#3}}}
```

xtr@p@acrshort@

```
3405 \def\@glxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
3406 \def\@Glsxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
3407 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
3408   {\mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#2}}#3}}}
```

r@p@acrshortpl@

```
3409 \def\@glxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\Glsentryshortpl{#2}}#3}}
```

```

r@p@acrshortpl@
3410 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
3411 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
3412   {\mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
3413 \def\@glxtr@p@acrlong@#1#2[#3]{\glentrylong{#2}#3}}

sxtr@p@acrlong@
3414 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
3415 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
3416   {\mfirstucMakeUppercase{\glentrylong{#2}#3}}}

tr@p@acrlongpl@
3417 \def\@glxtr@p@acrlongpl@#1#2[#3]{\glentrylongpl{#2}#3}}

tr@p@acrlongpl@
3418 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
3419 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
3420   {\mfirstucMakeUppercase{\glentrylongpl{#2}#3}}}

```

Commands to minimise conflict.

```

\@glxtrp@opt
3421 \newcommand*{\@glxtrp@opt}{hyper=false,noindex}

\glxtrsetpopts  Used in glossary to switch hyperlinks on for the \glxtrp type of commands.
3422 \newcommand*{\glxtrsetpopts}[1]{%
3423   \renewcommand*{\@glxtrp@opt}{#1}%
3424 }

lossxtrsetpopts  Used in glossary to switch hyperlinks on for the \glxtrp type of commands.
3425 \newcommand*{\glossxtrsetpopts}{%
3426   \glxtrsetpopts{noindex}%
3427 }

\@@glxtrp
3428 \newrobustcmd*{\@@glxtrp}[2]{%

```

Add scope.

```
3429 {%
3430   \let\glspostlinkhook\relax
3431   \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
3432 }%
3433 }
```

\@glsxtrp

```
3434 \newrobustcmd*{\@glsxtrp}[2]{%
3435   \ifcsdef{gls#1}%
3436   {%
3437     \@glsxtrp{gls#1}{#2}%
3438   }%
3439   {%
3440     \ifcsdef{glsxtr#1}%
3441     {%
3442       \@glsxtrp{glsxtr#1}{#2}%
3443     }%
3444     {%
3445       \PackageError{glossaries-extra}{‘#1’ not recognised by
3446         \string\glsxtrp}{}%
3447     }%
3448   }%
3449 }
```

\@Glsxtrp

```
3450 \newrobustcmd*{\@Glsxtrp}[2]{%
3451   \ifcsdef{Gls#1}%
3452   {%
3453     \@glsxtrp{Gls#1}{#2}%
3454   }%
3455   {%
3456     \ifcsdef{Glsxtr#1}%
3457     {%
3458       \@glsxtrp{Glsxtr#1}{#2}%
3459     }%
3460     {%
3461       \PackageError{glossaries-extra}{‘#1’ not recognised by
3462         \string\Glsxtrp}{}%
3463     }%
3464   }%
3465 }
```

\@GLSxtrp

```
3466 \newrobustcmd*{\@GLSxtrp}[2]{%
3467   \ifcsdef{GLS#1}%
3468   {%
3469     \@glsxtrp{GLS#1}{#2}%
3470   }%
```

```

3471 {%
3472   \ifcsdef{GLSxtr#1}%
3473   {%
3474     \@@glsxtrp{GLSxtr#1}{#2}%
3475   }%
3476   {%
3477     \PackageError{glossaries-extra}{‘#1’ not recognised by
3478       \string\GLSxtrp}{}%
3479   }%
3480 }%
3481 }

```

\glsxtr@entry@p

```

3482 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3483   \glsifattribute{#1}{headuc}{true}%
3484   {%
3485     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3486   }%
3487   {%
3488     \@gls@entry@field{#1}{#2}%
3489   }%
3490 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3491 \ifdef\texorpdfstring
3492 {
3493   \newcommand{\glsxtrp}[2]{%
3494     \protect\NoCaseChange
3495     {%
3496       \protect\texorpdfstring
3497       {%
3498         \protect\glsxtrifinmark
3499         {%
3500           \ifcsdef{glsxtrhead#1}%
3501           {%
3502             {\protect\csuse{glsxtrhead#1}{#2}}%
3503           }%
3504           {%
3505             \glsxtr@headentry@p{#2}{#1}%
3506           }%
3507         }%
3508         {%
3509           \@glsxtrp{#1}{#2}%
3510         }%
3511       }%
3512     }%
3513     \protect\@gls@entry@field{#2}{#1}%
3514   }%
3515 }%

```

```

3516 }
3517 }
3518 {
3519   \newcommand{\glxtrp}[2]{%
3520     \protect\NoCaseChange
3521     {%
3522       \protect\glxtrifinmark
3523       {%
3524         \ifcsdef{glxtrhead#1}%
3525         {%
3526           {\protect\csuse{glxtrhead#1}}%
3527         }%
3528         {%
3529           \glxtr@headentry@p{#2}{#1}%
3530         }%
3531       }%
3532     }%
3533     \@glxtrp{#1}{#2}%
3534   }%
3535 }%
3536 }
3537 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3538 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

3539 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

3540 \ifdef\textorpdfstring
3541 {
3542   \newcommand{\Glsxtrp}[2]{%
3543     \protect\NoCaseChange
3544     {%
3545       \protect\textorpdfstring
3546       {%
3547         \protect\glxtrifinmark
3548         {%
3549           \ifcsdef{Glsxtrhead#1}%
3550           {%
3551             {\protect\csuse{Glsxtrhead#1}{#2}}%
3552           }%
3553           {%
3554             \protect\@Gls@entry@field{#2}{#1}%
3555           }%

```

```

3556     }%
3557     {%
3558         \@Glsxtrp{#1}{#2}%
3559     }%
3560 }%
3561 {%
3562     \protect\@gls@entry@field{#2}{#1}%
3563 }%
3564 }%
3565 }
3566 }
3567 {
3568     \newcommand{\Glsxtrp}[2]{%
3569         \protect\NoCaseChange
3570         {%
3571             \protect\glsxtrifinmark
3572             {%
3573                 \ifcsdef{Glsxtrhead#1}%
3574                 {%
3575                     {\protect\csuse{Glsxtrhead#1}}%
3576                 }%
3577                 {%
3578                     \protect\@Gls@entry@field{#2}{#1}%
3579                 }%
3580             }%
3581             {%
3582                 \@Glsxtrp{#1}{#2}%
3583             }%
3584         }%
3585     }
3586 }

```

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

```

3587 \ifdef\texorpdfstring
3588 {
3589     \newcommand{\Glsxtrp}[2]{%
3590         \protect\NoCaseChange
3591         {%
3592             \protect\texorpdfstring
3593             {%
3594                 \protect\glsxtrifinmark
3595                 {%
3596                     \ifcsdef{Glsxtr#1}%
3597                     {%
3598                         {\protect\Glsxtrshort[noindex,hyper=false]{#1}[]}%
3599                     }%
3600                     {%
3601                         \protect\mfirstucMakeUppercase
3602                     }%

```

```

3603         \protect\@gls@entry@field{#2}{#1}%
3604     }%
3605 }%
3606 }%
3607 {%
3608     \@GLSxtrp{#1}{#2}%
3609 }%
3610 }%
3611 {%
3612     \protect\@gls@entry@field{#2}{#1}%
3613 }%
3614 }%
3615 }
3616 }
3617 {
3618     \newcommand{\GLSxtrp}[2]{%
3619         \protect\NoCaseChange
3620         {%
3621             \protect\glsxtrifinmark
3622             {%
3623                 \ifcsdef{GLSxtr#1}%
3624                 {%
3625                     {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3626                 }%
3627                 {%
3628                     \protect\mfirstucMakeUppercase
3629                     {%
3630                         \protect\@gls@entry@field{#2}{#1}%
3631                     }%
3632                 }%
3633             }%
3634             {%
3635                 \@GLSxtrp{#1}{#2}%
3636             }%
3637         }%
3638     }
3639 }

```

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.

```
3640 \newcommand*{\@glsxtr@unset}[1]{%
3641   \@@glsunset{#1}%
3642   \glsxtrpostunset{#1}%
3643 }
```

`\@glsunset` Global unset.

```
3644 \let\@glsunset\@glsxtr@unset
```

`\glsxtrpostunset`

```
3645 \newcommand*{\glsxtrpostunset}[1]{}
```

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

`\tUnsetBuffering`

```
3646 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3647   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3648 }
```

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

```
3649 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3650   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3651   \def\@glsxtr@unset@buffer{}%
3652   \let\@glsunset\@glsxtrbuffer@unset
3653 }
```

`\tUnsetBuffering` Starred version checks for duplicates.

```
3654 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3655   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3656   \def\@glsxtr@unset@buffer{}%
3657   \let\@glsunset\@glsxtrbuffer@nodup@unset
3658 }
```

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

```
3659 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3660   \listxadd\@glsxtr@unset@buffer{#1}%
3661 }
```

`\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.)


```

3662 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3663   \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}}%
3664   {\listxadd\@glsxtr@unset@buffer{#1}}}%
3665 }

```

pUnsetBuffering

```

3666 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3667   \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3668 }

```

pUnsetBuffering Unstarred form (global unset).

```

3669 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3670   \let\@glsunset\@glsxtr@unset
3671   \forlistloop\@glsunset\@glsxtr@unset@buffer
3672   \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3673 }

```

pUnsetBuffering Starred form (local unset).

```

3674 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3675   \forlistloop\@glslocalunset\@glsxtr@unset@buffer
3676   \let\@glsunset\@glsxtr@unset
3677 }

```

dUnsetBuffering Discards pending buffer and restores \glsunset.

```

3678 \newcommand*{\GlsXtrDiscardUnsetBuffering}{%
3679   \let\@glsunset\@glsxtr@unset
3680   \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3681 }

```

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

```

3682 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3683   \forlistloop#1\@glsxtr@unset@buffer
3684 }

```

\@glslocalunset Local unset.

```

3685 \renewcommand*{\@glslocalunset}[1]{%
3686   \@glslocalunset{#1}%
3687   \glsxtrpostlocalunset{#1}%
3688 }%

```

rpostlocalunset

```

3689 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

\@glsreset Global reset.

```

3690 \renewcommand*{\@glsreset}[1]{%
3691   \@glsreset{#1}%
3692   \glsxtrpostreset{#1}%
3693 }%

```

glsxtrpostreset

```
3694 \newcommand*{\glsxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
3695 \renewcommand*{\@glslocalreset}[1]{%
3696   \@glslocalreset{#1}%
3697   \glsxtrpostlocalreset{#1}%
3698 }%
```

rpostlocalreset

```
3699 \newcommand*{\glsxtrpostlocalreset}[1]{}
```

slocalreseteach Locally reset a list of entries.

```
3700 \newcommand*{\glslocalreseteach}[1]{%
3701   \gls@ifnotmeasuring
3702   {%
3703     \@for\@gls@thislabel:=#1\do{%
3704       \glsdoifexists{\@gls@thislabel}%
3705       {%
3706         \@glslocalreset{\@gls@thislabel}%
3707       }%
3708     }%
3709   }%
3710 }
```

slocalunseteach Locally unset a list of entries.

```
3711 \newcommand*{\glslocalunseteach}[1]{%
3712   \gls@ifnotmeasuring
3713   {%
3714     \@for\@gls@thislabel:=#1\do{%
3715       \glsdoifexists{\@gls@thislabel}%
3716       {%
3717         \@glslocalunset{\@gls@thislabel}%
3718       }%
3719     }%
3720   }%
3721 }
```

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

```
3722 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

 Enable entry counting:

```
3723   \glsenableentrycount
```

 Redefine \gls etc:

```
3724   \renewcommand*{\gls}{\cgl}%
3725   \renewcommand*{\Gls}{\cGls}%
3726   \renewcommand*{\glspl}{\cglspl}%
```

```

3727 \renewcommand*{\Glspl}{\cGlspl}%
3728 \renewcommand*{\GLS}{\cGLS}%
3729 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3730 \@glstr@setentrycountunsetattr{#1}{#2}%

```

In case this command is used again:

```

3731 \let\GlsXtrEnableEntryCounting\@glstr@setentrycountunsetattr
3732 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3733 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3734 can't be used with \string\GlsXtrEnableEntryCounting}%
3735 {Use one or other but not both commands}}%
3736 }

```

ycountunsetattr

```

3737 \newcommand*{\@glstr@setentrycountunsetattr}[2]{%
3738 \@for\@glstr@cat:=#1\do
3739 {%
3740 \ifdefempty{\@glstr@cat}{}%
3741 {%
3742 \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
3743 }%
3744 }%
3745 }

```

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

nableentrycount

```

3746 \renewcommand*{\glsenableentrycount}{%

```

Enable new fields:

```

3747 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%

```

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

```

3748 \renewcommand*{\gls@defdocnewglossaryentry}{%
3749 \renewcommand*\newglossaryentry[2]{%
3750 \PackageError{glossaries}{\string\newglossaryentry\space
3751 may only be used in the preamble when entry counting has
3752 been activated}{If you use \string\glsenableentrycount\space
3753 you must place all entry definitions in the preamble not in
3754 the document environment}%
3755 }%
3756 }%

```

New commands to access new fields:

```

3757 \newcommand*{\glsentrycurrcount}[1]{%
3758 \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3759 {0}{\@gls@entry@field{##1}{currcount}}}%
3760 }%
3761 \newcommand*{\glsentryprevcount}[1]{%

```

```

3762 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3763 {0}{\@gls@entry@field{##1}{prevcount}}}%
3764 }%

```

Adjust post unset and reset:

```

3765 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
3766 \renewcommand*{\glsxtrpostunset}[1]{%
3767   \@glsxtr@entrycount@org@unset{##1}%
3768   \@gls@increment@currcount{##1}%
3769 }%
3770 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
3771 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3772   \@glsxtr@entrycount@org@localunset{##1}%
3773   \@gls@local@increment@currcount{##1}%
3774 }%
3775 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
3776 \renewcommand*{\glsxtrpostreset}[1]{%
3777   \@glsxtr@entrycount@org@reset{##1}%
3778   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3779 }%
3780 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
3781 \renewcommand*{\glsxtrpostlocalreset}[1]{%
3782   \@glsxtr@entrycount@org@localreset{##1}%
3783   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3784 }%

```

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

```

3785 \let\@cgls@\@cgls@
3786 \let\@cglspl@\@cglspl@

3787 \let\@cGls@\@cGls@
3788 \let\@cGlspl@\@cGlspl@
3789 \let\@cGLS@\@cGLS@
3790 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

3791 \AtEndDocument{\@gls@write@entrycounts}%
3792 \renewcommand*{\@gls@entry@count}[2]{%
3793   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
3794 }%
3795 \let\glsenableentrycount\relax
3796 \renewcommand*{\glsenableentryunitcount}{%
3797   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3798     can't be used with \string\glsenableentrycount}%
3799   {Use one or other but not both commands}%
3800 }%
3801 }

```

`\@gls@write@entrycounts` Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```

3802 \renewcommand*{\@gls@write@entrycounts}{%
3803   \immediate\write\@auxout
3804     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3805   \count@=0\relax
3806   \forallglsentries{\@glsentry}{%
3807     \glshasattribute{\@glsentry}{entrycount}%
3808     {%
3809       \ifglsused{\@glsentry}%
3810       {%
3811         \immediate\write\@auxout
3812           {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3813       }%
3814     }%
3815     \advance\count@ by \@ne
3816   }%
3817 }%
3818 }%
3819 \ifnum\count@=0
3820   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3821     \MessageBreak with \string\glsenableentrycount\space but the
3822     \MessageBreak attribute 'entrycount' hasn't
3823     \MessageBreak been assigned to any of the defined
3824     \MessageBreak entries}%
3825 \fi
3826 }

```

rifcounttrigger

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

```

3827 \newcommand*{\glxtrifcounttrigger}[3]{%
3828   \glshasattribute{#1}{entrycount}%
3829   {%
3830     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
3831     #3%
3832   \else
3833     #2%
3834   \fi
3835 }%
3836 {#3}%
3837 }

```

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

`\@@cgl@`

```

3838 \def\@@cgl@#1#2[#3]{%
3839   \glxtrifcounttrigger{#2}%

```

```

3840 {%
3841   \cglformat{#2}{#3}%
3842   \glset{#2}%
3843 }%
3844 {%
3845   \@gls@{#1}{#2}[#3]%
3846 }%
3847 }%

```

\@@cglsp1@

```

3848 \def\@@cglsp1@#1#2[#3]{%
3849   \glxtrifcounttrigger{#2}%
3850   {%
3851     \cglsp1format{#2}{#3}%
3852     \glset{#2}%
3853   }%
3854   {%
3855     \@glspl@{#1}{#2}[#3]%
3856   }%
3857 }%

```

\@@cGls@

```

3858 \def\@@cGls@#1#2[#3]{%
3859   \glxtrifcounttrigger{#2}%
3860   {%
3861     \cGlsformat{#2}{#3}%
3862     \glset{#2}%
3863   }%
3864   {%
3865     \@Gls@{#1}{#2}[#3]%
3866   }%
3867 }%

```

\@@cGlspl@

```

3868 \def\@@cGlspl@#1#2[#3]{%
3869   \glxtrifcounttrigger{#2}%
3870   {%
3871     \cGlsplformat{#2}{#3}%
3872     \glset{#2}%
3873   }%
3874   {%
3875     \@Glspl@{#1}{#2}[#3]%
3876   }%
3877 }%

```

\@@cGLS@

```

3878 \def\@@cGLS@#1#2[#3]{%
3879   \glxtrifcounttrigger{#2}%
3880   {%

```

```

3881 \cGLSformat{#2}{#3}%
3882 \glsunset{#2}%
3883 }%
3884 {%
3885 \cGLS@{#1}{#2}[#3]%
3886 }%
3887 }%

```

\cGLSp1@

```

3888 \def\cGLSp1@#1#2[#3]{%
3889 \glstrifcounttrigger{#2}%
3890 {%
3891 \cGLSp1format{#2}{#3}%
3892 \glsunset{#2}%
3893 }%
3894 {%
3895 \cGLSp1@{#1}{#2}[#3]%
3896 }%
3897 }%

```

Remove default warnings from \cgl's etc so that it can be used interchangeable with \gls etc.

\cgl's@

```

3898 \def\cgl's@#1#2[#3]{\cgl's@{#1}{#2}[#3]}

```

\cGls@

```

3899 \def\cGls@#1#2[#3]{\cGls@{#1}{#2}[#3]}

```

\cgl'sp1@

```

3900 \def\cgl'sp1@#1#2[#3]{\cgl'sp1@{#1}{#2}[#3]}

```

\cGl'sp1@

```

3901 \def\cGl'sp1@#1#2[#3]{\cGl'sp1@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

\cGLS

```

3902 \newrobustcmd*{\cGLS}{\cgl's@hyp@opt\cGLS}

```

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

```

3903 \newcommand*{\cGLS}[2][{}]{%
3904 \new@ifnextchar[{\cGLS@{#1}{#2}}{\cGLS@{#1}{#2}[{}]}%
3905 }

```

\cGLS@

```

3906 \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.

```
3907 \newcommand*{\cGLSformat}[2]{%
3908   \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
3909 }
```

`\cGLSpl`

```
3910 \newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

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

```
3911 \newcommand*{\@cGLSpl}[2][ ]{%
3912   \new@ifnextchar[{\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[ ]}%
3913 }
```

`\@cGLSpl@`

```
3914 \def\@cGLSpl@#1#2[#3]{\@cGLSpl@{#1}{#2}[#3]}
```

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

```
3915 \newcommand*{\cGLSplformat}[2]{%
3916   \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3917 }
```

Modify the trigger formats to check for the regular attribute.

`\cglformat`

```
3918 \renewcommand*{\cglformat}[2]{%
3919   \glsifregular{#1}
3920   {\glsentryfirst{#1}}%
3921   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3922 }
```

`\cGlsformat`

```
3923 \renewcommand*{\cGlsformat}[2]{%
3924   \glsifregular{#1}
3925   {\Glsentryfirst{#1}}%
3926   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3927 }
```

`\cglsplformat`

```
3928 \renewcommand*{\cglsplformat}[2]{%
3929   \glsifregular{#1}
3930   {\glsentryfirstplural{#1}}%
3931   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3932 }
```


\cGlsplformat

```
3933 \renewcommand*{\cGlsplformat}[2]{%
3934   \glsifregular{#1}
3935   {\Glsentryfirstplural{#1}}%
3936   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2}%
3937 }
```

New code similar to above for unit counting.

defunitcounters

```
3938 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3939   \edef\@glo@countunit{\csuse{\glstr@categoryattr\@glo@category @unitcount}}%
3940   \ifdefined\@glo@countunit
3941     {}%
3942     {%
3943       \@glstr@ifunitcounter{\@glo@countunit}%
3944       {}%
3945       {\expandafter\@glstr@addunitcounter\expandafter{\@glo@countunit}}%
3946     }%
3947 }
```

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

```
3948 \newcommand*{\@glstr@unitcountlist}{}%
```

@addunitcounter

```
3949 \newcommand*{\@glstr@addunitcounter}[1]{%
3950   \listadd{\@glstr@unitcountlist}{#1}%
3951   \ifcsundef{glstr@theunit@#1}
3952     {%
3953       \ifcsdef{theH#1}%
3954       {\csdef{glstr@theunit@#1}{\csuse{theH#1}}}%
3955       {\csdef{glstr@theunit@#1}{\csuse{the#1}}}%
3956     }%
3957   {}%
3958 }
```

r@ifunitcounter

```
3959 \newcommand*{\@glstr@ifunitcounter}[3]{%
3960   \xifinlist{#1}{\@glstr@unitcountlist}{#2}{#3}%
3961 }
```

urrentunitcount

```
3962 \newcommand*{\@glstr@currentunitcount}[1]{%
3963   glo@\glstetoklabel{#1}@currunit@\glstetattribute{#1}{unitcount}.%
3964   \csuse{glstr@theunit@\glstetattribute{#1}{unitcount}}%
3965 }
```

previousunitcount

```
3966 \newcommand*\@glsxtr@previousunitcount[1]{%
3967   glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
3968   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3969 }
```

t@currunitcount

```
3970 \newcommand*\@gls@increment@currunitcount[1]{%
3971   \gls@hasattribute{#1}{unitcount}%
3972   {%
3973     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3974     \ifcsundef{\@glsxtr@csname}%
3975     {%
3976       \csgdef{\@glsxtr@csname}{1}%
3977       \listcsxadd
3978         {glo@\glsdetoklabel{#1}@unitlist}%
3979         {\glsgetattribute{#1}{unitcount}.%
3980         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3981       }%
3982     }%
3983     {%
3984       \csxdef{\@glsxtr@csname}%
3985       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3986     }%
3987   }%
3988   {}%
3989 }
```

t@currunitcount

```
3990 \newcommand*\@gls@local@increment@currunitcount[1]{%
3991   \gls@hasattribute{#1}{unitcount}%
3992   {%
3993     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3994     \ifcsundef{\@glsxtr@csname}%
3995     {%
3996       \csdef{\@glsxtr@csname}{1}%
3997       \listcseadd
3998         {glo@\glsdetoklabel{#1}@unitlist}%
3999         {\glsgetattribute{#1}{unitcount}.%
4000         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
4001       }%
4002     }%
4003     {%
4004       \csedef{\@glsxtr@csname}%
4005       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
4006     }%
4007   }%
4008   {}%
4009 }
```

r@currunitcount

```
4010 \newcommand*{\@glxstr@currunitcount}[2]{%
4011   \ifcsundef
4012   {glo@\glstetoklabel{#1}@currunit@#2}%
4013   {0}%
4014   {\csuse{glo@\glstetoklabel{#1}@currunit@#2}}%
4015 }%
```

r@prevunitcount

```
4016 \newcommand*{\@glxstr@prevunitcount}[2]{%
4017   \ifcsundef
4018   {glo@\glstetoklabel{#1}@prevunit@#2}%
4019   {0}%
4020   {\csuse{glo@\glstetoklabel{#1}@prevunit@#2}}%
4021 }%
```

entryunitcount

```
4022 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
4023   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
4024   \renewcommand*{\glstetoknewglossaryentry}{%
4025     \renewcommand*\newglossaryentry[2]{%
4026       \PackageError{glossaries}{\string\newglossaryentry\space
4027         may only be used in the preamble when entry counting has
4028         been activated}{If you use \string\glsenableentryunitcount\space
4029         you must place all entry definitions in the preamble not in
4030         the document environment}%
4031     }%
4032   }%
```

New commands to access new fields:

```
4033 \newcommand*{\glstentrycurrcount}[1]{%
4034   \@glxstr@currunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
4035   \csuse{glxstr@theunit@\glstgetattribute{##1}{unitcount}}}%
4036 }%
4037 \newcommand*{\glstentryprevcount}[1]{%
4038   \@glxstr@prevunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
4039   \csuse{glxstr@theunit@\glstgetattribute{##1}{unitcount}}}%
4040 }%
```

Access total count:

```
4041 \newcommand*{\glstentryprevtotalcount}[1]{%
4042   \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
4043   {0}%
4044   {%
4045     \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
4046   }%
4047 }%
```

Access max value:

```

4048 \newcommand*{\glstentryprevmaxcount}[1]{%
4049 \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
4050 {0}%
4051 {%
4052 \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
4053 }%
4054 }%

```

Adjust post unset and reset:

```

4055 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
4056 \renewcommand*{\glstxtrpostunset}[1]{%
4057 \@glstxtr@entryunitcount@org@unset{##1}%
4058 \@glst@increment@currunitcount{##1}%
4059 }%
4060 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
4061 \renewcommand*{\glstxtrpostlocalunset}[1]{%
4062 \@glstxtr@entryunitcount@org@localunset{##1}%
4063 \@glst@local@increment@currunitcount{##1}%
4064 }%
4065 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
4066 \renewcommand*{\glstxtrpostreset}[1]{%
4067 \glshasattribute{##1}{unitcount}%
4068 {%
4069 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
4070 \ifcsundef{\@glstxtr@csname}%
4071 {}%
4072 {\csgdef{\@glstxtr@csname}{0}}%
4073 }%
4074 {}%
4075 }%
4076 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
4077 \renewcommand*{\glstxtrpostlocalreset}[1]{%
4078 \@glstxtr@entryunitcount@org@localreset{##1}%
4079 \glshasattribute{##1}{unitcount}%
4080 {%
4081 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
4082 \ifcsundef{\@glstxtr@csname}%
4083 {}%
4084 {\csdef{\@glstxtr@csname}{0}}%
4085 }%
4086 {}%
4087 }%

```

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

```

4088 \let\@cglst@\@cglst@
4089 \let\@cglstpl@\@cglstpl@
4090 \let\@cGls@\@cGls@

```

```

4091 \let\@cGlspl@\@cGlspl@
4092 \let\@cGLS@\@cGLS@
4093 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

4094 \AtEndDocument{\@gls@write@entryunitcounts}%
4095 \renewcommand*{\@gls@entry@unitcount}[3]{%
4096   \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
4097   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
4098   {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
4099   {%
4100     \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
4101       \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
4102     }%
4103     \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
4104     {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
4105     {%
4106       \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
4107       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
4108     }%
4109   }%
4110 }%
4111 \let\glsenableentryunitcount\relax
4112 \renewcommand*{\glsenableentrycount}{%
4113   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
4114     can't be used with \string\glsenableentryunitcount}%
4115   {Use one or other but not both commands}%
4116 }%
4117 }
4118 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

4119 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

4120 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
4121   \immediate\write\@auxout
4122   {\string\@gls@entry@unitcount
4123     {\@glsentry}%
4124     {\@glsxtr@currunitcount{\@glsentry}{#1}%
4125     }%
4126     {#1}}%
4127 }

```

entryunitcounts

```

4128 \newcommand*{\@gls@write@entryunitcounts}{%
4129   \immediate\write\@auxout
4130   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
4131   \count@=0\relax

```

```

4132 \foralllglsentries{\@glsentry}{%
4133   \glshasattribute{\@glsentry}{unitcount}%
4134   {%
4135     \ifglused{\@glsentry}%
4136     {%
4137       \forlistcsloop
4138         {\@gls@write@entryunitcounts@do}%
4139         {glo@\glsdetoklabel{\@glsentry}@unitlist}%
4140     }%
4141   }%
4142   \advance\count@ by \@ne
4143 }%
4144 {}%
4145 }%
4146 \ifnum\count@=0
4147   \GlossariesExtraWarningNoLine{Entry counting has been enabled
4148     \MessageBreak with \string\glsenableentryunitcount\space but the
4149     \MessageBreak attribute ‘unitcount’ hasn’t
4150     \MessageBreak been assigned to any of the defined
4151     \MessageBreak entries}%
4152 \fi
4153 }

```

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.

```

4154 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

4155   \glsenableentryunitcount

```

Redefine \gls etc:

```

4156   \renewcommand*{\gls}{\cgl}%
4157   \renewcommand*{\Gls}{\cGls}%
4158   \renewcommand*{\glspl}{\cglspl}%
4159   \renewcommand*{\Glspl}{\cGlspl}%
4160   \renewcommand*{\GLS}{\cGLS}%
4161   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

4162   \@glxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

4163   \let\GlsXtrEnableEntryUnitCounting\@glxtr@setentryunitcountunsetattr
4164   \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
4165     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
4166       can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
4167     {Use one or other but not both commands}}%
4168 }

```

tcountunsetattr

```

4169 \newcommand*{\@glstr@setentryunitcountunsetattr}[3]{%
4170   \@for\@glstr@cat:=#1\do
4171   {%
4172     \ifdefempty{\@glstr@cat}{}%
4173     {%
4174       \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
4175       \glsssetcategoryattribute{\@glstr@cat}{unitcount}{#3}%
4176     }%
4177   }%
4178 }

```

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.

`\SetGenericNewAcronym`

```

4179 \renewcommand*{\SetGenericNewAcronym}{%
    Make sure \RestoreAcronyms has been used.
4180   \ifdefequal\@addtoacronymlists\@glstr@org@addtoacronymlists
4181   {}%
4182   {%
4183     \GlossariesWarning{\string\SetGenericNewAcronym\space used
4184       without restoring base acronym functions with
4185       \string\RestoreAcronyms}%
4186   }%
4187   \let\@Gls@entryname\@Gls@acrenryname

    Redefine \newacronym:
4188   \renewcommand{\newacronym}[4][ ]{%
4189     \ifdefempty{\@glsacronymlists}%
4190     {%
4191       \def\@glo@type{\acronymtype}%
4192       \setkeys{glossentry}{##1}%
4193       \DeclareAcronymList{\@glo@type}%
4194     }%
4195     {}%
4196     \glskeylisttok{##1}%
4197     \glslabeltok{##2}%
4198     \glsshorttok{##3}%
4199     \glslongtok{##4}%
4200     \newacronymhook
4201     \protected@edef\@do@newglossaryentry{%
4202       \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

4203     {%
4204         type=\acronymtype,%
4205         name={\expandonce{\acronymentry{##2}}},%
4206         sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
4207         text={\the\glsshorttok},%
4208         short={\the\glsshorttok},%
4209         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4210         long={\the\glslongtok},%
4211         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4212         category=acronym,
4213         \GenericAcronymFields,%
4214         \the\glskeylisttok
4215     }%
4216 }%
4217 \do@newglossaryentry
4218 }%
4219 \renewcommand*{\acrfullfmt}[3]{%
4220     \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
4221 \renewcommand*{\Acrfullfmt}[3]{%
4222     \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
4223 \renewcommand*{\ACRfullfmt}[3]{%
4224     \glslink[##1]{##2}{%
4225         \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
4226 \renewcommand*{\acrfullplfmt}[3]{%
4227     \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
4228 \renewcommand*{\Acrfullplfmt}[3]{%
4229     \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
4230 \renewcommand*{\ACRfullplfmt}[3]{%
4231     \glslink[##1]{##2}{%
4232         \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
4233 \renewcommand*{\glstryfull}[1]{\genacrfullformat{##1}{}}}%
4234 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
4235 \renewcommand*{\glstryfullpl}[1]{\genplacrfullformat{##1}{}}}%
4236 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
4237 }

```

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:

```

4238 \let\@glxtr@org@setacronymstyle\setacronymstyle
4239 \let\@glxtr@org@newacronymstyle\newacronymstyle

```

Save the list of acronyms in case they are required.

tr@acronymlists

```

4240 \let\@glxtr@acronymlists\@glsacronymlists

```

dtoacronymlists

```

4241 \let\@glxtr@org@addtoacronymlists\@addtoacronymlists

```


setacronymlists

```
4242 \let\@glxtr@org@setacronymlists\SetAcronymLists
```

Need to provide a replacement for `\forallacronyms` since `\@glsacronymlists` isn't available.

lsxtr@abbrlists

```
4243 \newcommand{\@glxtr@abbrlists}{}%
```

breviationlists

```
4244 \newcommand*{\forallabbreviationlists}[2]{%
4245   \@for#1:=\@glxtr@abbrlists\do{\ifdefempty{#1}{-}{#2}}%
4246 }
```

bbreviationlist

```
4247 \newcommand*{\@glxtr@addabbreviationlist}[1]{%
4248   \edef\@glo@type{#1}%
4249   \ifdefempty\@glxtr@abbrlists
4250   {\let\@glxtr@abbrlists\@glo@type}%
4251   {%
4252     \ifdefequal\@glxtr@abbrlists\@glo@type
4253     {}%
4254     {%
4255       \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@abbrlists}{}%
4256       {\eappto\@glxtr@abbrlists{,\@glo@type}}%
4257     }%
4258   }%
4259 }
```

\forallacronyms Modify to add warning.

```
4260 \renewcommand*{\forallacronyms}[2]{%
4261   \@glxtr@base@acrcmd\forallacronyms\forallabbreviationlists
4262   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
4263 }
```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbreviationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```
4264 \newcommand*{\MakeAcronymsAbbreviations}{%
```

Undo acronym display style:

```
4265   \@for\@gls@type:=\@glsacronymlists\do{%
4266     \csgdef{gls@\@gls@type @entryfmt}{\glsentryfmt}%
4267   }%
```

Save and clear acronym list.

```
4268   \let\@glxtr@acronymlists\@glsacronymlists
4269   \let\@glsacronymlists\@empty
4270   \let\@addtoacronymlists\@gobble
4271   \let\SetAcronymLists\@gobble
```

Warn if \acrshort etc are used.

```
4272 \let\@glxtr@base@acrcmd\@glxtr@base@acrcmd@warn
```

Redefine \newacronym to use same interface as \newabbreviation.

```
4273 \renewcommand*\newacronym}[4][]{%
4274   \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4275 }%
4276 \renewcommand*\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4277 \renewcommand*\acronymfont}[1]{\glsabbrvfont{##1}}%
4278 \renewcommand*\setacronymstyle}[1]{%
4279   \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
4280     unavailable.
4281     Use \string\setabbreviationstyle[acronym]\space instead.
4282     The original acronym interface can be restored with
4283     \string\RestoreAcronyms}{}%
4284 }%
4285 \renewcommand*\newacronymstyle}[1]{%
4286   \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4287     available unless you restore the original acronym interface with
4288     \string\RestoreAcronyms}%
4289   \@glxtr@org@newacronymstyle{##1}%
4290 }%
4291 }
```

Switch acronyms to abbreviations:

```
4292 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```
4293 \newcommand*\RestoreAcronyms{%
```

Restore acronym list.

```
4294 \let\@glxacronymlists\@glxtr@acronymlists
4295 \let\@addtoacronymlists\@glxtr@org@addtoacronymlists
4296 \let\SetAcronymLists\@glxtr@org@setacronymlists
```

Suppress warnings if \acrshort etc are used.

```
4297 \let\@glxtr@base@acrcmd\@gobbletwo
```

Restore acronym display style:

```
4298 \@for\@glstype:=\@glxacronymlists\do{%
4299   \SetDefaultAcronymDisplayStyle{\@glstype}%
4300 }%
```

Switch to the generic acronym mechanism.

```
4301 \SetGenericNewAcronym
4302 \renewcommand*\firstacronymfont}[1]{\acronymfont{##1}}%
4303 \renewcommand*\acronymfont}[1]{##1}%
4304 \let\setacronymstyle\@glxtr@org@setacronymstyle
4305 \let\newacronymstyle\@glxtr@org@newacronymstyle
```

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

```

4306 \renewcommand*\@gls@link@checkfirsthyper{%
4307   \ifglused{\glslabel}%
4308   {\let\glstrifwasfirstuse\@secondoftwo}
4309   {\let\glstrifwasfirstuse\@firstoftwo}%
4310   \@glstr@org@checkfirsthyper
4311 }
4312 \glssetcategoryattribute{acronym}{regular}{false}%
4313 \setacronymstyle{long-short}%
4314 }

```

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

```

4315 \renewcommand*\glsacspace[1]{%
4316   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4317   \ifdim\dimen@<\glsacspacemax~\else\space\fi
4318 }

```

`\glsacspacemax` Value used in the above.

```

4319 \newcommand*\glsacspacemax{3em}

```

1.3.7 Indexing and Displaying Glossaries

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

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

`r@reg@glosslist`

```

4320 \newcommand*\@glstr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

4321 \let\@glstr@org@makeglossaries\makeglossaries

```

`omakeglossaries` glossaries v4.45 introduced `\@domakeglossaries` to provide a way of disabling `\makeglossaries`.

If it hasn't been defined, define here to do its argument:

```

4322 \providecommand{\@domakeglossaries}[1]{#1}

```

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`

```

4323 \renewcommand*\makeglossaries[1][]{%
4324   \@domakeglossaries
4325   {%
4326     \@glstr@if@record@only

```

```

4327 {%
4328   \PackageError{glossaries-extra}{\string\makeglossaries\space
4329     not permitted\MessageBreak with record=\@glxtr@record@setting\space
4330     package option}%
4331   {You may only use \string\makeglossaries\space with
4332     record=off or record=alsoindex options}%
4333 }%
4334 {%
4335   \ifblank{#1}%
4336   {\@glxtr@org@makeglossaries}%
4337   {%
4338     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
4339     \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4340       not permitted\MessageBreak with record=alsoindex package option}%
4341     {You may only use the hybrid \string\makeglossaries[...]\space with
4342       record=off option}%
4343   \else
4344     \@gls@@automake@immediate was introduced to glossaries v4.42 so it may not be defined.
4345     \ifdef\@gls@@automake@immediate{\@gls@@automake@immediate}{}%
4346     \edef\@glxtr@reg@glosslist{#1}%
4347     \ifundef{\glswrite}{\newwrite\glswrite}{}%
4348     \protected@write\@auxout{}{\string\providecommand
4349       \string\@glsorder[1]{} }%
4350     \protected@write\@auxout{}{\string\providecommand
4351       \string\@istfilename[1]{} }%
4352     \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4353     \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
4354     \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}%
4355     \write\@auxout{\string\providecommand\string\@gls@reference[3]{} }%
4356
4357     Iterate through each supplied glossary type and activate it.
4358     \for\@glo@type:=#1\do{%
4359       \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4360     }%
4361
4362     New glossaries must be created before \makeglossaries:
4363     \renewcommand*\newglossary[4][]{%
4364       \PackageError{glossaries}{New glossaries
4365         must be created before \string\makeglossaries}{You need
4366         to move \string\makeglossaries\space after all your
4367         \string\newglossary\space commands}}%
4368
4369     Any subsequence instances of this command should have no effect.
4370     \let\@makeglossary\@gobble
4371
4372     Version 1.42 removed letting \makeglossary to \relax (no kernel redefs may be in effect).
4373     \renewcommand\makeglossaries[1][]{}%
4374
4375     Disable all commands that have no effect after \makeglossaries
4376     \@disable@onlypremakeg

```

Allow see key:

```
4366 \let\gls@checkseeallowed\relax
```

Adjust \do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```
4367 \renewcommand*{\do@seeglossary}[2]{%
4368 \glsdoifexists{##1}%
4369 {%
4370 \edef\@gls@label{\glsdetoklabel{##1}}%
4371 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4372 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4373 {\@glsxtr@org@doseeglossary{##1}{##2}}%
4374 {%
4375 \@glsxtrwrglossmark
4376 \protected@write\@auxout{%
4377 \string\@gls@reference
4378 {\@gls@type}{\@gls@label}{\string\glsseeformat##2}}%
4379 }%
4380 }%
4381 }%
4382 }%
```

Adjust @@do@@wrglossary

```
4383 \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
4384 \def\@do@@wrglossary{%
4385 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4386 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4387 {\@glsxtr@@do@@wrglossary}%
4388 {\gls@noidxglossary}%
4389 }%
```

Suppress warning about no \makeglossaries

```
4390 \let\warn@nomakeglossaries\relax
4391 \def\warn@noprintglossary{%
4392 \GlossariesWarningNoLine{No \string\printglossary\space
4393 or \string\printglossaries\space
4394 found.^^J(Remove \string\makeglossaries\space if you don't want
4395 any glossaries.)^^JThis document will not have a glossary}%
4396 }%
```

Only warn for glossaries not listed.

```
4397 \renewcommand{\@gls@noref@warn}[1]{%
4398 \edef\@gls@type{##1}%
4399 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4400 {%
4401 \GlossariesExtraWarning{Can't use
4402 \string\printnoidxglossary[type={\@gls@type}]
4403 when '\@gls@type' is listed in the optional argument of
4404 \string\makeglossaries}%
4405 }%
4406 }%
```

```

4407         \GlossariesWarning{Empty glossary for
4408         \string\printnoidxglossary[type={##1}].
4409         Rerun may be required (or you may have forgotten to use
4410         commands like \string\gls)}%
4411     }%
4412 }%

```

Adjust display number list to check for type:

```

4413     \renewcommand*{\glsdisplaynumberlist}[1]{%
4414     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4415     {\@glsxtr@idx@displaynumberlist{##1}}%
4416     {\@glsxtr@noidx@displaynumberlist{##1}}%
4417 }%

```

Adjust entry list:

```

4418     \renewcommand*{\glsentrynumberlist}[1]{%
4419     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4420     {\@glsxtr@idx@entrynumberlist{##1}}%
4421     {\@glsxtr@noidx@entrynumberlist{##1}}%
4422 }%

```

Adjust number list loop

```

4423     \renewcommand*{\glsnumberlistloop}[2]{%
4424     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4425     {%
4426         \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4427         not available for glossary ‘##1’}{}%
4428     }%
4429     {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
4430 }%

```

Only sanitize sort for normal indexing glossaries.

```

4431     \renewcommand*{\glsprestandardsort}[3]{%
4432     \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4433     {%
4434         \glsdosanitizesort
4435     }%
4436     {%
4437         \ifglssanitizesort
4438         \@gls@noidx@sanitizesort
4439         \else
4440         \@gls@noidx@nosanitizesort
4441         \fi
4442     }%
4443 }%

```

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

```

4444     \renewcommand*{\new@glossaryentry}[2]{%
4445     \PackageError{glossaries-extra}{Glossary entries must be defined
4446     in the preamble\MessageBreak when you use the optional argument

```

```

4447         of \string\makeglossaries}{Either move your definitions to the
4448         preamble or don't use the optional argument of
4449         \string\makeglossaries}%
4450     }%

```

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).

```

4451     \let\glo@assign@sortkey\glsxtr@mixed@assign@sortkey
4452     \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

4453         \expandafter\glsxtr@gettype\expandafter,\glsxtr@printglossopts,%
4454         type=\glsdefaulttype,\@end\glsxtr@gettype
4455         \def\glo@sorttype{\glo@default@sorttype}%
4456     }%

```

Check automake setting:

```

4457     \ifglsautomake
4458     \renewcommand*{\@gls@doautomake}{%
4459         \for\gls@type:=\glsxtr@reg@glosslist\do{%
4460             \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
4461         }%
4462     }%
4463 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

4464     \ifdef\glo@check@sortallowed{\glo@check@sortallowed\makeglossaries}{}%
4465 \fi
4466 }%
4467 }%
4468 }%
4469 }

```

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

Earlier versions of glossaries-extra simply saved the original version of \@printglossary with \let \@glsxtr@orgprintglossary. This was later changed to actually defining \@glsxtr@orgprintglossary to something similar with some alterations to allow for ignored glossaries, which don't have an associated title and to by-pass the existence check with \ifglossaryexists which doesn't recognise ignored glossaries. (bib2gls writes \provideignoredglossary to the glstex file for some settings, so the glossary might not been defined on the first \LaTeX run and it needs to be allowed with \printunsrtglossary on subsequent runs.)

Unfortunately, removing the existence check will cause an error if \printglossary is used with an ignored glossary.

As from glossaries v4.46, some new commands have been included to allow the existence check to be varied depending on whether or not ignored glossaries should be allowed, so check for them:

oss@checkexists

```

4470 \ifdef\@printgloss@checkexists
4471 {\newcommand{\glxtr@printgloss@checkexists}{\@printgloss@checkexists}}
4472 {\newcommand{\glxtr@printgloss@checkexists}[2]{#2}}

```

rgprintglossary (This command is also used for on-the-fly setting.)

```

4473 \newcommand{\@glxtr@orgprintglossary}[2]{%
4474   \def\@glo@type{\gldefaulttype}%

```

Add check here.

```

4475   \def\glossarytitle{%
4476     \ifcsdef{\@glo@type\@glo@type @title}%
4477     {\csuse{\@glo@type\@glo@type @title}}%
4478     {\glossaryname}}%
4479   \def\glossarytoctitle{\glossarytitle}%
4480   \let\org@glossarytitle\glossarytitle
4481   \def\@glossarystyle{%
4482     \ifx\@glossary@default@style\relax
4483       \GlossariesWarning{No default glossary style provided \MessageBreak
4484         for the glossary '\@glo@type'. \MessageBreak
4485         Using deprecated fallback. \MessageBreak
4486         To fix this set the style with \MessageBreak
4487         \string\setglossarystyle\space or use the \MessageBreak
4488         style key=value option}%
4489     \fi
4490   }%
4491   \def\gls@dotoc@title{\glsettoctitle{\@glo@type}}%
4492   \let\@org@glossaryentrynumbers\glossaryentrynumbers
4493   \bgroup
4494     \@printgloss@setsort
4495     \setkeys{printgloss}{#1}%
4496     \ifx\glossarytitle\org@glossarytitle
4497     \else
4498       \cslet{\@glo@type\@glo@type @title}{\glossarytitle}%
4499     \fi
4500     \let\currentglossary\@glo@type
4501     \let\org@glossaryentrynumbers\glossaryentrynumbers
4502     \let\glsnonextpages\@glsnonextpages
4503     \let\glsnextpages\@glsnextpages

4504     \glxtractivatenopost
4505     \gls@dotoc@title
4506     \@glossarystyle
4507     \let\gls@org@glossaryentryfield\glossentry
4508     \let\gls@org@glossarysubentryfield\subglossentry
4509     \renewcommand{\glossentry}[1]{%
4510       \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4511       \gls@org@glossaryentryfield{##1}%
4512     }%
4513     \renewcommand{\subglossentry}[2]{%
4514       \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%

```



```

4515     \gls@org@glossarysubentryfield{##1}{##2}%
4516   }%
4517   \@gls@preglossaryhook
4518   \glsxtr@printgloss@checkexists{\@glo@type}{#2}%
4519 \egroup
4520 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4521 \global\let\warn@noprintglossary\relax
4522 }

```

ractivatenopost Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```

4523 \newcommand*{\glsxtractivatenopost}{%
4524   \let\nopostdesc\@nopostdesc
4525   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4526 }

```

lsxtrnopostpunc

```

4527 \newrobustcmd*{\glsxtrnopostpunc}{%

```

lsxtr@nopostpunc Provide a command that works like `\nopostdesc` but only switches of the punctuation without suppressing the post-description hook.

```

4528 \newcommand{\@glsxtr@nopostpunc}{%
4529   \let\@glsxtr@org@postdescription\glspostdescription
4530   \ifglsnopostdot
4531     \renewcommand{\glspostdescription}{%
4532       \glsnopostdottrue
4533       \let\glspostdescription\@glsxtr@org@postdescription
4534       \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
4535       \glsxtrpostdescription
4536       \@glsxtr@nopostpunc@postdesc}%
4537   \else
4538     \renewcommand{\glspostdescription}{%
4539       \let\glspostdescription\@glsxtr@org@postdescription
4540       \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
4541       \glsxtrpostdescription
4542       \@glsxtr@nopostpunc@postdesc}%
4543   \fi
4544   \glsnopostdotfalse
4545 }

```

stpunc@postdesc

```

4546 \newcommand*{\@glsxtr@nopostpunc@postdesc}{%

```

estore@postpunc

```

4547 \newcommand*{\@glsxtr@restore@postpunc}{%
4548   \def\@glsxtr@nopostpunc@postdesc{%
4549     \@glsxtr@org@postdescription
4550     \let\@glsxtr@nopostpunc@postdesc\@empty
4551     \let\glsxtrrestorepostpunc\@empty

```

```

4552 }%
4553 }

```

`\restorepostpunc` Does nothing outside of glossary.

```

4554 \newcommand*{\glxtrrestorepostpunc}{}

```

`\@printglossary` Redefine.

```

4555 \renewcommand{\@printglossary}[2]{%
4556   \def\@glxtr@printglossopts{#1}%
4557   \@glxtr@orgprintglossary{#1}{#2}%
4558 }

```

Add a key that switches off the entry targets:

```

4559 \define@choicekey{printgloss}{target}%
4560 [\@glxtr@printglossval\@glxtr@printglossnr]%
4561 {true,false}[true]%
4562 {%
4563   \ifcase\@glxtr@printglossnr
4564     \def\@glstarget{\glsdohypertarget}%
4565   \else
4566     \let\@glstarget\@secondoftwo
4567   \fi
4568 }

```

`\hypernameprefix`

```

4569 \newcommand{\@glxtrhypernameprefix}{}

```

New to v1.20:

```

4570 \define@key{printgloss}{targetnameprefix}{%
4571   \renewcommand{\@glxtrhypernameprefix}{#1}%
4572 }

4573 \define@key{printgloss}{prefix}{%
4574   \renewcommand{\glolinkprefix}{#1}%
4575 }

4576 \define@key{printgloss}{label}{%
4577   \glxtrsetglossarylabel{#1}%
4578 }

```

`\setglossarylabel` Set the label for subsequent glossaries. If the label is fixed (that is, doesn't change with each glossary) this will need to be scoped or changed again to prevent duplicate labels.

```

4579 \newcommand{\glxtrsetglossarylabel}[1]{%
4580   \renewcommand*{\@@glossaryseclabel}{%
4581     \protected@edef\@currentlabelname{\glossarytoctitle}%
4582     \label{#1}%
4583   }%
4584 }

```

xtr@leveloffset

```
4585 \newcount\@glxtr@leveloffset
```

New to vl.44:

```
4586 \define@key{printgloss}{leveloffset}{%  
4587   \@glxtr@assign@leveloffset#1\relax  
4588 }
```

ign@leveloffset

```
4589 \newcommand*{\@glxtr@assign@leveloffset}{%  
4590   \@ifnextchar+{\p@glxtr@assign@leveloffset}{\np@glxtr@assign@leveloffset}%  
4591 }
```

ign@leveloffset Discard initial + character.

```
4592 \newcommand*{\p@glxtr@assign@leveloffset}[1]{%  
4593   \@ifnextchar+{\pp@glxtr@assign@leveloffset}{\np@glxtr@assign@leveloffset}%  
4594 }
```

ign@leveloffset

```
4595 \def\np@glxtr@assign@leveloffset#1\relax{\@glxtr@leveloffset=#1\relax}
```

ign@leveloffset

```
4596 \def\pp@glxtr@assign@leveloffset#1\relax{\advance\@glxtr@leveloffset by #1\relax}
```

```
4597 \define@boolkey{printgloss}[glxtr@printgloss@]{groups}[true]{}  
4598 \glxtr@printgloss@groupstrue
```

lsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```
4599 \let\@glxtr@org@glldohypertarget\glldohypertarget  
4600 \renewcommand{\glldohypertarget}[2]{%  
4601   \@glxtr@org@glldohypertarget{\@glxtrhypernameprefix#1}{#2}%  
4602 }
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
4603 \ifx\@glstarget\@glxtr@org@glldohypertarget  
4604   \def\@glstarget{\glldohypertarget}%  
4605 \fi
```

@makeglossaries For the benefit of makeglossaries

```
4606 \newcommand*{\glxtr@makeglossaries}[1]{}%
```

@glxtr@gettype Get just the type.

```
4607 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%  
4608   \def\@glo@type{#2}%  
4609 }
```

@assign@sortkey Assign the sort key.

```
4610 \newcommand\@glstr@mixed@assign@sortkey[1]{%
4611   \edef\@glo@type{\@glo@type}%
4612   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glstr@reg@glosslist}%
4613   {%
4614     \@glo@no@assign@sortkey{#1}%
4615   }%
4616   {%
4617     \@glo@assign@sortkey{#1}%
4618   }%
4619 }%
```

Display number list for the regular version:

splaynumberlist

```
4620 \let\@glstr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4621 \newcommand*{\@glstr@noidx@displaynumberlist}[1]{%
4622   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4623   \ifdef\@gls@loclist
4624   {%
4625     \def\@gls@noidx@loclist@sep{%
4626       \def\@gls@noidx@loclist@sep{%
4627         \def\@gls@noidx@loclist@sep{%
4628           \glsnumlistsep
4629         }%
4630         \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4631       }%
4632     }%
4633     \def\@gls@noidx@loclist@finalsep{}%
4634     \def\@gls@noidx@loclist@prev{}%
4635     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4636     \@gls@noidx@loclist@finalsep
4637     \@gls@noidx@loclist@prev
4638   }%
4639   {%
4640     \glstrundeftag
4641     \glsdoifexists{#1}%
4642     {%
4643       \GlossariesWarning{Missing location list for ‘#1’. Either
4644         a rerun is required or you haven’t referenced the entry.}%
4645     }%
4646   }%
4647 }%
4648
```

And for the number list loop:

@numberlistloop

```
4649 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
4650   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4651   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4652   \let\@gls@org@glsseeformat\glsseeformat
4653   \let\glsnoidxdisplayloc#2\relax
4654   \let\glsseeformat#3\relax
4655   \ifdef\@gls@loclist
4656   {%
4657     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4658   }%
4659   {%

4660     \glsxtrundeftag
4661     \glsdoifexists{#1}%
4662     {%
4663       \GlossariesWarning{Missing location list for ‘##1’. Either
4664         a rerun is required or you haven’t referenced the entry.}%
4665     }%
4666   }%
4667   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4668   \let\glsseeformat\@gls@org@glsseeformat
4669 }%
```

Same for entry number list.

entrynumberlist

```
4670 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4671   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4672   \ifdef\@gls@loclist
4673   {%
4674     \glsnoidxloclist{\@gls@loclist}%
4675   }%
4676   {%

4677     \glsxtrundeftag
4678     \glsdoifexists{#1}%
4679     {%
4680       \GlossariesWarning{Missing location list for ‘#1’. Either
4681         a rerun is required or you haven’t referenced the entry.}%
4682     }%
4683   }%
4684 }%
```

entrynumberlist

```
4685 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
4686 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4687   \protected@edef\@glsxtr@titlelabel{#1}%
```

```

4688 \ifdefvoid\@glsxtr@titlelabel
4689 {}%
4690 {%
4691   \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4692 }%
4693 \ifdefvoid{\@glsxtr@titlelabel}%
4694 {%
4695   \DTLifint{#1}%
4696   {%
4697     \ifnum#1<256\relax
4698       \edef#2{\char#1\relax}%
4699     \else
4700       \edef#2{#1}%
4701     \fi
4702   }%
4703   {%
4704     \ifcsundef{#1groupname}%
4705       {\def#2{#1}}%
4706       {\letcs#2{#1groupname}}%
4707   }%
4708 }%
4709 {%
4710   \let#2\@glsxtr@titlelabel
4711 }%
4712 }

```

gls@getgrouptitle Save original definition of `\@gls@getgrouptitle`

```
4713 \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.

```

4714 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4715   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4716   \@onelevel@sanitize\@glsxtr@titlelabel
4717   \ifcsdef{\@glsxtr@titlelabel}
4718     {\letcs{#2}{\@glsxtr@titlelabel}}%
4719     {\glsxtr@org@getgrouptitle{#1}{#2}}%
4720 }
4721 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

4722 \newcommand{\glsxtrsetgrouptitle}[2]{%
4723   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4724   \@onelevel@sanitize\@glsxtr@titlelabel
4725   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4726 }

```

alsetgrouptitle As above put only locally defines the title.

```
4727 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
```

```

4728 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
4729 \@onelevel@sanitize\@glstr@titlelabel
4730 \protected@csedef{\@glstr@titlelabel}{#2}%
4731 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4732 \renewcommand*{\glsnavigation}{%
4733   \def\@gls@between{}%
4734   \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
4735   {%
4736     \def\@gls@list{}%
4737   }%
4738   {%
4739     \expandafter\let\expandafter\@gls@list
4740       \csname \@gls@hypergroup@list@\@glo@type\endcsname
4741   }%
4742   \@for\@gls@tmp:=\@gls@list\do{%
4743     \@gls@between
4744     \glstrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4745     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4746     \let\@gls@between\glshypernavsep
4747   }%
4748 }

```

`@noidx@glossary`

```

4749 \renewcommand*{\@print@noidx@glossary}{%
4750   \ifcsdef{\@glsref@\@glo@type}%
4751   {%
4752     \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4753     {%
4754       \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4755     }%
4756     {%
4757       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4758     }%
4759     \glossarysection[\glossarytoctitle]{\glossarytitle}%
4760     \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4761   \def\@gls@currentlettergroup{}%
4762   \begin{theglossary}%
4763     \glossaryheader
4764     \glsresetentrylist
4765     \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
4766     \end{theglossary}%
4767     \glossarypostamble
4768   }%
4769   {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4770 \glxtrifemptyglossary{\@glo@type}%
4771 {}%
4772 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4773 \@gls@noref@warn{\@glo@type}%
4774 }%
4775 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4776 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4777 \setentrycounter[#1]{#2}%
4778 \@glxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4779 }

```

`xtr@display@loc` Patch to check for range formations.

```

4780 \def\@glxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4781 \ifx#1(\relax
4782 \glxtrdisplaystartloc{#2}{#3}%
4783 \else
4784 \ifx#1)\relax
4785 \glxtrdisplayendloc{#2}{#3}%
4786 \else
4787 \glxtrdisplaysingleloc{#1#2}{#3}%
4788 \fi
4789 \fi
4790 }

```

`isplayingleloc` Single location.

```

4791 \newcommand*{\glxtrdisplaysingleloc}[2]{%
4792 \csuse{#1}{#2}%
4793 }

```

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 `\glxtrlocrangefmt`.

`displaystartloc` Start of a location range.

```

4794 \newcommand*{\glxtrdisplaystartloc}[2]{%
4795 \edef\glxtrlocrangefmt{#1}%
4796 \ifx\glxtrlocrangefmt\empty
4797 \def\glxtrlocrangefmt{glsnumberformat}%
4798 \fi
4799 \expandafter\glxtrdisplaysingleloc
4800 \expandafter{\glxtrlocrangefmt}{#2}%
4801 }

```

`trdisplayendloc` End of a location range.

```

4802 \newcommand*{\glxtrdisplayendloc}[2]{%

```



```

4803 \edef\@glsxtr@tmp{#1}%
4804 \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
4805 \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4806 \else
4807   \GlossariesExtraWarning{Mismatched end location range
4808     (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
4809 \fi
4810 \expandafter\glsxtrdisplayendlochook\expandafter{\@glsxtr@tmp}{#2}%
4811 \expandafter\glsxtrdisplaysingleloc
4812   \expandafter{\glsxtrlocrangefmt}{#2}%
4813 \def\glsxtrlocrangefmt{}%
4814 }

```

`splayendlochook` Allow the user to hook into the end of range command.

```

4815 \newcommand*{\glsxtrdisplayendlochook}[2]{}

```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```

4816 \newcommand*{\glsxtrlocrangefmt}{}

```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4817 \renewcommand*{\setentrycounter}[2][{}]{%
4818   \def\glsxtrcounterprefix{#1}%
4819   \ifx\glsxtrcounterprefix\@empty
4820     \def\@glo@counterprefix{.}%
4821   \else
4822     \def\@glo@counterprefix{.#1.}%
4823   \fi
4824   \def\glsentrycounter{#2}%
4825 }

```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4826 \def\@gls@removespaces#1 #2\@nil{%
4827   \toks@=\expandafter{\the\toks@#1}%
4828   \ifx\@#2\%
4829     \edef\@glo@tmp{\the\toks@}%
4830     \ifx\@glo@tmp\empty
4831       \else

```

Expand location (just in case `\toks@` is needed for something else).

```

4832   \expandafter\glsxtrlocationhyperlink\expandafter
4833     \glsentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4834   \fi
4835 \else
4836   \@gls@ReturnAfterFi{%
4837     \@gls@removespaces#2\@nil
4838   }%
4839 \fi
4840 }

```

cationhyperlink

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

```
4841 \newcommand*{\glsxtrlocationhyperlink}[3]{%
4842   \ifdefvoid\glsxtrsupplocationurl
4843   {%
4844     \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4845   }%
4846   {%
4847     \hyperref{\glsxtrsupplocationurl}{#1#2#3}{#3}%
4848   }%
4849 }
```

supphypernumber

```
4850 \newcommand*{\glsxtrsupphypernumber}[1]{%
4851   {%
4852     \glshasattribute{\glscurrententrylabel}{externallocation}%
4853     {%
4854       \def\glsxtrsupplocationurl{%
4855         \glsgetattribute{\glscurrententrylabel}{externallocation}}%
4856     }%
4857     {%
4858       \def\glsxtrsupplocationurl{}%
4859     }%
4860     \glshypernumber{#1}%
4861   }%
4862 }
```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```
4863 \renewcommand{\@print@glossary}{%
4864   \makeatletter
4865   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4866   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4867   {%
4868     {\glsxtrNoGlossaryWarning{\@glo@type}}%
4869     \ifglxindy
4870       \ifcsundef{@xdy@\@glo@type @language}%
4871       {%
4872         \edef\@do@auxoutstuff{%
4873           \noexpand\AtEndDocument{%
4874             \noexpand\immediate\noexpand\write\@auxout{%
4875               \string\providecommand\string\@xdy@language[2]{}}%
4876             \noexpand\immediate\noexpand\write\@auxout{%
4877               \string\@xdy@language{\@glo@type}{\@xdy@main@language}}%

```

```

4878     }%
4879 }%
4880 }%
4881 {%
4882     \edef\@do@auxoutstuff{%
4883         \noexpand\AtEndDocument{%
4884             \noexpand\immediate\noexpand\write\@auxout{%
4885                 \string\providecommand\string\@xdylanguage[2]{}}%
4886             \noexpand\immediate\noexpand\write\@auxout{%
4887                 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4888                     @language\endcsname}}%
4889             }%
4890         }%
4891     }%
4892     \@do@auxoutstuff
4893     \edef\@do@auxoutstuff{%
4894         \noexpand\AtEndDocument{%
4895             \noexpand\immediate\noexpand\write\@auxout{%
4896                 \string\providecommand\string\@gls@codepage[2]{}}%
4897             \noexpand\immediate\noexpand\write\@auxout{%
4898                 \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4899             }%
4900         }%
4901     \@do@auxoutstuff
4902 \fi
4903 \renewcommand*{\@warn@nomakeglossaries}{%
4904     \GlossariesWarningNoLine{\string\makeglossaries\space
4905     hasn't been used,^^Jthe glossaries will not be updated}%
4906 }%
4907 }

```

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

oGlsWarningHead Header message.

```

4908 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4909     This document is incomplete. The external file associated with
4910     the glossary '#1' (which should be called \texttt{#2})
4911     hasn't been created.%
4912 }

```

rningEmptyStart No entries have been added to the glossary.

```

4913 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4914     This has probably happened because there are no entries defined
4915     in this glossary.%
4916 }

```

arningEmptyMain The default “main” glossary is empty.

```

4917 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4918     If you don't want this glossary,

```

```

4919 add \texttt{nomain} to your package option list when you load
4920 \texttt{glossaries-extra.sty}. For example:%
4921 }

```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```

4922 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4923 Did you forget to use \texttt{type=#1} when you defined your
4924 entries? If you tried to load entries into this glossary with
4925 \texttt{\string\loadglsentries} did you remember to use
4926 \texttt{[#1]} as the optional argument? If you did, check that
4927 the definitions in the file you loaded all had the type set
4928 to \texttt{\string\glsdefaulttype}.%
4929 }

```

arningCheckFile Advisory message to check the file contents.

```

4930 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4931 Check the contents of the file \texttt{#1}. If
4932 it's empty, that means you haven't indexed any of your entries in this
4933 glossary (using commands like \texttt{\string\gls} or
4934 \texttt{\string\glsadd}) so this list can't be generated.
4935 If the file isn't empty, the document build process hasn't been
4936 completed.%
4937 }

```

WarningAutoMake Message when automake option has been used.

```

4938 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4939 You may need to rerun \LaTeX. If you already have, it may be that
4940 \TeX's shell escape doesn't allow you to run
4941 \ifglxindy xindy\else makeindex\fi. Check the
4942 transcript file \texttt{\jobname.log}. If the shell escape is
4943 disabled, try one of the following:
4944
4945 \begin{itemize}
4946 \item Run the external (Lua) application:
4947
4948 \texttt{makeglossaries-lite \string"\jobname\string"}
4949
4950 \item Run the external (Perl) application:
4951
4952 \texttt{makeglossaries \string"\jobname\string"}
4953 \end{itemize}
4954
4955 Then rerun \LaTeX\ on this document.
4956 \GlossariesExtraWarning{Rerun required to build the
4957 glossary '#1' or check TeX's shell escape allows
4958 you to run \ifglxindy xindy\else makeindex\fi}%
4959 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```
4960 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4961   You need to either replace \texttt{\string\makenoidxglossaries}
4962   with \texttt{\string\makeglossaries} or replace
4963   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4964   \texttt{\string\printnoidxglossary}
4965   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4966   this document.%
4967 }
```

WarningBuildInfo Build advice.

```
4968 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4969   Try one of the following:
4970   \begin{itemize}
4971     \item Add \texttt{automake} to your package option list when you load
4972           \texttt{glossaries-extra.sty}. For example:
4973
4974           \texttt{\string\usepackage[automake]%
4975                 \glssopenbrace glossaries-extra\glsclosebrace}
4976
4977     \item Run the external (Lua) application:
4978
4979           \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4980
4981     \item Run the external (Perl) application:
4982
4983           \texttt{makeglossaries \string"\jobname\string"}
4984   \end{itemize}
4985
4986   Then rerun \LaTeX\ on this document.%
4987 }
```

trRecordWarning Paragraph for record=only.

```
4988 \newcommand{\GlsXtrRecordWarning}[1]{%
4989   \texttt{\string\printglossary} doesn't work
4990   with the \texttt{record=@glsxtr@record@setting} package option
4991   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4992   instead (or change the package option).%
4993 }
```

oGlsWarningTail Final paragraph.

```
4994 \newcommand{\GlsXtrNoGlsWarningTail}{%
4995   This message will be removed once the problem has been fixed.%
4996 }
```

GlsWarningNoOut No out file created. Build advice.

```
4997 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4998   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4999   \texttt{\string\makeglossaries} or you have used
```

```

5000 \texttt{\string\nofiles}. If this is just a draft version of the
5001 document, you can suppress this message using the
5002 \texttt{nomissingglstext} package option.%
5003 }

```

glossarywarning

```

5004 \newcommand*{\@glstr@defaultnoglossarywarning}[1]{%
5005   \glossarysection[\glossarytoctitle]{\glossarytitle}
5006   \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
5007   \par
5008   \glstrifemptyglossary{#1}%
5009   {%
5010     \GlsXtrNoGlsWarningEmptyStart\space
5011     \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
5012     \medskip
5013     \noindent\texttt{\string\usepackage[nomain\ifglstacronym ,acronym\fi]%
5014       \glstopenbrace glossaries-extra\glstclosebrace}
5015     \medskip
5016     }%
5017     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
5018   }%
5019   {%
5020     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
5021     {%
5022       \GlsXtrNoGlsWarningCheckFile
5023       {\jobname.\csname @glotype@\@glo@type @out\endcsname}
5024
5025       \ifglstautomake
5026
5027       \GlsXtrNoGlsWarningAutoMake{#1}
5028
5029       \else
5030
5031       \ifthenelse{\equal{#1}{main}}{%
5032       {%
5033         \GlsXtrNoGlsWarningEmptyMain\par
5034         \medskip
5035         \noindent\texttt{\string\usepackage[nomain]%
5036           \glstopenbrace glossaries-extra\glstclosebrace}
5037         \medskip
5038       }%
5039       }%
5040
5041       \ifdefequal\makeglossaries\@no@makeglossaries
5042       {%
5043         \GlsXtrNoGlsWarningMisMatch
5044       }%
5045       {%
5046         \GlsXtrNoGlsWarningBuildInfo

```

```

5047     }%
5048   \fi
5049 }%
5050 {%
5051   \GlsXtrNoGlsWarningNoOut
5052   {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
5053 }%
5054 }%
5055 \par
5056 \GlsXtrNoGlsWarningTail
5057 }

```

glossarywarning Warn about using `\printglossary` with `record`

```

5058 \newcommand*{\@glxtr@record@noglossarywarning}[1]{%
5059   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
5060   with record=\@glxtr@record@setting\space package option\MessageBreak(use
5061   \string\printunsrtglossary[type=#1])\MessageBreak
5062   instead (or change the package option)}%
5063   \glossarysection[\glossarytoctitle]{\glossarytitle}
5064   \GlsXtrRecordWarning{#1}
5065   \GlsXtrNoGlsWarningTail
5066 }

```

Provide some commands to accompany the `record` option for use with **bib2gls**.

ResourceOptions Default resource options.

```

5067 \newcommand*{\GlsXtrDefaultResourceOptions}{}

```

xtrresourcefile Since it's dangerous for an external application to create a file with a `.tex` extension, as from v1.11 this enforces a `.glstex` extension to avoid conflict.

```

5068 \newcommand*{\glxtrresourcefile}[2] []{%

```

The `record` option can't be set after this command.

```

5069   \disable@keys{glossaries-extra.sty}{record}%
5070   \glxtr@writefields
5071   \ifdefempty\GlsXtrDefaultResourceOptions
5072   {%
5073     \protected@write\@auxout{\glxtrresourceinit}%
5074     {\string\glxtr@resource{#1}{#2}}%
5075   }%
5076   {%
5077     \protected@write\@auxout{\glxtrresourceinit}%
5078     {\string\glxtr@resource{\GlsXtrDefaultResourceOptions,#1}{#2}}%
5079   }%
5080   \let\@glxtr@org@see@noindex\@gl@see@noindex
5081   \let\@gl@see@noindex\relax
5082   \IfFileExists{#2.glstex}%
5083   {%

```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```
5084 \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
5085 \makeatletter
5086 \@input{#2.glstex}%
5087 \@bibgls@restoreat
```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```
5088 \@glxtr@check@bibgls@nameref
5089 }%
5090 {%
5091 \GlossariesExtraWarning{No file '#2.glstex'}%
5092 }%
5093 \let\@gls@see@noindex\@glxtr@org@see@noindex
5094 }
5095 \@onlypreamble\glxtrresourcefile
```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```
5096 \newcommand{\@glxtr@check@bibgls@nameref}{%
5097 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
5098 \ifdef\bibglshrefchar
5099 {}%
5100 {%
5101 \GlossariesExtraWarning{record=nameref requires at least
5102 version 1.8 of bib2gls}%
5103 }%
5104 \fi
5105 \let\@glxtr@check@bibgls@nameref\relax
5106 }
```

trresourceinit Code used during the protected write operation.

```
5107 \newcommand*\glxtrresourceinit{}
```

trresourcecount

```
5108 \newcount\glxtrresourcecount
```

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```
5109 \newcommand*\GlsXtrLoadResources[1][\relax]{%
5110 \ifnum\glxtrresourcecount=0\relax
5111 \glxtrresourcefile[#1]{\jobname}%
5112 \else
5113 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
5114 \fi
5115 \advance\glxtrresourcecount by 1\relax
5116 }
```

glxtr@resource

```
5117 \newcommand*\glxtr@resource[2]{}
```



```

\glxtr@fields
5118 \newcommand*{\glxtr@fields}[1]{}

xtr@texencoding
5119 \newcommand*{\glxtr@texencoding}[1]{}

\glxtr@langtag
5120 \newcommand*{\glxtr@langtag}[1]{}

@pluralsuffixes
5121 \newcommand*{\glxtr@pluralsuffixes}[4]{}

tr@shortcutsval
5122 \newcommand*{\glxtr@shortcutsval}[1]{}

sxtr@linkprefix
5123 \newcommand*{\glxtr@linkprefix}[1]{}

xtr@writefields This information only needs to be written once, so disable it after it's been used.
5124 \newcommand*{\glxtr@writefields}{%

5125 \protected@write\@auxout{}%
5126 {\string\providecommand*{\string\glxtr@fields}[1]{}}%
5127 \protected@write\@auxout{}%
5128 {\string\providecommand*{\string\glxtr@resource}[2]{}}%
5129 \protected@write\@auxout{}%
5130 {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}}%
5131 \protected@write\@auxout{}%
5132 {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}%
5133 \protected@write\@auxout{}%
5134 {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}%
5135 \protected@write\@auxout{}{\string\glxtr@fields{\@gls@keymap}}%

5136 \protected@write\@auxout{}%
5137 {\string\providecommand*{\string\glxtr@record}[5]{}}%

5138 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
5139 \protected@write\@auxout{}%
5140 {\string\providecommand*{\string\glxtr@record@nameref}[8]{}}%
5141 \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 `\glxtrresourcefile`.

```

5142 \ifdef\CurrentTrackedLanguageTag
5143 {%
5144 \protected@write\@auxout{}%
5145 \string\glxtr@langtag{\CurrentTrackedLanguageTag}%

```

```

5146 }%
5147 {}%
5148 \protected@write\@auxout{}\string\glxtr@pluralsuffixes
5149   {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
5150   {\glxtrabbrvpluralsuffix}}%
5151 \ifdef\inputencodingname
5152 {%
5153   \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
5154 }%
5155 {%

```

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.)

```

5156   \ifpackageloaded{fontspec}%
5157   {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}}%
5158   {}%
5159 }%
5160 \protected@write\@auxout{}\string\glxtr@shortcutsval{\glxtr@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.

```

5161 \AtBeginDocument
5162   {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}}%
5163 \let\glxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. This has to be done before the aux file is opened (so package options `automake=immediate` and `automake=true` are identical if just bib2gls is used). The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```

5164 \ifglautomake
5165   \IfFileExists{\jobname.aux}%
5166   {\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`.

```

5167   \ifx\@gls@doautomake\@gls@doautomake@err
5168   \let\@gls@doautomake\relax
5169   \fi
5170 \fi

```

Check if `order=letter` has been used by mistake (but not if `record=alsoindex` has been used).

```

5171 \@glxtr@if@record@only
5172 {\ifdefstring{\glsorder}{letter}%
5173   {\GlossariesExtraWarningNoLine{Package option 'order=letter' isn't
5174     supported with 'record=\@glxtr@record@setting'. Use 'break-at=none'
5175     resource option instead}}%
5176   {}%
5177 }%
5178 {}%
5179 }

```

do@automake@err

```
5180 \newcommand*{\@gls@doautomake@err}{%
5181   \PackageError{glossaries}{You must use
5182   \string\makeglossaries\space with automake=true}
5183   {%
5184     Either remove the automake=true setting or
5185     add \string\makeglossaries\space to your document preamble.%
5186   }%
5187 }
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```
5188 \newcommand*{\glsxtr@record}[5]{}
```

@record@nameref Used with record=nameref to include current label information.

```
5189 \newcommand*{\glsxtr@record@nameref}[8]{}
```

r@counterrecord Aux file command.

```
5190 \newcommand*{\glsxtr@counterrecord}[3]{%
5191   \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
5192 }
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
5193 \newcommand*{\@glsxtr@counterrecordhook}{}%
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
5194 \newcommand*{\GlsXtrRecordCounter}[1]{%
5195   \@glsxtr@recordcounter{#1}%
5196 }
5197 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
5198 \newcommand*{\@glsxtr@docounterrecord}[1]{%
5199   \protected@write\@auxout{}{\string\glsxtr@counterrecord
5200     {\@gls@label}{#1}{\csuse{the#1}}}%
5201 }
```

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.

```
5202 \newcommand*{\glsxtrglossentry}[1]{%
5203   \glsxtrtitleorpdforheading
5204   {\@glsxtrglossentry{#1}}%
5205   {\glsentryname{#1}}%
```

```

5206 {\glxstrheadname{#1}}%
5207 }

```

`\glxtrglossentry` Another test is needed in case `\@glxtrglossentry` has been written to the table of contents.

```

5208 \newrobustcmd*{\@glxtrglossentry}[1]{%
5209   \glxstrtitleorpdforheading
5210   {%
5211     \glsdodefexists{#1}%
5212     {%
5213       \begingroup
5214         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
5215         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5216         \ifglshasparent{#1}%
5217           {\GlsXtrStandaloneSubEntryItem{#1}}%
5218           {\glentryitem{#1}}%
5219           \GlsXtrStandaloneEntryName{#1}%
5220         \endgroup
5221       }%
5222     }%
5223     {\glsentryname{#1}}%
5224     {\glxstrheadname{#1}}%
5225 }

```

`\glxtrStandaloneEntryName`

```

5226 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
5227   \glstarget{#1}{\glossentryname{#1}}%
5228 }

```

`\glxtrStandaloneGlossaryType` To make it easier to adjust the definition of `\currentglossary` within `\glxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

5229 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}

```

`\glxtrStandaloneSubEntryItem` Used for sub-entries in standalone format. The argument is the entry's label.

```

5230 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
5231   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}%
5232 }

```

`\glxtrglossentryother` As `\glxtrglossentry` 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.

```

5233 \newcommand*{\glxtrglossentryother}[3]{%
5234   \ifstrempy{#1}%
5235   {%
5236     \ifcsdef{glxtrhead#3}%
5237     {%

```

```

5238 \glxtrtitleorpdforheading
5239 {\@glxtrglossentryother{#2}{#3}{#1}}%
5240 {\@gls@entry@field{#2}{#3}}%
5241 {\csuse{glxtrhead#3}{#2}}%
5242 }%
5243 {%
5244 \glxtrtitleorpdforheading
5245 {\@glxtrglossentryother{#2}{#3}{#1}}%
5246 {\@gls@entry@field{#2}{#3}}%
5247 {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
5248 }%
5249 }%
5250 {%
5251 \glxtrtitleorpdforheading
5252 {\@glxtrglossentryother{#2}{#3}{#1}}%
5253 {\@gls@entry@field{#2}{#3}}%
5254 {#1}}%
5255 }%
5256 }

```

`glossentryother` As `\@glxtrglossentry` but uses a different field.

```

5257 \newrobustcmd*{\@glxtrglossentryother}[3]{%
5258 \glxtrtitleorpdforheading
5259 {%
5260 \glsdoifexists{#1}%
5261 {%
5262 \begingroup
5263 \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
5264 \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5265 \ifglshasparent{#1}%
5266 {\GlsXtrStandaloneSubEntryItem{#1}}%
5267 {\glsentryitem{#1}}%
5268 \GlsXtrStandaloneEntryOther{#1}%
5269 \endgroup
5270 }%
5271 }%
5272 {\@gls@entry@field{#1}{#2}}%
5273 {#3}}%
5274 }

```

`aloneEntryOther`

```

5275 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
5276 \glstarget{#1}{\glossentrynameother{#1}{#2}}%
5277 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting. Check for `\@printgloss@checkexists` which was introduced to glossaries v4.46.

```

5278 \ifdef\@printgloss@checkexists

```

```

5279 {
5280   \newcommand*{\printunsrtglossary}{%
5281     \let\@printgloss@checkexists\@printgloss@checkexists@allowignored
5282     \@ifstar\s@printunsrtglossary\@printunsrtglossary
5283   }
5284 }
5285 {
5286   \newcommand*{\printunsrtglossary}{%
5287     \@ifstar\s@printunsrtglossary\@printunsrtglossary
5288   }
5289 }

```

`\printunsrtglossary` Unstarred version.

```

5290 \newcommand*{\@printunsrtglossary}[1][\@printunsrtglossary]{%
5291   \@printglossary{type=\glsdefaulttype,#1}{\@printunsrtglossary}%
5292 }

```

`\printunsrtglossary` Starred version.

```

5293 \newcommand*{\s@printunsrtglossary}[2][\@printunsrtglossary]{%
5294   \begingroup
5295     #2%
5296     \@printglossary{type=\glsdefaulttype,#1}{\@printunsrtglossary}%
5297   \endgroup
5298 }

```

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

```

5299 \newcommand*{\printunsrtglossaries}{%
5300   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
5301 }

```

`\@printunsrtglossary`

```

5302 \newcommand*{\@printunsrtglossary}{%
5303   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5304   \glossarypreamble
5305   check for empty list
5306   \glsxtrifemptyglossary{\@glo@type}%
5307   {%
5308     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
5309   }%
5310   \key@ifundefined{glossentry}{group}%
5311   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
5312   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
5313   \def\@gls@currentlettergroup{}}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

5314 \def\@glxtr@doglossary{%
5315     \begin{theglossary}%
5316     \glossaryheader
5317     \glsresetentrylist
5318 }%
5319 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
5320 : \expandafter=\cename glolist@\@glo@type\endcsname\do{%
5321     \ifdefempty{\glscurrententrylabel}
5322     {}%
5323     {%

```

Provide a hook (for example to measure width).

```

5324     \let\glxtr@process\@firstofone
5325     \let\printunsrtglossaryskipentry
5326         \@glxtr@printunsrtglossaryskipentry
5327     \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5328     \glxtr@process
5329     {%
5330     \ifglxtr@printgloss@groups

```

This still uses `\ifglshasparent` to determine whether or not to check for a change in the letter group. (It doesn't take the level offset into account because `bib2gls` only saves the group information for parentless entries.)

```

5331     \ifglshasparent{\glscurrententrylabel}{}%
5332     {%
5333         \@glxtr@checkgroup\glscurrententrylabel
5334         \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
5335         {\@glxtr@groupheading}%
5336     }%
5337     \fi
5338     \eappto\@glxtr@doglossary{%
5339         \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5340     }%
5341 }%
5342 }%
5343 \appto\@glxtr@doglossary{\end{theglossary}}%
5344 \printunsrtglossarypredoglossary
5345 \@glxtr@doglossary
5346 }%
5347 \glossarypostamble
5348 }

```

`\rtinnerglossary` Similar to `\printunsrtglossary` but doesn't add the section heading, preamble, postamble or start and end of theglossary. Grouping is automatically applied so it may cause a problem within tabular-like environments. The beginning and ending of theglossary should be added around this command (but ensure the style has been set first). The simplest way of doing this is to place `\printunsrtinnerglossary` inside the `\printunsrtglossarywrap` environment.

```

5349 \newcommand*{\printunsrtinnerglossary}[3] [] {%

```

```

5350 \begingroup
5351 \def\@glxtr@printglossopts{#1}%
5352 \def\@glo@type{\glsdefaulttype}%
5353 \setkeys{printgloss}[title,toctitle,style,numberedsection,sort,label]{#1}%
5354 \let\currentglossary\@glo@type
5355 #2%
5356 \@print@unsrt@innerglossary
5357 #3%
5358 \endgroup
5359 }

```

srtglossarywrap

```

5360 \newenvironment{printunsrtglossarywrap}[1][ ]%
5361 {%
5362 \def\@glxtr@printglossopts{#1}%
5363 \def\@glo@type{\glsdefaulttype}%
5364 \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
5365 \def\glossarytoctitle{\glossarytitle}%
5366 \let\org@glossarytitle\glossarytitle
5367 \def\@glossarystyle{%
5368 \ifx\@glossary@default@style\relax
5369 \GlossariesWarning{No default glossary style provided \MessageBreak
5370 for the glossary '\@glo@type'. \MessageBreak
5371 Using deprecated fallback. \MessageBreak
5372 To fix this set the style with \MessageBreak
5373 \string\setglossarystyle\space or use the \MessageBreak
5374 style key=value option}%
5375 \fi
5376 }%
5377 \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
5378 \let\@org@glossaryentrynumbers\glossaryentrynumbers
5379 \@printgloss@setsort
5380 \setkeys{printgloss}{#1}%

```

The type key simply allows the title to be set if the title key isn't supplied.

```

5381 \ifglossaryexists*\@glo@type}%
5382 {%
5383 \ifx\glossarytitle\org@glossarytitle
5384 \else
5385 \expandafter\let\csname @glotype@\@glo@type @title\endcsname
5386 \glossarytitle
5387 \fi
5388 \let\currentglossary\@glo@type
5389 }%
5390 {}%
5391 \let\org@glossaryentrynumbers\glossaryentrynumbers
5392 \let\glsnonextpages\@glsnonextpages
5393 \let\glsnextpages\@glsnextpages
5394 \let\nopostdesc\@nopostdesc
5395 \gls@dotoc@title

```



```

5396 \@glossarystyle
5397 \let\gls@org@glossaryentryfield\glossentry
5398 \let\gls@org@glossarysubentryfield\subglossentry
5399 \renewcommand{\glossentry}[1]{%
5400   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5401   \gls@org@glossaryentryfield{##1}%
5402 }%
5403 \renewcommand{\subglossentry}[2]{%
5404   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5405   \gls@org@glossarysubentryfield{##1}{##2}%
5406 }%
5407 \@gls@preglossaryhook
5408 \glossarysection[\glossarytoctitle]{\glossarytitle}%
5409 \glossarypreamble
5410 \begin{theglossary}%
5411 \glossaryheader
5412 \glsresetentrylist
5413 }%
5414 {%
5415 \end{theglossary}%
5416 \glossarypostamble
5417 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
5418 \global\let\warn@noprintglossary\relax
5419 }

```

t@innerglossary This is much like \@print@unsrt@innerglossary but only contains what would normally be the content of the theglossary.

```

5420 \newcommand*{\@print@unsrt@innerglossary}{%
  No section header or preamble.
5421 \glsxtrifemptyglossary{\@glo@type}%
5422 {%
5423   \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
5424 }%
5425 {%
5426   \key@ifundefined{glossentry}{group}%
5427   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
5428   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
5429   \def\@gls@currentlettergroup{}}%

```

No header or reset.

```

5430 \def\@glsxtr@doglossary{%
5431 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
5432 : \expandafter=\csname glo@list@\@glo@type\endcsname\do{%
5433 \ifdefempty{\glscurrententrylabel}
5434 {}%
5435 {%

```

Provide a hook (for example to measure width).

```

5436 \let\glsxtr@process\@firstofone

```

```

5437 \let\printunsrtglossaryskipentry
5438 \@glxtr@printunsrtglossaryskipentry
5439 \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

5440 \glxtr@process
5441 {%
5442 \ifglxtr@printgloss@groups

```

This still uses \ifglshasparent to determine whether or not to check for a change in the letter group. (It doesn't take the level offset into account because bib2gls only saves the group information for parentless entries.)

```

5443 \ifglshasparent{\glscurrententrylabel}{}%
5444 {%
5445 \@glxtr@checkgroup\glscurrententrylabel
5446 \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
5447 {\@glxtr@groupheading}%
5448 }%
5449 \fi
5450 \eappto\@glxtr@doglossary{%
5451 \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5452 }%
5453 }%
5454 }%
5455 \printunsrtglossarypredoglossary
5456 \@glxtr@doglossary
5457 }%

```

No postamble.

```

5458 }

```

entryprocesshook

```

5459 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

ossaryskipentry

```

5460 \newcommand*{\printunsrtglossaryskipentry}{%
5461 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
5462 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5463 }

```

entryprocesshook

```

5464 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
5465 \let\glxtr@process\@gobble
5466 }

```

rypredoglossary

```

5467 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```

5468 \newcommand{\@printunsrtglossary@handler}[1]{%
5469   \xdef\glscurrententrylabel{#1}%
5470   \printunsrtglossaryhandler\glscurrententrylabel
5471 }

```

glossaryhandler

```

5472 \newcommand{\printunsrtglossaryhandler}[1]{%
5473   \glsxtrunsrtdo{#1}%
5474 }

```

triflabelinlist

```
\glsxtriflabelinlist{<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.

```

5475 \newrobustcmd*{\glsxtriflabelinlist}[4]{%
5476   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
5477   \@glxtr@doiflabelinlist{#3}{#4}%
5478 }

```

srtglossaryunit

```

5479 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5480   \s@printunsrtglossary[type=\glstypedefaulttype,#1]{%
5481     \printunsrtglossaryunitsetup{#2}%
5482   }%
5483 }

```

ossaryunitsetup

```

5484 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
5485   \renewcommand{\printunsrtglossaryhandler}[1]{%
5486     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5487     {\glsxtrunsrtdo{##1}}%
5488     {}%
5489   }%

```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glolinkprefix`. The `\@gobble` part discards `\glolinkprefix`.

```

5490   \ifcsundef{theH#1}%
5491   {%
5492     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
5493   }%
5494   {%
5495     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
5496   }%
5497   \renewcommand*{\glossarysection}[2][ ]{}%

```

```

5498 \appto\glossarypostamble{\glspare\medskip\glspare}%
5499 }

```

srtglossaryunit

```

5500 \newcommand{\printnoop@unsrtglossaryunit}[2][]{%
5501 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5502 requires the record=only or record=alsoindex package option}{}%
5503 }

```

t@getgrouptitle

```

5504 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
5505 \protected@edef\@glxtr@titlelabel{glxtr@grouptitle@#1}%
5506 \@onelevel@sanitize\@glxtr@titlelabel
5507 \ifcsdef{\@glxtr@titlelabel}
5508 {\letcs{#2}{\@glxtr@titlelabel}}%
5509 {\def#2{#1}}%
5510 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

5511 \newcommand{\glxtrunsrtdo}{\@glxtr@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).

```

5512 \newcommand*{\glxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glxtr@doglossary is being constructed rather than in the handler.

lsxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glxtr@groupheading, which will be empty if no heading is required.

```

5513 \newcommand*{\@glxtr@checkgroup}[1]{%
5514 \def\@glxtr@groupheading{}%
5515 \key@ifundefined{glossentry}{group}%
5516 {%
5517 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
5518 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5519 }%
5520 {%
5521 \protected@edef\@glo@thislettergrp{%
5522 \csuse{glo@\glsdetoklabel{#1}@\glxtrgroupfield}}%
5523 }%
5524 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5525 {}%

```

```

5526 {%
5527   \ifdefempty{\@gls@currentlettergroup}{}%
5528   {\def\@glstr@groupheading{\glsgroupskip}}%
5529   \eappto\@glstr@groupheading{%
5530     \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
5531   }%
5532 }%
5533 \let\@gls@currentlettergroup\@glo@thislettergrp
5534 }

```

trLocationField Stores the internal name of the location field.

```
5535 \newcommand*{\GlsXtrLocationField}{location}
```

glstr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present, but also need to check for the group field.

```

5536 \newcommand{\@glstr@noidx@do}[1]{%
5537   \ifglstryexists{#1}%
5538   {%
5539     \global\letcs{\@gls@loclist}{glo@\glstetoklabel{#1}@loclist}%
5540     \global\letcs{\@gls@location}{glo@\glstetoklabel{#1}@\GlsXtrLocationField}%

```

Use level number to determine whether or not this entry has a parent.

```

5541     \gls@level=\numexpr\csuse{glo@\glstetoklabel{#1}@level}+\@glstr@leveloffset\relax
5542     \ifnum\gls@level>0
5543       \let\@glstr@ifischild\@firstoftwo
5544     \else
5545       \let\@glstr@ifischild\@secondoftwo
5546     \fi

```

Some glossary styles (such as topicmcols) save the level using \def so make sure \gls@level is expanded before being passed to \subglossentry.

```

5547   \@glstr@ifischild
5548   {%
5549     \ifdefvoid{\@gls@location}%
5550     {%
5551       \ifdefvoid{\@gls@loclist}%
5552       {%
5553         \expandafter\subglossentry\expandafter{\number\gls@level}{#1}{}%
5554       }%
5555       {%
5556         \expandafter\subglossentry\expandafter{\number\gls@level}{#1}%
5557       }%
5558       \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5559     }%
5560   }%
5561 }%
5562 {%
5563   \expandafter\subglossentry\expandafter
5564   {\number\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5565 }%

```

```

5566 }%
5567 {%

5568 \ifdefvoid{\@gls@location}%
5569 {%
5570 \ifdefvoid{\@gls@loclist}
5571 {%
5572 \glossentry{#1}{}%
5573 }%
5574 {%
5575 \glossentry{#1}%
5576 {%
5577 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5578 }%
5579 }%
5580 }%
5581 {%
5582 \glossentry{#1}%
5583 {%
5584 \glossaryentrynumbers{\@gls@location}%
5585 }%
5586 }%
5587 }%
5588 }%
5589 {}%
5590 }

```

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`

```

5591 \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`

```

5592 \newcommand*{\@glsxtr@providenewgls}{%
5593 \protected@write\@auxout{}\string\providecommand{\string\@glsxtr@newglslike}[2]{}%
5594 \let\@glsxtr@providenewgls\relax
5595 }

```

`identifyglslike` Identify the command given in the second argument for the benefit of **bib2gls**.

```

5596 \newcommand{\glsxtridentifyglslike}[2]{%
5597 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
5598 }%
5599 {%

```

```

5600 \@glsxtr@providenewgls
5601 \protected@write\@auxout{}\string\@glsxtr@newglslike{#1}\string#2}}%
5602 }%
5603 }

```

\@glsxtrnewgls

\glsxtrnewgls[*<options>*]{*<prefix>*}{*<cs>*}{*<inner cs name>*}

```

5604 \newcommand*\@glsxtrnewgls[4]{%
5605 \ifdef{#3}%
5606 {%
5607 \PackageError{glossaries-extra}{Command \string#3\space already
5608 defined}{}%
5609 }%
5610 {%

```

Write information to the aux file for bib2gls.

```

5611 \glsxtridentifyglslike{#2}{#3}%
5612 \ifcsdef{@#4like@#2}%
5613 {%
5614 \advance\@glsxtrnewgls@inner by \@ne
5615 \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
5616 }%
5617 {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
5618 \expandafter\newrobustcmd\expandafter*\expandafter
5619 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
5620 \ifstrempy{#1}%
5621 {%
5622 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
5623 \new@ifnextchar[%
5624 {\csname @#4@\endcsname{##1}{#2##2}}%
5625 {\csname @#4@\endcsname{##1}{#2##2} []}%
5626 }%
5627 }%
5628 {%
5629 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
5630 \new@ifnextchar[%
5631 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5632 {\csname @#4@\endcsname{#1,##1}{#2##2} []}%
5633 }%
5634 }%
5635 }%
5636 }

```

\glsxtrnewgls

```
\glxstrnewgls[<options>]{<prefix>}{<cs>}
```

The first argument prepends to the options and the second argument is the prefix.

```
5637 \newrobustcmd*{\glxstrnewgls}[3] [] {%
5638   \@glxstrnewgls{#1}{#2}{#3}{gls}%
5639 }
```

glxstrnewglslike Provide a way to conveniently define commands that behave like `\gls`, `\glsp1`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5640 \newrobustcmd*{\glxstrnewglslike}[6] [] {%
5641   \@glxstrnewgls{#1}{#2}{#3}{gls}%
5642   \@glxstrnewgls{#1}{#2}{#4}{glsp1}%
5643   \@glxstrnewgls{#1}{#2}{#5}{Gls}%
5644   \@glxstrnewgls{#1}{#2}{#6}{Glspl}%
5645 }
```

glxstrnewGLSlike Provide a way to conveniently define commands that behave like `\GLS`, `\GLSp1` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
5646 \newrobustcmd*{\glxstrnewGLSlike}[4] [] {%
5647   \@glxstrnewgls{#1}{#2}{#3}{GLS}%
5648   \@glxstrnewgls{#1}{#2}{#4}{GLSp1}%
5649 }
```

\glxstrnewrgls As `\glxstrnewgls` but for `\rgls`.

```
5650 \newrobustcmd*{\glxstrnewrgls}[3] [] {%
5651   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5652 }
```

glxstrnewrglslike As `\glxstrnewglslike` but for `\rgls` etc.

```
5653 \newrobustcmd*{\glxstrnewrglslike}[6] [] {%
5654   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5655   \@glxstrnewgls{#1}{#2}{#4}{rglsp1}%
5656   \@glxstrnewgls{#1}{#2}{#5}{rGls}%
5657   \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
5658 }
```

glxstrnewrGLSlike As `\glxstrnewGLSlike` but for `\rGLS` etc.

```
5659 \newrobustcmd*{\glxstrnewrGLSlike}[4] [] {%
5660   \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
5661   \@glxstrnewgls{#1}{#2}{#4}{rGLSp1}%
5662 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
5663 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
5664   \ifcsdef{glo@glstdetoklabel{#1}@recordcount}%
```



```

5665 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
5666 {0}%
5667 }

```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```

5668 \newcommand*{\GlsXtrRecordCount}[2]{%
5669 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
5670 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
5671 {0}%
5672 }

```

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 `\glsxtrdetoklocation` can be set to something sensible.

```

5673 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
5674 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}}%
5675 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}\endcsname}%
5676 {0}%
5677 }

```

trdetoklocation

```

5678 \newcommand*{\glsxtrdetoklocation}[1]{#1}

```

ablerecordcount

```

5679 \newcommand*{\glsxtrenablerecordcount}{%
5680 \renewcommand*{\gls}{\rgls}%
5681 \renewcommand*{\Gls}{\rGls}%
5682 \renewcommand*{\glspl}{\rglspl}%
5683 \renewcommand*{\Glspl}{\rGlspl}%
5684 \renewcommand*{\GLS}{\rGLS}%
5685 \renewcommand*{\GLSpl}{\rGLSpl}%
5686 }

```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```

5687 \newcommand*{\glsxtrrecordtriggervalue}[1]{%
5688 \GlsXtrTotalRecordCount{#1}%
5689 }

```

dCountAttribute

```

5690 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
5691 \@for\@glsxtr@cat:=#1\do
5692 {%
5693 \ifdefempty{\@glsxtr@cat}{}%
5694 {%
5695 \glssetcategoryattribute{\@glsxtr@cat}{recordcount}{#2}%
5696 }%

```

```
5697 }%
5698 }
```

ifrecordtrigger

$\backslash\text{glxstrifrecordtrigger}\{\langle label \rangle\}\{\langle trigger\ format \rangle\}\{\langle normal \rangle\}$
--

```
5699 \newcommand*{\glxstrifrecordtrigger}[3]{%
5700   \glshasattribute{#1}{recordcount}%
5701   {%
5702     \ifnum\glxstrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
5703     #3%
5704     \else
5705     #2%
5706     \fi
5707   }%
5708   {#3}%
5709 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
5710 \newcommand*{\@glxstr@rglstrigger@record}[3]{%
5711   \edef\glslabel{\glsdetoklabel{#2}}%
5712   \let\@gls@link@label\glslabel
5713   \def\@glxstr@thevalue{%
5714     \def\@glxstr@theHvalue{\@glxstr@thevalue}%
5715     \def\@glsnumberformat{glstriggerrecordformat}%
5716     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5717     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5718     \def\@glxstr@thevalue{%
5719       \def\@glxstr@theHvalue{\@glxstr@thevalue}%
5720       \glxstrinitwrgloss
5721       \glslinkpresetkeys
5722       \setkeys{glslink}{#1}%
5723       \glslinkpostsetkeys
5724       \ifdefempty{\@glxstr@thevalue}%
5725       {%
5726         \@gls@saveentrycounter
5727       }%
5728       {%
5729         \let\theglsentrycounter\@glxstr@thevalue
5730         \def\theHglentrycounter{\@glxstr@theHvalue}%
5731       }%
5732       \ifglxstrinitwrglossbefore
5733       \@do@wrglossary{#2}%
5734       \fi
5735       #3%
5736       \ifglxstrinitwrglossbefore
```

```

5737 \else
5738   \@do@wrglossary{#2}%
5739 \fi
5740 \ifKV@glslink@local
5741   \glsllocalunset{#2}%
5742 \else
5743   \glunset{#2}%
5744 \fi
5745 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5746 \newcommand*{\glstriggerrecordformat}[1]{

```

\rgls

```

5747 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

5748 \newcommand*{\@rgls}[2] [] {%
5749   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2} []}%
5750 }

```

\@rgls@

```

5751 \def\@rgls@#1#2[#3] {%
5752   \glxtrifrecordtrigger{#2}%
5753   {%
5754     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5755   }%
5756   {%
5757     \@gls@{#1}{#2}[#3]%
5758   }%
5759 }%

```

\rglspl

```

5760 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

```

\@rglspl

```

5761 \newcommand*{\@rglspl}[2] [] {%
5762   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2} []}%
5763 }

```

\@rglspl@

```

5764 \def\@rglspl@#1#2[#3] {%
5765   \glxtrifrecordtrigger{#2}%
5766   {%
5767     \@glxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5768   }%
5769   {%

```

```

5770 \@glsp1@{#1}-{#2}[#3]%
5771 }%
5772 }%

```

\rGls

```

5773 \newrobustcmd*{\rGls}{\@gls@hyp@opt\rGls}

```

\@rGls

```

5774 \newcommand*{\@rGls}[2][{}]{%
5775 \new@ifnextchar[{\@rGls@{#1}-{#2}}{\@rGls@{#1}-{#2}[]}%
5776 }

```

\@rGls@

```

5777 \def\@rGls@#1#2[#3]{%
5778 \glstrifrecordtrigger{#2}%
5779 {%
5780 \@glstr@rglstrigger@record{#1}-{#2}{\rGlsformat{#2}{#3}}%
5781 }%
5782 {%
5783 \@Gls@{#1}-{#2}[#3]%
5784 }%
5785 }%

```

\rGlspl

```

5786 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

```

\@rGlspl

```

5787 \newcommand*{\@rGlspl}[2][{}]{%
5788 \new@ifnextchar[{\@rGlspl@{#1}-{#2}}{\@rGlspl@{#1}-{#2}[]}%
5789 }

```

\@rGlspl@

```

5790 \def\@rGlspl@#1#2[#3]{%
5791 \glstrifrecordtrigger{#2}%
5792 {%
5793 \@glstr@rglstrigger@record{#1}-{#2}{\rGlsplformat{#2}{#3}}%
5794 }%
5795 {%
5796 \@Glspl@{#1}-{#2}[#3]%
5797 }%
5798 }%

```

\rGLS

```

5799 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

```

\@rGLS

```

5800 \newcommand*{\@rGLS}[2][{}]{%
5801 \new@ifnextchar[{\@rGLS@{#1}-{#2}}{\@rGLS@{#1}-{#2}[]}%
5802 }

```

\@rGLS@

```
5803 \def\@rGLS@#1#2[#3]{%
5804   \glxtrifrecordtrigger{#2}%
5805   {%
5806     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
5807   }%
5808   {%
5809     \@GLS@{#1}{#2}[#3]%
5810   }%
5811 }%
```

\rGLSpl

```
5812 \newrobustcmd*{\rGLSpl}{\@glshyp@opt\@rGLSpl}
```

\@rGLSpl

```
5813 \newcommand*{\@rGLSpl}[2][{}]{%
5814   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[]}%
5815 }
```

\@rGLSpl@

```
5816 \def\@rGLSpl@#1#2[#3]{%
5817   \glxtrifrecordtrigger{#2}%
5818   {%
5819     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5820   }%
5821   {%
5822     \@GLSpl@{#1}{#2}[#3]%
5823   }%
5824 }%
```

\rglsformat

```
5825 \newcommand*{\rglsformat}[2]{%
5826   \glshifregular{#1}
5827   {\glsentryfirst{#1}}%
5828   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5829 }
```

\rglsplformat

```
5830 \newcommand*{\rglsplformat}[2]{%
5831   \glshifregular{#1}
5832   {\glsentryfirstplural{#1}}%
5833   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5834 }
```

\rGlsformat

```
5835 \newcommand*{\rGlsformat}[2]{%
5836   \glshifregular{#1}
5837   {\rGlsentryfirst{#1}}%
```

```

5838 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}\#2%
5839 }

```

`\rGlsplformat`

```

5840 \newcommand*{\rGlsplformat}[2]{%
5841   \glsifregular{#1}
5842   {\Glsentryfirstplural{#1}}%
5843   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}\#2%
5844 }

```

`\rGLSformat`

```

5845 \newcommand*{\rGLSformat}[2]{%
5846   \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5847 }

```

`\rGLSplformat`

```

5848 \newcommand*{\rGLSplformat}[2]{%
5849   \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5850 }

```

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`.)

`\do@inc@linkcount` This performs the actual incrementing and counter definition. The counter is given by `\c@glsxtr@linkcount@<label>` where `<label>` is the entry’s label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```

5851 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
5852   \glsifattribute{\glslabel}{linkcount}{true}%
5853   {%

```

Does the counter exist?

```

5854   \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5855   {%

```

Counter doesn’t exist, so define it.

```

5856     \newcounter{glsxtr@linkcount@\glslabel}%

```

If `linkcountmaster` is set, add to counter reset.

```

5857     \glshasattribute{\glslabel}{linkcountmaster}%
5858     {%

```

Need to ensure values are fully expanded.

```

5859      \begingroup
5860      \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@glslabel}%
5861      {\glsggetattribute{glslabel}{linkcountmaster}}}%
5862      \x
5863      }%
5864      {}%
5865      }%

```

Increment counter:

```

5866 \glxtrinclinkcounter{glxtr@linkcount@glslabel}%
5867 }%
5868 {}%
5869 }

```

`\glxtrinclinkcounter` May be redefined to use `\refstepcounter` if required.

```

5870 \newcommand*{\glxtrinclinkcounter}[1]{\stepcounter{#1}}

```

`\glxtrlinkcounterValue` Expands to the associated link counter register or 0 if not defined.

```

5871 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5872 \ifcsundef{c@glxtr@linkcount@#1}{0}{\csname c@glxtr@linkcount@#1\endcsname}%
5873 }

```

`\glxtrTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```

5874 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5875 \ifcsundef{theglxtr@linkcount@#1}{0}%
5876 {\csname theglxtr@linkcount@#1\endcsname}%
5877 }

```

`\ifLinkCounterDef` Tests if the counter has been defined

```

5878 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5879 \ifcsundef{theglxtr@linkcount@#1}{#3}{#2}%
5880 }

```

`\glxtrLinkCounterName` Expands to the associated link counter name. (No check for existence.)

```

5881 \newcommand*{\GlsXtrLinkCounterName}[1]{glxtr@linkcount@#1}

```

`\enableLinkCounting`

`\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```

5882 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5883 \let\glxtr@inc@linkcount\@glxtr@do@inc@linkcount
5884 \@for\@glxtr@label:=#2\do
5885 {%
5886 \glsssetcategoryattribute{\@glxtr@label}{linkcount}{true}%

```

```

5887 \ifstrempy{#1}{}%
5888 {%
5889 \ifcsundef{c@#1}%
5890 {\@nocounterr{#1}}%
5891 {\glissetcategoryattribute{\@glstr@label}{linkcountmaster}{#1}}%
5892 }%
5893 }%
5894 }
5895 \@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.

```

5896 \@ifpackageloaded{glossaries-accsupp}
5897 {

```

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

```

\glsaccessname  Display the name value (no link and no check for existence).
5898 \newcommand*{\glsaccessname}[1]{%
5899 \glsnameaccessdisplay
5900 {%
5901 \glstryname{#1}%
5902 }%
5903 {#1}%
5904 }

```

```

\Glsaccessname  Display the name value (no link and no check for existence) with the first letter converted to
upper case.
5905 \newcommand*{\Glsaccessname}[1]{%
5906 \glsnameaccessdisplay
5907 {%
5908 \Glsstryname{#1}%
5909 }%
5910 {#1}%
5911 }

```

```

\GLSaccessname  Display the name value (no link and no check for existence) converted to upper case.
5912 \newcommand*{\GLSaccessname}[1]{%
5913 \glsnameaccessdisplay
5914 {%
5915 \mfirstucMakeUppercase{\glstryname{#1}}%
5916 }%
5917 {#1}%
5918 }

```


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

```
5919 \newcommand*{\glsaccesstext}[1]{%
5920   \glstextaccessdisplay
5921   {%
5922     \glstrytext{#1}%
5923   }%
5924   {#1}%
5925 }
```

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

```
5926 \newcommand*{\Glsaccesstext}[1]{%
5927   \glstextaccessdisplay
5928   {%
5929     \Glsstrytext{#1}%
5930   }%
5931   {#1}%
5932 }
```

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

```
5933 \newcommand*{\GLSaccesstext}[1]{%
5934   \glstextaccessdisplay
5935   {%
5936     \mfirstucMakeUppercase{\glstrytext{#1}}%
5937   }%
5938   {#1}%
5939 }
```

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

```
5940 \newcommand*{\glsaccessplural}[1]{%
5941   \glspluralaccessdisplay
5942   {%
5943     \glstryplural{#1}%
5944   }%
5945   {#1}%
5946 }
```

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

```
5947 \newcommand*{\Glsaccessplural}[1]{%
5948   \glspluralaccessdisplay
5949   {%
5950     \Glsstryplural{#1}%
5951   }%
5952   {#1}%
5953 }
```

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

```

5954 \newcommand*{\GLSaccessplural}[1]{%
5955   \glspluralaccessdisplay
5956   {%
5957     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5958   }%
5959   {#1}%
5960 }

```

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

```

5961 \newcommand*{\glsaccessfirst}[1]{%
5962   \glsfirstaccessdisplay
5963   {%
5964     \glsentryfirst{#1}%
5965   }%
5966   {#1}%
5967 }

```

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

```

5968 \newcommand*{\Glsaccessfirst}[1]{%
5969   \glsfirstaccessdisplay
5970   {%
5971     \Glsentryfirst{#1}%
5972   }%
5973   {#1}%
5974 }

```

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

```

5975 \newcommand*{\GLSaccessfirst}[1]{%
5976   \glsfirstaccessdisplay
5977   {%
5978     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
5979   }%
5980   {#1}%
5981 }

```

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

```

5982 \newcommand*{\glsaccessfirstplural}[1]{%
5983   \glsfirstpluralaccessdisplay
5984   {%
5985     \glsentryfirstplural{#1}%
5986   }%
5987   {#1}%
5988 }

```

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

```

5989 \newcommand*{\Glsaccessfirstplural}[1]{%

```

```

5990   \glsfirstpluralaccessdisplay
5991   {%
5992       \Glsentryfirstplural{#1}%
5993   }%
5994   {#1}%
5995 }

```

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

```

5996   \newcommand*{\Glsaccessfirstplural}[1]{%
5997       \glsfirstpluralaccessdisplay
5998       {%
5999           \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
6000       }%
6001       {#1}%
6002   }

```

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

```

6003   \newcommand*{\glsaccesssymbol}[1]{%
6004       \glssymbolaccessdisplay
6005       {%
6006           \glsentrysymbol{#1}%
6007       }%
6008       {#1}%
6009   }

```

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

```

6010   \newcommand*{\Glsaccesssymbol}[1]{%
6011       \glssymbolaccessdisplay
6012       {%
6013           \Glsentrysymbol{#1}%
6014       }%
6015       {#1}%
6016   }

```

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

```

6017   \newcommand*{\GLSaccesssymbol}[1]{%
6018       \glssymbolaccessdisplay
6019       {%
6020           \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
6021       }%
6022       {#1}%
6023   }

```

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

```

6024   \newcommand*{\glsaccesssymbolplural}[1]{%
6025       \glssymbolpluralaccessdisplay
6026       {%

```

```

6027     \glentrysymbolplural{#1}%
6028   }%
6029   {#1}%
6030 }

```

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

```

6031 \newcommand*{\Glsaccesssymbolplural}[1]{%
6032   \glssymbolpluralaccessdisplay
6033   {%
6034     \glentrysymbolplural{#1}%
6035   }%
6036   {#1}%
6037 }

```

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

```

6038 \newcommand*{\GLSaccesssymbolplural}[1]{%
6039   \glssymbolpluralaccessdisplay
6040   {%
6041     \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
6042   }%
6043   {#1}%
6044 }

```

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

```

6045 \newcommand*{\glsaccessdesc}[1]{%
6046   \glsdescriptionaccessdisplay
6047   {%
6048     \glentrydesc{#1}%
6049   }%
6050   {#1}%
6051 }

```

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

```

6052 \newcommand*{\GLSaccessdesc}[1]{%
6053   \glsdescriptionaccessdisplay
6054   {%
6055     \glentrydesc{#1}%
6056   }%
6057   {#1}%
6058 }

```

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

```

6059 \newcommand*{\GLSAccessdesc}[1]{%
6060   \glsdescriptionaccessdisplay
6061   {%
6062     \mfirstucMakeUppercase{\glentrydesc{#1}}%

```

```

6063     }%
6064     {#1}%
6065 }

```

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

```

6066 \newcommand*{\glsaccessdescplural}[1]{%
6067   \glsdescriptionpluralaccessdisplay
6068   {%
6069     \glsentrydescplural{#1}%
6070   }%
6071   {#1}%
6072 }

```

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

```

6073 \newcommand*{\Glsaccessdescplural}[1]{%
6074   \glsdescriptionpluralaccessdisplay
6075   {%
6076     \Glsentrydescplural{#1}%
6077   }%
6078   {#1}%
6079 }

```

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

```

6080 \newcommand*{\Glsaccessdescplural}[1]{%
6081   \glsdescriptionpluralaccessdisplay
6082   {%
6083     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
6084   }%
6085   {#1}%
6086 }

```

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

```

6087 \newcommand*{\glsaccessshort}[1]{%
6088   \glsshortaccessdisplay
6089   {%
6090     \glsentryshort{#1}%
6091   }%
6092   {#1}%
6093 }

```

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

```

6094 \newcommand*{\Glsaccessshort}[1]{%
6095   \glsshortaccessdisplay
6096   {%
6097     \Glsentryshort{#1}%
6098   }%

```

```

6099     {#1}%
6100 }

```

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

```

6101 \newcommand*{\GLSaccessshort}[1]{%
6102   \glsshortaccessdisplay
6103   {%
6104     \mfirstucMakeUppercase{\glsentryshort{#1}}%
6105   }%
6106   {#1}%
6107 }

```

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

```

6108 \newcommand*{\lsaccessshortpl}[1]{%
6109   \glsshortpluralaccessdisplay
6110   {%
6111     \glsentryshortpl{#1}%
6112   }%
6113   {#1}%
6114 }

```

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

```

6115 \newcommand*{\lSaccessshortpl}[1]{%
6116   \glsshortpluralaccessdisplay
6117   {%
6118     \lSentryshortpl{#1}%
6119   }%
6120   {#1}%
6121 }

```

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

```

6122 \newcommand*{\LSaccessshortpl}[1]{%
6123   \glsshortpluralaccessdisplay
6124   {%
6125     \mfirstucMakeUppercase{\lSentryshortpl{#1}}%
6126   }%
6127   {#1}%
6128 }

```

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

```

6129 \newcommand*{\glsaccesslong}[1]{%
6130   \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
6131 }

```

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

```

6132
6133 \newcommand*{\Glsaccesslong}[1]{%

```

```

6134     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
6135 }

```

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

```

6136 \newcommand*{\GLSaccesslong}[1]{%
6137     \glslongaccessdisplay
6138     {%
6139         \mfirstucMakeUppercase{\glsentrylong{#1}}%
6140     }%
6141     {#1}%
6142 }

```

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

```

6143 \newcommand*{\glsaccesslongpl}[1]{%
6144     \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
6145 }

```

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

```

6146
6147 \newcommand*{\GLsaccesslongpl}[1]{%
6148     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
6149 }

```

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

```

6150 \newcommand*{\GLSaccesslongpl}[1]{%
6151     \glslongpluralaccessdisplay
6152     {%
6153         \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
6154     }%
6155     {#1}%
6156 }

```

Keys for accessibility support.

```

6157 \define@key{glsxtrabbrv}{access}{%
6158     \def\@gls@nameaccess{#1}%
6159 }
6160 \define@key{glsxtrabbrv}{textaccess}{%
6161     \def\@gls@textaccess{#1}%
6162 }
6163 \define@key{glsxtrabbrv}{pluralaccess}{%
6164     \def\@gls@pluralaccess{#1}%
6165 }
6166 \define@key{glsxtrabbrv}{firstaccess}{%
6167     \def\@gls@firstaccess{#1}%
6168 }
6169 \define@key{glsxtrabbrv}{firstpluralaccess}{%
6170     \def\@gls@firstpluralaccess{#1}%
6171 }

```

```

6172 \define@key{glsxtrabrv}{shortaccess}{%
6173   \def\@gls@shortaccess{#1}%
6174 }
6175 \define@key{glsxtrabrv}{shortpluralaccess}{%
6176   \def\@gls@shortaccesspl{#1}%
6177 }
6178 \define@key{glsxtrabrv}{longaccess}{%
6179   \def\@gls@longaccess{#1}%
6180 }
6181 \define@key{glsxtrabrv}{shortlonglaccess}{%
6182   \def\@gls@longaccesspl{#1}%
6183 }

```

@initaccesskeys

```

6184 \newcommand*{\@gls@initaccesskeys}{%
6185   \def\@gls@nameaccess{%
6186     \def\@gls@textaccess{%
6187       \def\@gls@pluralaccess{%
6188         \def\@gls@firstaccess{%
6189           \def\@gls@firstpluralaccess{%
6190             \def\@gls@shortaccess{%
6191               \def\@gls@shortaccesspl{%
6192                 \def\@gls@longaccess{%
6193                   \def\@gls@longaccesspl{%
6194                     }

```

ssattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```

6195 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
6196   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
6197   {#2}%
6198   {%
6199     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
6200     {#3}%
6201     {%
6202       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
6203       {#2}%
6204       {#3}%
6205     }%
6206   }%
6207 }

```

As from glossaries v4.45, the replacement text support has been corrected so that the accessibility support for abbreviations use the “E” (expanded value) element. This should actually

contain the long form since it's supposed to explain the abbreviation. This is a bit redundant on first use for styles like long-short.

aultshortaccess

```
\glsdefaultshortaccess{<long>}{<short>}
```

This command was only introduced to glossaries-accsupp 1.42 so it may not be defined.

```
6208 \def\glsdefaultshortaccess#1#2{#1 (#2)}
```

signactualsetup

```
6209 \newcommand{\glstrassignactualsetup}{%
6210 \let\@empty
6211 \let\emph\@firstofone
6212 \let\textbf\@firstofone
6213 \let\textmd\@firstofone
6214 \let\textit\@firstofone
6215 \let\textsl\@firstofone
6216 \let\textsc\@firstofone
6217 \let\textrm\@firstofone
6218 \let\textsf\@firstofone
6219 \let\texttt\@firstofone
6220 }
```

s@assign@actual

```
6221 \ifdef\pdfstringdef
6222 {
6223 \newcommand{\@gls@assign@actual}{%
6224 \begingroup
6225 \glstrassignactualsetup
6226 \pdfstringdef\@gls@actualshort{\glstrorgshort}%
6227 \pdfstringdef\@gls@actuaallong{\glstrorglong}%
6228 \pdfstringdef\@gls@actualshortpl{\@gls@shortpl}%
6229 \pdfstringdef\@gls@actuaallongpl{\@gls@longpl}%
6230 \protected@edef\@gls@tmp{\endgroup
6231 \def\noexpand\@gls@actualshort{\expandonce\@gls@actualshort}%
6232 \def\noexpand\@gls@actuaallong{\expandonce\@gls@actuaallong}%
6233 \def\noexpand\@gls@actualshortpl{\expandonce\@gls@actualshortpl}%
6234 \def\noexpand\@gls@actuaallongpl{\expandonce\@gls@actuaallongpl}%
6235 }%
6236 \@gls@tmp
6237 }
6238 }
6239 {
6240 \newcommand{\@gls@assign@actual}{%
6241 \begingroup
6242 \glstrassignactualsetup
```

```

6243     \protected@edef\@gls@tmp{\endgroup
6244     \def\noexpand\@gls@actualshort{\glstrorgshort}%
6245     \def\noexpand\@gls@actuallong{\glstrorglong}%
6246     \def\noexpand\@gls@actualshortpl{\@gls@shortpl}%
6247     \def\noexpand\@gls@actuallongpl{\@gls@longpl}%
6248     }%
6249     \@gls@tmp
6250   }
6251 }

```

lt@short@access Renamed \@gls@setup@default@access and removed argument since it can be obtained from \glstrorgshort.

@default@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation. The shortaccess value should explain the abbreviation.

```

6252 \newcommand{\@gls@setup@default@access}{%
6253   \@gls@assign@actual
6254   \ifdefempty\@gls@shortaccess
6255   {%

```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```

6256   \@gls@ifaccessattribute@set{insertdots}%
6257   {%
6258     \expandafter\@glstr@insertdots\expandafter\@gls@actualshort\expandafter
6259     {\@gls@actualshort}%
6260   }%
6261   {%
6262     \ifdefempty\@gls@longaccess
6263     {%
6264       \edef\@gls@shortaccess{\glsdefaultshortaccess
6265         {\expandonce\@gls@actuallong}{\expandonce\@gls@actualshort}}%
6266     }%
6267     {%
6268       \edef\@gls@shortaccess{\glsdefaultshortaccess
6269         {\expandonce\@gls@longaccess}{\expandonce\@gls@actualshort}}%
6270     }%
6271     \eappto\ExtraCustomAbbreviationFields{shortaccess={\@gls@shortaccess},}%

```

If shortaccessplural hasn't been set, assign plural form.

```

6272   \ifdefempty\@gls@shortaccesspl
6273   {%
6274     \@gls@ifaccessattribute@set{aposplural}%
6275     {%
6276       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
6277         \@gls@actualshort'\glstrabbrvpluralsuffix}%
6278     }%
6279     {%
6280       \@gls@ifaccessattribute@set{noshortplural}%
6281     }%
6282     \let\@gls@shortaccesspl\@gls@shortaccess

```

```

6283         }%
6284         {%
6285             \let\@gls@shortaccesspl\@gls@actualshortpl
6286         }%
6287     }%
6288     \ifdefempty\@gls@longaccesspl
6289     {%
6290         \edef\@gls@shortaccesspl{\glsdefaultshortaccess
6291             {\expandonce\@gls@actuallongpl}{\expandonce\@gls@actualshortpl}}%
6292     }%
6293     {%
6294         \edef\@gls@shortaccesspl{\glsdefaultshortaccess
6295             {\expandonce\@gls@longaccesspl}{\expandonce\@gls@actualshort}}%
6296     }%
6297     \eappto\ExtraCustomAbbreviationFields{shortpluralaccess={\@gls@shortaccesspl},}%
6298     }%
6299     {}%
6300 }%
6301 {%
6302     \ifdefempty\@gls@shortaccesspl
6303     {\let\@gls@shortaccesspl\@gls@shortaccess}%
6304     {}%
6305 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

6306     \ifdefempty\@gls@nameaccess
6307     {%
6308         \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
6309     {%
6310         \eappto\ExtraCustomAbbreviationFields{access={\@gls@shortaccess},}%
6311     }%
6312     {}%
6313 }%
6314 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

6315     \ifdefempty\@gls@textaccess
6316     {%
6317         \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
6318     {%
6319         \eappto\ExtraCustomAbbreviationFields{textaccess={\@gls@shortaccess},}%
6320     }%
6321     {}%
6322 }%
6323 {}%
6324     \ifdefempty\@gls@pluralaccess
6325     {%
6326         \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
6327     {%
6328         \eappto\ExtraCustomAbbreviationFields{%

```

```

6329         pluralaccess={\@gls@shortaccesspl},%
6330     }%
6331 }%
6332 {}%
6333 }%
6334 {}%

    If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.
6335     \ifdefempty\@gls@firstaccess
6336     {%
6337         \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
6338         {%
6339             \eappto\ExtraCustomAbbreviationFields{firstaccess={\@gls@shortaccess},}%
6340             }%
6341             {}%
6342         }%
6343         {}%
6344     \ifdefempty\@gls@firstpluralaccess
6345     {%
6346         \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
6347         {%
6348             \eappto\ExtraCustomAbbreviationFields{%
6349                 firstpluralaccess={\@gls@shortaccesspl},%
6350             }%
6351             }%
6352             {}%
6353         }%
6354         {}%
6355     }

```

Provide hooks for `\setabbreviationstyle` that automatically set the attributes appropriate for the style. If the name is just the short form and the description contains the long form, then it may not be necessary to set `nameshortaccess` but it would depend on the glossary style.

Need to provide `\glxtr<category><field>accsupp` if not already defined.

provideaccsuppcmd

```

6356 \newcommand*{\glxtrprovideaccsuppcmd}[2]{%
6357     \ifcsundef{glxtr#1#2accsupp}%
6358     {\csdef{glxtr#1#2accsupp}{\glsshortaccsupp}}%
6359     {}%
6360 }

```

`rrSetNoLongAttrs` For styles where the name, first and text are just the abbreviation.

```

6361 \newcommand*{\glxtrAccSuppAbbrSetNoLongAttrs}[1]{%
6362     \glssetcategoryattribute{#1}{nameshortaccess}{true}%
6363     \glssetcategoryattribute{#1}{firstshortaccess}{true}%
6364     \glssetcategoryattribute{#1}{textshortaccess}{true}%
6365     \glxtrprovideaccsuppcmd{#1}{name}%
6366     \glxtrprovideaccsuppcmd{#1}{first}%

```

```

6367 \glxtrprovideaccsuppcmd{#1}{firstpl}%
6368 \glxtrprovideaccsuppcmd{#1}{text}%
6369 \glxtrprovideaccsuppcmd{#1}{plural}%
6370 }

```

tFirstLongAttrs For styles where the name and text are just the abbreviation. The first form may just be long or may be short and long.

```

6371 \newcommand*{\glxtrAccSuppAbbrSetFirstLongAttrs}[1]{%
6372 \glsssetcategoryattribute{#1}{nameshortaccess}{true}%
6373 \glsssetcategoryattribute{#1}{textshortaccess}{true}%
6374 \glxtrprovideaccsuppcmd{#1}{name}%
6375 \glxtrprovideaccsuppcmd{#1}{text}%
6376 \glxtrprovideaccsuppcmd{#1}{plural}%
6377 }

```

tTextShortAttrs For styles where only the text is just the abbreviation. The name and first form may just be long or may be short and long. The name may also be short but followed by the long form in the description.

```

6378 \newcommand*{\glxtrAccSuppAbbrSetTextShortAttrs}[1]{%
6379 \glsssetcategoryattribute{#1}{textshortaccess}{true}%
6380 \glxtrprovideaccsuppcmd{#1}{text}%
6381 \glxtrprovideaccsuppcmd{#1}{plural}%
6382 }

```

tNameShortAttrs For styles where only the name is just the abbreviation. The first and subsequent form may just be long or may be short and long.

```

6383 \newcommand*{\glxtrAccSuppAbbrSetNameShortAttrs}[1]{%
6384 \glsssetcategoryattribute{#1}{nameshortaccess}{true}%
6385 \glxtrprovideaccsuppcmd{#1}{name}%
6386 }

```

etNameLongAttrs For styles where the first and text are just the abbreviation. The name may just be long or may be short and long or the name may be short.

```

6387 \newcommand*{\glxtrAccSuppAbbrSetNameLongAttrs}[1]{%
6388 \glsssetcategoryattribute{#1}{firstshortaccess}{true}%
6389 \glsssetcategoryattribute{#1}{textshortaccess}{true}%
6390 \glxtrprovideaccsuppcmd{#1}{first}%
6391 \glxtrprovideaccsuppcmd{#1}{firstpl}%
6392 \glxtrprovideaccsuppcmd{#1}{text}%
6393 \glxtrprovideaccsuppcmd{#1}{plural}%
6394 }

```

End of if accsupp part

```

6395 }
6396 {

```

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

`\glsaccessname` Display the name value (no link and no check for existence).
6397 `\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.
6398 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.
6399 `\newcommand*{\GLSaccessname}[1]{%`
6400 `\protect\mfirstucMakeUppercase{\glsentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).
6401 `\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.
6402 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
6403 `\newcommand*{\GLSaccesstext}[1]{%`
6404 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
6405 `\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.
6406 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
6407 `\newcommand*{\GLSaccessplural}[1]{%`
6408 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
6409 `\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.
6410 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
6411 `\newcommand*{\GLSaccessfirst}[1]{%`
6412 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
6413 `\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.

```
6414 \newcommand*{\GLsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}
```

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

```
6415 \newcommand*{\GLSaccessfirstplural}[1]{%
6416 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}
```

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

```
6417 \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.

```
6418 \newcommand*{\GLsaccesssymbol}[1]{\Glsentrysymbol{#1}}
```

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

```
6419 \newcommand*{\GLSaccesssymbol}[1]{%
6420 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}
```

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

```
6421 \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.

```
6422 \newcommand*{\GLsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}
```

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

```
6423 \newcommand*{\GLSaccesssymbolplural}[1]{%
6424 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}
```

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

```
6425 \newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}
```

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

```
6426 \newcommand*{\GLsaccessdesc}[1]{\Glsentrydesc{#1}}
```

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

```
6427 \newcommand*{\GLSaccessdesc}[1]{%
6428 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}
```

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

```
6429 \newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}
```

ccessdesclplural Display the desclplural value (no link and no check for existence) with the first letter converted to upper case.
6430 \newcommand*{\GLsaccessdesclplural}[1]{\Glsentrydesclplural{#1}}

ccessdesclplural Display the desclplural value (no link and no check for existence). converted to upper case.
6431 \newcommand*{\GLSaccessdesclplural}[1]{%
6432 \protect\mfirstucMakeUppercase{\glsentrydesclplural{#1}}}

\glsaccessshort Display the short form (no link and no check for existence).
6433 \newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}

\GLsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).
6434 \newcommand*{\GLsaccessshort}[1]{\Glsentryshort{#1}}

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.
6435 \newcommand*{\GLSaccessshort}[1]{%
6436 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}

lsaccessshortpl Display the short plural form (no link and no check for existence).
6437 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).
6438 \newcommand*{\GLsaccessshortpl}[1]{\Glsentryshortpl{#1}}

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.
6439 \newcommand*{\GLSaccessshortpl}[1]{%
6440 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}

\glsaccesslong Display the long form (no link and no check for existence).
6441 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}

\GLsaccesslong Display the long form (no link and no check for existence).
6442 \newcommand*{\GLsaccesslong}[1]{\Glsentrylong{#1}}

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.
6443 \newcommand*{\GLSaccesslong}[1]{%
6444 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}

glsaccesslongpl Display the long plural form (no link and no check for existence).
6445 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}

GLsaccesslongpl Display the long plural form (no link and no check for existence).
6446 \newcommand*{\GLsaccesslongpl}[1]{\Glsentrylongpl{#1}}

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

```
6447 \newcommand*{\GLSaccesslongpl}[1]{%
6448 \protect\mfirstucMakeUppercase{\glstrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
6449 \newcommand*{\@gls@initaccesskeys}{}
```

@default@access This does nothing if there's no accessibility support.

```
6450 \newcommand{\@gls@setup@default@access}{}
```

rSetNoLongAttrs This does nothing if there's no accessibility support.

```
6451 \newcommand*{\glxtrAccSuppAbbrSetNoLongAttrs}[1]{}
```

tFirstLongAttrs This does nothing if there's no accessibility support.

```
6452 \newcommand*{\glxtrAccSuppAbbrSetFirstLongAttrs}[1]{}
```

tTextShortAttrs This does nothing if there's no accessibility support.

```
6453 \newcommand*{\glxtrAccSuppAbbrSetTextShortAttrs}[1]{}
```

tNameShortAttrs This does nothing if there's no accessibility support.

```
6454 \newcommand*{\glxtrAccSuppAbbrSetNameShortAttrs}[1]{}
```

etNameLongAttrs This does nothing if there's no accessibility support.

```
6455 \newcommand*{\glxtrAccSuppAbbrSetNameLongAttrs}[1]{}
```

End of else part

```
6456 }
```

1.6 Categories

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

```
6457 \glsaddstoragekey{category}{general}{\glscategory}
```

\glusifcategory Convenient shortcut to determine if an entry has the given category.

```
6458 \newcommand{\glusifcategory}[4]{%
6459 \ifglstfield{#1}{category}{#2}{#3}{#4}%
6460 }
```

Categories can have attributes.

tegrityattribute

```
\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

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

```
6461 \newcommand*{\glissetcategoryattribute}[3]{%
6462 \csdef{@glxtr@categoryattr@#1@#2}{#3}%
6463 }
```

tegoryattribute

```
\glsggetcategoryattribute{<category>}{<attribute-label>}
```

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

```
6464 \newcommand*{\glsggetcategoryattribute}[2]{%
6465   \csuse{@glxtr@categoryattr@#1@#2}%
6466 }
```

tegoryattribute

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
6467 \newcommand*{\glshascategoryattribute}[4]{%
6468   \ifcsvoid{@glxtr@categoryattr@#1@#2}{#4}{#3}%
6469 }
```

glissetattribute

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

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

```
6470 \newcommand*{\glissetattribute}[3]{%
6471   \glsgsetcategoryattribute{\glscategory{#1}}{#2}{#3}%
6472 }
```

glsggetattribute

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

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

```
6473 \newcommand*{\glsggetattribute}[2]{%
6474   \glsggetcategoryattribute{\glscategory{#1}}{#2}%
6475 }
```

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.

```

6476 \newcommand*{\glshasattribute}[4]{%
6477   \ifglstryexists{#1}%
6478   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
6479   {#4}%
6480 }

```

categoryattribute

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

True if category has the attribute with the given value.

```

6481 \newcommand{\glshcategoryattribute}[5]{%
6482   \ifcsundef{@glstr@categoryattr@#1@#2}%
6483   {#5}%
6484   {\ifcsstring{@glstr@categoryattr@#1@#2}{#3}{#4}{#5}}%
6485 }

```

\glshattribute

`\glshattribute{<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.

```

6486 \newcommand{\glshattribute}[5]{%
6487   \ifglstryexists{#1}%
6488   {\glshcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
6489   {#5}%
6490 }

```

Set attributes for the default general category:

```
6491 \glissetcategoryattribute{general}{regular}{true}
```

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

```
6492 \glissetcategoryattribute{acronym}{regular}{true}
```

regularcategory Convenient shortcut to add the regular attribute.

```

6493 \newcommand*{\glissetregularcategory}[1]{%
6494   \glissetcategoryattribute{#1}{regular}{true}%
6495 }

```

regularcategory

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

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

```

6496 \newcommand{\glsifregularcategory}[3]{%
6497   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
6498 }

```

regularcategory

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

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

```

6499 \newcommand{\glsifnotregularcategory}[3]{%
6500   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
6501 }

```

\glsifregular

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

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

```

6502 \newcommand{\glsifregular}[3]{%
6503   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
6504 }

```

glsifnotregular

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

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

```

6505 \newcommand{\glsifnotregular}[3]{%
6506   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
6507 }

```

reachincategory

```
\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.

```

6508 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
6509   \forallglossaries[#1]{#3}%

```

```

6510  {%
6511      \forlsglentries[#3]{#4}%
6512      {%
6513          \glsifcategory{#4}{#2}{#5}{}%
6514      }%
6515  }%
6516 }

```

chwithattribute

```

\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.

```

6517 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
6518     \forallglossaries[#1]{#4}%
6519     {%
6520         \forlsglentries[#4]{#5}%
6521         {%
6522             \glsifattribute{#5}{#2}{#3}{#6}{}%
6523         }%
6524     }%
6525 }

```

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

```

6526 \ifdef\newterm
6527 {%

```

`\newterm`

```

6528     \renewcommand*{\newterm}[2][ ]{%
6529         \newglossaryentry{#2}%
6530         {type={index},category=index,name={#2},%
6531          description={\glstrpostdescription\nopostdesc},#1}%
6532     }

```

Indexed terms are regular by default.

```

6533     \glssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

6534     \newcommand*{\glstrpostdescindex}{}

6535 }
6536 {}

```

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.

```
6537 \ifdef\printsymbols
6538 {%
```

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

```
6539 \newcommand*{\glsxtrnewsymbol}[3] [] {%
6540 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
6541 }
```

Symbols are regular by default.

```
6542 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
6543 \newcommand*{\glsxtrpostdescsymbol}{}

6544 }
6545 {}
```

Similar for the numbers option.

```
6546 \ifdef\printnumbers
6547 {%
```

`glsxtrnewnumber`

```
6548 \ifdef\printnumbers
6549 \newcommand*{\glsxtrnewnumber}[3] [] {%
6550 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
6551 }
```

Numbers are regular by default.

```
6552 \glsssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
6553 \newcommand*{\glsxtrpostdescnumber}{}

6554 }
6555 {}
```

`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.

```
6556 \newcommand*{\glsxtrsetcategory}[2] {%
6557 \@for\@glsxtr@label:=#1\do
6558 {%
6559 \glsfieldxddef{\@glsxtr@label}{category}{#2}%
6560 }%
6561 }
```

`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.

```

6562 \newcommand*{\glstrsetcategoryforall}[2]{%
6563   \forallglossaries[#1]{\@glstr@type}{%
6564     \forallsentries[\@glstr@type]{\@glstr@label}%
6565     {%
6566       \glsexdef{\@glstr@label}{category}{#2}%
6567     }%
6568   }%
6569 }

```

`rfieldtitlecase`

`\glstrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```

6570 \newcommand*{\glstrfieldtitlecase}[2]{%
6571   \expandafter\glstrfieldtitlecasesecs\expandafter
6572   {\csname glo@\glsetoklabel{#1}@#2\endcsname}%
6573 }

```

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

```

6574 \newcommand*{\glstrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}

```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

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

```

6575 \@ifpackageloaded{glossaries-accsupp}
6576 {
6577   \renewcommand*{\glossentrydesc}[1]{%
6578     \glsoifexistsorwarn{#1}%
6579     {%
6580       \glsssetabbrvfmt{\glscategory{#1}}%

```

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

```

6581     \glshasattribute{#1}{glossdescfont}%
6582     {%
6583       \edef\@glstr@attrval{\glsggetattribute{#1}{glossdescfont}}%
6584       \ifcsdef{\@glstr@attrval}%
6585       {%
6586         \letcs{\@glstr@glossdescfont}{\@glstr@attrval}%
6587       }%
6588       {%
6589         \GlossariesExtraWarning{Unknown control sequence name

```

```

6590         '\@glxtr@attrval' supplied in glossdescfont attribute
6591         for entry '#1'. Ignoring}%
6592         \let\@glxtr@glossdescfont\@firstofone
6593     }%
6594 }%
6595 {\let\@glxtr@glossdescfont\@firstofone}%
6596 \glsifattribute{#1}{glossdesc}{firstuc}%
6597 {%
6598     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6599 }%
6600 {%
6601     \glsifattribute{#1}{glossdesc}{title}%
6602     {%
6603         \@glxtr@do@titlecaps@warn
6604         \glsdescriptionaccessdisplay
6605         {%
6606             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6607         }%
6608         {#1}%
6609     }%
6610     {%
6611         \@glxtr@glossdescfont{\glsaccessdesc{#1}}%
6612     }%
6613 }%
6614 }%
6615 }
6616 }
6617 {
6618 \renewcommand*{\glossentrydesc}[1]{%
6619     \glsdoifexistsorwarn{#1}%
6620     {%
6621         \glssetabbrvfmt{\glscategory{#1}}%
6622         \glsattribute{#1}{glossdescfont}%
6623         {%
6624             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6625             \ifcsdef{\@glxtr@attrval}%
6626             {%
6627                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6628             }%
6629             {%
6630                 \GlossariesExtraWarning{Unknown control sequence name
6631                 '\@glxtr@attrval' supplied in glossdescfont attribute
6632                 for entry '#1'. Ignoring}%
6633                 \let\@glxtr@glossdescfont\@firstofone
6634             }%
6635         }%
6636         {\let\@glxtr@glossdescfont\@firstofone}%
6637         \glsifattribute{#1}{glossdesc}{firstuc}%
6638         {%

```



```

6639      \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
6640    }%
6641    {%
6642      \glsifattribute{#1}{glossdesc}{title}%
6643      {%
6644        \@glsxtr@do@titlecaps@warn
6645        \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
6646      }%
6647      {%
6648        \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
6649      }%
6650    }%
6651  }%
6652 }
6653 }

```

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

```

6654 \ifpackageloaded{glossaries-accsupp}
6655 {
6656   \renewcommand*{\glossentryname}[1]{%
6657     \@glsdoifexistsorwarn{#1}%
6658     {%
6659       \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

6660   \glshasattribute{#1}{glossnamefont}%
6661   {%
6662     \edef\@glsxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6663     \ifcsdef{\@glsxtr@attrval}%
6664     {%
6665       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6666     }%
6667     {%
6668       \GlossariesExtraWarning{Unknown control sequence name
6669         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
6670         for entry ‘#1’. Reverting to default \string\glsgnamefont}%
6671       \let\@glsxtr@glossnamefont\glsgnamefont
6672     }%
6673   }%
6674   {\let\@glsxtr@glossnamefont\glsgnamefont}%
6675   \glsifattribute{#1}{glossname}{firstuc}%
6676   {%
6677     \glsgnameaccessdisplay
6678     {%
6679       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6680     }%
6681     {#1}%
6682   }%
6683   {%

```

```

6684 \glsifattribute{#1}{glossname}{title}%
6685 {%
6686 \@glstr@do@titlecaps@warn
6687 \glsnameaccessdisplay
6688 {%
6689 \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
6690 }%
6691 {#1}%
6692 }%
6693 {%
6694 \glsifattribute{#1}{glossname}{uc}%
6695 {%
6696 \glsnameaccessdisplay
6697 {%

```

Hide the label from the upper-casing command.

```

6698 \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6699 \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6700 }%
6701 {#1}%
6702 }%
6703 {%
6704 \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
6705 \glsnameaccessdisplay
6706 {%
6707 \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6708 }%
6709 {#1}%
6710 }%
6711 }%
6712 }%

```

Do post-name hook:

```

6713 \glstrpostnamehook{#1}%
6714 }%
6715 }
6716 }
6717 {
6718 \renewcommand*{\glossentryname}[1]{%
6719 \@glsdofexistsorwarn{#1}%
6720 {%
6721 \glsetabbrvfmt{\glscategory{#1}}%
6722 \glshasattribute{#1}{glossnamefont}%
6723 {%
6724 \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
6725 \ifcsdef{\@glstr@attrval}%
6726 {%
6727 \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6728 }%
6729 {%

```

```

6730      \GlossariesExtraWarning{Unknown control sequence name
6731      ‘\@glxtr@attrval’ supplied in glossnamefont attribute
6732      for entry ‘#1’. Reverting to default \string\glnamefont}%
6733      \let\@glxtr@glossnamefont\glnamefont
6734      }%
6735      }%
6736      {\let\@glxtr@glossnamefont\glnamefont}%
6737      \gl@ifattribute{#1}{glossname}{firstuc}%
6738      {%
6739      \@glxtr@glossnamefont{\Glsentryname{#1}}%
6740      }%
6741      {%
6742      \gl@ifattribute{#1}{glossname}{title}%
6743      {%
6744      \@glxtr@do@titlecaps@warn
6745      \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
6746      }%
6747      {%
6748      \gl@ifattribute{#1}{glossname}{uc}%
6749      {%

```

Hide the label from the upper-casing command.

```

6750      \letcs{\glo@name}{glo@glsetoklabel{#1}@name}%
6751      \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6752      }%
6753      {%

```

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

```

6754      \letcs{\glo@name}{glo@glsetoklabel{#1}@name}%
6755      \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
6756      }%
6757      }%
6758      }%

```

Do post-name hook.

```

6759      \glxtrpostnamehook{#1}%
6760      }%
6761      }
6762      }

```

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

```

6763 \ifpackageloaded{glossaries-accsupp}
6764 {
6765   \renewcommand*{\Glossentryname}[1]{%
6766     \@glsetoklabel{#1}%
6767     {%
6768       \glsetabbrvfmt{\glscategory{#1}}%

```

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

```

6769     \glshasattribute{#1}{glossnamefont}%

```

```

6770    {%
6771        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6772        \ifcsdef{\@glsxtr@attrval}%
6773        {%
6774            \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6775        }%
6776        {%
6777            \GlossariesExtraWarning{Unknown control sequence name
6778                '\@glsxtr@attrval' supplied in glossnamefont attribute
6779                for entry '#1'. Reverting to default \string\glsnamefont}%
6780            \let\@glsxtr@glossnamefont\glsnamefont
6781        }%
6782    }%
6783    {\let\@glsxtr@glossnamefont\glsnamefont}%
6784    \glsnameaccessdisplay
6785    {%
6786        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6787    }%
6788    {#1}%

```

Do post-name hook:

```

6789        \glsxtrpostnamehook{#1}%
6790    }%
6791 }
6792 }
6793 {
6794     \renewcommand*{\Glossentryname}[1]{%
6795         \@glsdoifexistsorwarn{#1}%
6796         {%
6797             \glssetabbrvfmt{\glscategory{#1}}%
6798             \glschasattribute{#1}{glossnamefont}%
6799             {%
6800                 \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6801                 \ifcsdef{\@glsxtr@attrval}%
6802                 {%
6803                     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6804                 }%
6805                 {%
6806                     \GlossariesExtraWarning{Unknown control sequence name
6807                         '\@glsxtr@attrval' supplied in glossnamefont attribute
6808                         for entry '#1'. Reverting to default \string\glsnamefont}%
6809                     \let\@glsxtr@glossnamefont\glsnamefont
6810                 }%
6811             }%
6812             {\let\@glsxtr@glossnamefont\glsnamefont}%
6813             \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6814        \glsxtrpostnamehook{#1}%
6815    }%

```

```
6816 }
6817 }
```

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.

```
6818 \newcommand*{\glstrpostnamehook}[1]{%
6819   \let\@glsnumberformat\@glstr@defaultnumberformat
6820   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
6821   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
6822   \csuse{glstrpostname\glscategory{#1}}%
6823 }
```

`\glsextrapostnamehook`

```
6824 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
6825 \newcommand*{\glsdefpostname}[2]{%
6826   \csdef{glstrpostname#1}{#2}%
6827 }
```

`\setaccessdisplay`

```
6828 \@ifpackageloaded{glossaries-accsupp}
6829 {
6830   \newcommand*{\glstr@setaccessdisplay}[1]{%
6831     \ifcsdef{gls#1accessdisplay}%
6832       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
6833     {%
```

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.

```
6834     \edef\@gls@thisval{#1}%
6835     \@for\@gls@map:=\@gls@keymap\do{%
6836       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6837       \ifdefequal{\@this@key}{\@gls@thisval}%
6838       {%
6839         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6840         \@endfortrue
6841       }%
6842     }%
6843   }%
6844   \ifcsdef{gls\@gls@thisval accessdisplay}%
```

```

6845      {\letcs\@glxtr@accessdisplay{gls\@gls@thisval accessdisplay}}}%
6846      {\let\@glxtr@accessdisplay\@firstoftwo}%
6847    }%
6848  }
6849 }
6850 {%
6851 \newcommand*{\@glxtr@setaccessdisplay}[1]{%
6852   \let\@glxtr@accessdisplay\@firstoftwo}
6853 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6854 \newrobustcmd*{\glossentrynameother}[2]{%
6855   \@glsdoifexistsorwarn{#1}%
6856   {%

```

Accessibility support:

```

6857   \@glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6858   \@glsssetabbrfmt{\@glscategory{#1}}}%
6859   \@glshasattribute{#1}{glossnamefont}%
6860   {%
6861     \edef\@glxtr@attrval{\@glsggetattribute{#1}{glossnamefont}}}%
6862     \ifcsdef{\@glxtr@attrval}%
6863     {%
6864       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6865     }%
6866     {%
6867       \GlossariesExtraWarning{Unknown control sequence name
6868         '\@glxtr@attrval' supplied in glossnamefont attribute
6869         for entry '#1'. Reverting to default \string\glsnamefont}%
6870       \let\@glxtr@glossnamefont\glsnamefont
6871     }%
6872   }%
6873   {\let\@glxtr@glossnamefont\glsnamefont}%
6874   \@glusifattribute{#1}{glossname}{firstuc}%
6875   {%
6876     \@glxtr@accessdisplay
6877     {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6878     {#1}%
6879   }%
6880   {%
6881     \@glusifattribute{#1}{glossname}{title}%
6882     {%
6883       \@glxtr@do@titlecaps@warn
6884       \@glxtr@accessdisplay
6885       {\@glxtr@glossnamefont{\@glxtr@field@titlecase{#1}{#2}}}%
6886       {#1}%
6887     }%

```

```

6888      {%
6889      \glsifattribute{#1}{glossname}{uc}%
6890      {%
6891      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6892      \@glsxtr@accessdisplay
6893      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6894      {#1}%
6895      }%
6896      {%
6897      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6898      \@glsxtr@accessdisplay
6899      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6900      {#1}%
6901      }%
6902      }%
6903      }%

```

Do post-name hook.

```

6904      \glsxtrpostnamehook{#1}%
6905      }%
6906  }

```

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

```

6907 \newif\ifglsxtr@format@override
6908 \@glsxtr@format@overridefalse

```

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

xFormatOverride

```

6909 \@ifpackageloaded{hyperref}
6910 {

```

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

```

6911   \ifHy@hyperindex
6912     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6913       \@glsxtr@format@override true
6914       \appto\theindex{\let\glshypernumber\@firstofone}%
6915     }
6916   \else
6917     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6918       \@glsxtr@format@override true
6919       \appto\theindex{\let\glshypernumber\hyperpage}%
6920     }
6921   \fi
6922 }
6923 {
6924   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6925     \@glsxtr@format@override true

```

```

6926 }
6927 }
6928 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6929 \newcommand*{\glxtrdoautoindexname}[2]{%
6930   \glshasattribute{#1}{#2}%
6931   {%

```

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.

```

6932   \@glxtr@autoindex@setname{#1}%

```

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

```

6933   \protected@edef\@glxtr@attrval{\glsggetattribute{#1}{#2}}%
6934   \if@glxtr@format@override

6935     \ifx\@glsnnumberformat\@glxtr@defaultnumberformat
6936     \else
6937       \let\@glxtr@attrval\@glsnnumberformat
6938     \fi
6939   \fi
6940   \ifdefstring{\@glxtr@attrval}{true}%
6941   {}%
6942   {\eappto\@glo@name{\@glxtr@autoindex@encap\@glxtr@attrval}}%
6943   \expandafter\glxtrautoindex\expandafter{\@glo@name}%
6944   }%
6945   {}%
6946 }

```

glxtrautoindex

```

6947 \newcommand*{\glxtrautoindex}{\index}

```

xtrautoindexesc

```

6948 \newcommand{\glxtrautoindexesc}{%
6949   \@gls@checkmkidxchars\@glo@sort
6950   \@glxtr@autoindex@doextra@esc\@glo@sort
6951 }

```

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

```

6952 \newcommand*{\@glxtr@autoindex@setname}[1]{%
6953   \protected@edef\@glo@name{\glxtrautoindexentry{#1}}%
6954   \glxtrautoindexassignsort{\@glo@sort}{#1}%
6955   \glxtrautoindexesc
6956   \epreto\@glo@name{\@glo@sort\@glxtr@autoindex@at}%
6957 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6958 \newcommand*{\glxtrautoindexentry}[1]{\string\glsentryname{#1}}

```


indexassignsort Used to assign the sort value when auto-indexing.

```
6959 \newcommand*{\glxtrautoindexassignsort}[2]{%
6960   \glslentryfield{#1}{#2}{sort}%
6961 }
```

dex@doextra@esc

```
6962 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6963   \ifx\@glxtr@autoindex@esc\@gl@quotechar
6964   \else
6965     \def\@gl@checkedmkidx{}%
6966     \edef\@glxtr@checkspch{%
6967       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6968       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6969       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6970     \@glxtr@checkspch
6971     \let#1\@gl@checkedmkidx\relax
6972   \fi
```

Escape actual character unless it has already been escaped.

```
6973   \ifx\@glxtr@autoindex@at\@gl@actualchar
6974   \else
6975     \def\@gl@checkedmkidx{}%
6976     \edef\@glxtr@checkspch{%
6977       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6978       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6979       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6980     \@glxtr@checkspch
6981     \let#1\@gl@checkedmkidx\relax
6982   \fi
```

Escape level character unless it has already been escaped.

```
6983   \ifx\@glxtr@autoindex@level\@gl@levelchar
6984   \else
6985     \def\@gl@checkedmkidx{}%
6986     \edef\@glxtr@checkspch{%
6987       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6988       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6989       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6990     \@glxtr@checkspch
6991     \let#1\@gl@checkedmkidx\relax
6992   \fi
```

Escape encap character unless it has already been escaped.

```
6993   \ifx\@glxtr@autoindex@encap\@gl@encapchar
6994   \else
6995     \def\@gl@checkedmkidx{}%
6996     \edef\@glxtr@checkspch{%
6997       \noexpand\@glxtr@autoindex@escencap\expandonce{#1}%
6998       \noexpand\@empty\@glxtr@autoindex@encap\noexpand\@nnil
```

```

6999      \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
7000      \@glsxtr@checkspch
7001      \let#1\@gls@checkedmkidx\relax
7002      \fi
7003 }

```

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`.

```
7004 \newcommand*{\@glsxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```

7005 \newcommand*{\GlsXtrSetActualChar}[1]{%
7006   \gdef\@glsxtr@autoindex@at{#1}%
7007   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
7008     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
7009   }%
7010 }
7011 \@onlypreamble\GlsXtrSetActualChar
7012 \makeatother
7013 \GlsXtrSetActualChar{@}
7014 \makeatletter

```

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

```
7015 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```

7016 \newcommand*{\GlsXtrSetEncapChar}[1]{%
7017   \gdef\@glsxtr@autoindex@encap{#1}%
7018   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
7019     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
7020   }%
7021 }
7022 \GlsXtrSetEncapChar{|}
7023 \@onlypreamble\GlsXtrSetEncapChar

```

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

```
7024 \newcommand*{\@glsxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```

7025 \newcommand*{\GlsXtrSetLevelChar}[1]{%
7026   \gdef\@glsxtr@autoindex@level{#1}%
7027   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
7028     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
7029   }%
7030 }
7031 \GlsXtrSetLevelChar{!}
7032 \@onlypreamble\GlsXtrSetLevelChar

```

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

```
7033 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
7034 \newcommand*{\GlsXtrSetEscChar}[1]{%
7035   \gdef\@glsxtr@autoindex@esc{#1}%
7036   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
7037     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
7038   }%
7039 }
7040 \GlsXtrSetEscChar{"}
7041 \@onlypreamble\GlsXtrSetEscChar
```

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

```
7042 \ifdef\actualchar
7043 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
7044 {}
```

Quote character \quotechar:

```
7045 \ifdef\quotechar
7046 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
7047 {}
```

Level character \levelchar:

```
7048 \ifdef\levelchar
7049 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
7050 {}
```

Encap character \encapchar:

```
7051 \ifdef\encapchar
7052 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
7053 {}
```

leto@endescspch

```
7054 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

oindex@esc@spch

```
\@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
7055 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
7056   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
7057   \toks@={#3}%
7058   \ifx\@nnil#3\relax
7059     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
7060   \else
7061     \ifx\@nnil#4\relax
7062       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

7063 \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
7064 #4#5\@glsxtr@endescspch}%
7065 \else
7066 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
7067 \@glsxtr@autoindex@esc#1}%
7068 \def\@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
7069 \fi
7070 \fi
7071 \@glsxtr@checkspch
7072 }

```

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

```

7073 \renewcommand*{\Glossentrydesc}[1]{%
7074 \glsdoifexistsorwarn{#1}%
7075 {%
7076 \glssetabbrvfmt{\glscategory{#1}}%
7077 \Glsaccessdesc{#1}%
7078 }%
7079 }

```

\glossentrysymbol Redefine to set the format and accessibility support. Allow for the possibility of being used in a section heading for standalone entry definitions.

```

7080 \ifdef\teorpdfstring
7081 {
7082 \renewcommand*{\glossentrysymbol}[1]{%
7083 \teorpdfstring{\@glossentrysymbol{#1}}{\glsentrypdfsymbol{#1}}%
7084 }
7085 }
7086 {
7087 \renewcommand*{\glossentrysymbol}[1]{\@glossentrysymbol{#1}}
7088 }

```

\glsentrypdfsymbol May be redefined to a field that expands to a value that's more suitable for PDF bookmarks.

```

7089 \newcommand{\glsentrypdfsymbol}[1]{\glsentrysymbol{#1}}

```

\glossentrysymbol There are no case-changing attributes as it's less usual for symbols.

```

7090 \newrobustcmd*{\@glossentrysymbol}[1]{%
7091 \glsdoifexistsorwarn{#1}%
7092 {%
7093 \begingroup
7094 \glssetabbrvfmt{\glscategory{#1}}%
7095 \glsattribute{#1}{glosssymbolfont}%
7096 {%
7097 \edef\@glsxtr@attrval{\glsattribute{#1}{glosssymbolfont}}%
7098 \ifcsdef{\@glsxtr@attrval}%
7099 {%
7100 \letcs{\@glsxtr@glosssymbolfont}{\@glsxtr@attrval}%
7101 }%
7102 }%

```

```

7103         \GlossariesExtraWarning{Unknown control sequence name
7104         ‘\@glxtr@attrval’ supplied in glosssymbolfont attribute
7105         for entry ‘#1’. Ignoring}%
7106         \let\@glxtr@glosssymbolfont\@firstofone
7107     }%
7108 }%
7109 {\let\@glxtr@glosssymbolfont\@firstofone}%
7110 \@glxtr@glosssymbolfont{\@glxtr@glosssymbolfont{#1}}%
7111 \endgroup
7112 }%
7113 }

```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

7114 \renewcommand*{\Glossentrysymbol}[1]{%
7115     \glsoifexistsorwarn{#1}%
7116     {%
7117         \glsssetabbrvfmt{\@glxtr@category{#1}}%
7118         \Glsaccesssymbol{#1}%
7119     }%
7120 }

```

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

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

```

7121 \newcommand*{\GlsXtrEnableInitialTagging}{%
7122     \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
7123 }
7124 \@onlypreamble\GlsXtrEnableInitialTagging

```

r@enabletagging Starred version undefines command.

```

7125 \newcommand*{\s@glxtr@enabletagging}[2]{%
7126     \undef#2%
7127     \@glxtr@enabletagging{#1}{#2}%
7128 }

```

r@enabletagging Internal command.

```

7129 \newcommand*{\@glxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
7130     \@for\@glxtr@cat:=#1\do
7131     {%
7132         \ifdefempty\@glxtr@cat
7133         {}%
7134         {\glsssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%
7135     }%
7136     \newrobustcmd*#2[1]{##1}%
7137     \def\@glxtr@taggingcs{#2}%

```

```

7138 \renewcommand*\@glsxtr@activate@initialtagging{%
7139   \let#2\@glsxtr@tag
7140 }%
7141 \ifundef\@gls@preglossaryhook
7142 {\GlossariesExtraWarning{Initial tagging requires at least
7143   glossaries.sty v4.19 to work correctly}}%
7144 {}%
7145 }

```

Are we using an old version of mfirstuc that has a bug in \capitalisewords? If so, patch it so we don't have a problem with a combination of tagging and title case.

\mfu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```

7146 \ifundef\mfu@checkword@do
7147 {
7148   \newcommand*\mfu@checkword@do[1]{%
7149     \ifdefstring{\mfu@checkword@arg}{#1}%
7150     {%
7151       \let\@mfu@domakefirstuc\@firstofone
7152       \listbreak
7153     }%
7154   }%
7155 }

```

\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.

```

7156 \ifundef\mfu@checkword
7157 {
7158   \newcommand{\@glsxtr@do@titlecaps@warn}{%
7159     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
7160       support not available}%

```

One warning should suffice.

```

7161     \let\@glsxtr@do@titlecaps@warn\relax
7162   }
7163 }
7164 {
7165   \renewcommand*\mfu@checkword[1]{%
7166     \def\mfu@checkword@arg{#1}%
7167     \let\@mfu@domakefirstuc\makefirstuc
7168     \forlistloop\mfu@checkword@do\@mfu@nocaplist
7169   }
7170 }
7171 }
7172 {}% no patch required

```

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

```

7173 \newcommand*\@glsxtr@do@titlecaps@warn{}

```

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

```
7174 \newcommand*\@glstr@activate@initialtagging{}
```

\@glstr@tag Definition of tagging command when used in glossary.

```
7175 \newrobustcmd*{\@glstr@tag}[1]{%
7176   \glsifattribute{\glscurrententrylabel}{tagging}{true}%
7177   {\glstrtagfont{#1}}{#1}%
7178 }
```

\glstrtagfont Used in the glossary.

```
7179 \newcommand*{\glstrtagfont}[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.

```
7180 \ifdef\@glspreglossaryhook
```

```
7181 {
```

```
7182   \renewcommand*{\@glspreglossaryhook}{%
```

```
7183     \@glstr@activate@initialtagging
```

Since the glossaries are automatically scoped, \@glstr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```
7184   \ifundef\@glstr@org@postdescription
```

```
7185   {%
```

```
7186     \let\@glstr@org@postdescription\glspostdescription
```

```
7187     \renewcommand*{\glspostdescription}{%
```

```
7188       \ifglentryexists{\glscurrententrylabel}%
```

```
7189       {%
```

```
7190         \glstrpostdescription
```

```
7191         \@glstr@org@postdescription
```

```
7192       }%
```

```
7193     {}%
```

```
7194   }%
```

```
7195 }%
```

```
7196 {}%
```

Enable the options used by \@glstrp:

```
7197   \glossxtrsetpopts
```

```
7198 }%
```

```
7199 }
```

```
7200 {}
```

postdescription This command will only be used if \@glspreglossaryhook 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.

```
7201 \newcommand*{\glstrpostdescription}{%
```

```

7202 \csuse{glxstrpostdesc\glscategory{\glscurrententrylabel}}%
7203 }

postdescgeneral
7204 \newcommand*{\glxstrpostdescgeneral}{}

xtrpostdescterm
7205 \newcommand*{\glxstrpostdescterm}{}

postdescacronym
7206 \newcommand*{\glxstrpostdescacronym}{}

descabbreviation
7207 \newcommand*{\glxstrpostdescabbreviation}{}

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.
7208 \newcommand*{\glsdefpostdesc}[2]{%
7209 \csdef{glxstrpostdesc#1}{#2}%
7210 }

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.
7211 \renewcommand*{\glspostlinkhook}{%
7212 \ifglisentryexists{\glslabel}{\glxstrpostlinkhook}{}%
7213 }

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.
7214 \newcommand*{\glxstrpostlinkhook}{%
7215 \glxstrdiscardperiod{\glslabel}%
7216 {\glxstrpostlinkendsentence}%
7217 {\glxstrifcustomdiscardperiod
7218 {\glxstrifperiod{\glxstrpostlinkendsentence}{\glxstrpostlink}}}%
7219 {\glxstrpostlink}%
7220 }%
7221 }

omdiscardperiod Allow user to provide a custom check. Should expand to #2 if no check is required otherwise
expand to #1.
7222 \newcommand*{\glxstrifcustomdiscardperiod}[2]{#2}

\glxstrpostlink
7223 \newcommand*{\glxstrpostlink}{%
7224 \csuse{glxstrpostlink\glscategory{\glslabel}}%
7225 }

```


`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glsxtrpostlink`.

```
7226 \newcommand*{\glsdefpostlink}[2]{%
```

`\ifthenelse` is used to ensure that the expanded value is tested. (The category label must be fully expandable.)

```
7227 \ifthenelse{\equal{#1}{}}{%
```

```
7228 {\PackageError{glossaries-extra}
```

```
7229 {Invalid empty category label in \string\glsdefpostlink}{}}%
```

```
7230 {\csdef{glsxtrpostlink#1}{#2}}%
```

```
7231 }
```

`linkendsentence` Done by `\glsxtrpostlinkhook` if a full stop is discarded.

```
7232 \newcommand*{\glsxtrpostlinkendsentence}{%
```

```
7233 \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}{%
```

```
7234 {%
```

```
7235 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
7236 .\spacefactor\sfcode'\. \relax
```

```
7237 }%
```

```
7238 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
7239 \spacefactor\sfcode'\. \relax
```

```
7240 }%
```

```
7241 }
```

`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.

```
7242 \newcommand*{\glsxtrpostlinkAddDescOnFirstUse}{%
```

```
7243 \glsxtrifwasfirstuse{\space\glsxtrparen{\glsaccessdesc{\glslabel}}}{}}%
```

```
7244 }
```

`ymbolOnFirstUse` 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.

```
7245 \newcommand*{\glsxtrpostlinkAddSymbolOnFirstUse}{%
```

```
7246 \glsxtrifwasfirstuse
```

```
7247 {%
```

```
7248 \ifglshassymbol{\glslabel}}%
```

```
7249 {\space\glsxtrparen{\glsaccesssymbol{\glslabel}}}{}}%
```

```
7250 {}%
```

```
7251 }%
```

```
7252 {}%
```

```
7253 }
```

`lDescOnFirstUse` Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```

7254 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
7255   \glxtrifwasfirstuse
7256   {%
7257     \space\glxtrparen
7258     {%
7259       \ifglshassymbol{\glslabel}%
7260       {\glssaccesssymbol{\glslabel}, }%
7261       {}%
7262       \glssaccessdesc{\glslabel}%
7263     }%
7264   }%
7265   {}%
7266 }

```

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).

```

7267 \newcommand*{\glxtrdiscardperiod}[3]{%
7268   \glxtrifwasfirstuse
7269   {%
7270     \glusifattribute{#1}{retainfirstuseperiod}{true}%
7271     {#3}%
7272     {%
7273       \glusifattribute{#1}{discardperiod}{true}%
7274       {%
7275         \glusifplural
7276         {%
7277           \glusifattribute{#1}{pluraldiscardperiod}{true}%
7278           {\glxtrifperiod{#2}{#3}}%
7279           {#3}%
7280         }%
7281         {%
7282           \glxtrifperiod{#2}{#3}%
7283         }%
7284       }%
7285       {#3}%
7286     }%
7287   }%
7288   {%
7289     \glusifattribute{#1}{discardperiod}{true}%
7290     {%
7291       \glusifplural
7292       {%
7293         \glusifattribute{#1}{pluraldiscardperiod}{true}%
7294         {\glxtrifperiod{#2}{#3}}%
7295         {#3}%
7296       }%
7297     }%

```

```

7298      \glxtrifperiod{#2}{#3}%
7299      }%
7300      }%
7301      {#3}%
7302      }%
7303      }

```

`\glxtrifperiod` 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`

```

7304 \newcommand*\glxtrifperiod}[1]{\new@ifnextchar.\@firstoftwo{#1}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glxtr@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 ' ').

```

7305 \newcommand*\glxtr@punclist{.,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

7306 \newcommand*\glxtraddpunctuationmark}[1]{\appto\glxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

7307 \newcommand*\glxtrsetpunctuationmarks}[1]{\def\glxtr@punclist{#1}}

```

`\glxtrifpunc`

```

\glxtrifnextpunc{<true part>}{<false part>}

```

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```

7308 \newcommand*\glxtrifnextpunc}[2]{%
7309   \def\reserved@a{#1}%
7310   \def\reserved@b{#2}%
7311   \futurelet\@glspunc@token\glxtr@ifnextpunc
7312 }

```

`\glxtr@ifnextpunc`

```

7313 \newcommand*\glxtr@ifnextpunc{%-
7314   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{-}%
7315   \reserved@b
7316 }

```

`\glxtr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```

7317 \newcommand*\glxtr@ifpunctoken}[1]{%
7318   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
7319 }

```

xtr@ifpunctoken

```
7320 \def\@glxstr@ifpunctoken#1#2{%
7321   \let\reserved@d=#2%
7322   \ifx\reserved@d\@nnil
7323     \let\glxstr@next\@glxstr@notfoundinlist
7324   \else
7325     \ifx#1\reserved@d
7326       \let\glxstr@next\@glxstr@foundinlist
7327     \else
7328       \let\glxstr@next\@glxstr@ifpunctoken
7329     \fi
7330   \fi
7331   \glxstr@next#1%
7332 }
```

xtr@foundinlist

```
7333 \def\@glxstr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
7334 \def\@glxstr@notfoundinlist#1{\@secondoftwo}
```

lsxtrdopostpunc

`\glxtrdopostpunc{<code>}`

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```
7335 \newcommand{\glxtrdopostpunc}[1]{%
7336   \glxstrifnextpunc{\@glxstr@swaptwo{#1}}{#1}%
7337 }
```

@glxstr@swaptwo

```
7338 \newcommand{\@glxstr@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, it needs to be applied by `\newabbreviation`.

```
7339 \define@key{glxstrabbrv}{category}{%
7340   \edef\glscategorylabel{#1}%
7341 }
```

Save the short plural form. This may be needed before the entry is defined.

```
7342 \define@key{glxstrabbrv}{shortplural}{%
7343   \def\@gls@shortpl{#1}%
7344 }
```

Similarly for the long plural form.

```
7345 \define@key{glxstrabbrv}{longplural}{%
7346   \def\@gls@longpl{#1}%
7347 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

`\glsshortpltok`

```
7348 \newtoks\glsshortpltok
```

`\glslongpltok`

```
7349 \newtoks\glslongpltok
```

`glxstr@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.

```
7350 \newcommand*{\@glxstr@insertdots}[2]{%
7351   \def#1{}%
7352   \@glxstr@insert@dots#1#2\@nnil
7353 }
```

`glxstr@insert@dots`

```
7354 \newcommand*{\@glxstr@insert@dots}[2]{%
7355   \ifx\@nnil#2\relax
7356   \let\@glxstr@insert@dots@next\@gobble
7357   \else
7358   \ifx\relax#2\relax
7359   \else
7360     \appto#1{#2.}%
7361   \fi
7362   \let\@glxstr@insert@dots@next\@glxstr@insert@dots
7363   \fi
7364   \@glxstr@insert@dots@next#1%
7365 }
```

Similarly provide a way of replacing spaces with `\glxstrwordsep`, which first needs to be defined:

`\glxstrwordsep`

```
7366 \newcommand*{\glxstrwordsep}{\space}
```

Each word is marked with

`\glxtrword`

```
7367 \newcommand*{\glxtrword}[1]{#1}
```

`tr@markwordseps`

```
7368 \newcommand*{\@glxtr@markwordseps}[2]{%
7369   \def#1{}%
7370   \@glxtr@mark@wordseps#1#2 \@nnil
7371 }
```

`r@mark@wordseps`

```
7372 \def\@glxtr@mark@wordseps#1#2 #3{%
7373   \ifdefempty{#1}%
7374   {\def#1{\protect\glxtrword{#2}}}%
7375   {\appto#1{\protect\glxtrwordsep\protect\glxtrword{#2}}}%
7376   \ifx\@nnil#3\relax
7377   \let\@glxtr@mark@wordseps@next\relax
7378   \else
7379   \def\@glxtr@mark@wordseps@next{%
7380     \@glxtr@mark@wordseps#1#3}%
7381   \fi
7382   \@glxtr@mark@wordseps@next
7383 }
```

`newabbreviation` Define a new generic abbreviation.

```
7384 \newcommand*{\newabbreviation}[4][ ]{%
7385   \glxtr@newabbreviation{#1}{#2}{#3}{#4}%
7386 }
```

`newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This is just makes it easier to save and restore the original definition.)

```
7387 \newcommand*{\glxtr@newabbreviation}[4]{%
7388   \glskeylisttok{#1}%
7389   \glslabeltok{#2}%
7390   \glsshorttok{#3}%
7391   \glslongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
7392   \def\glxtrorgshort{#3}%
7393   \def\glxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
7394   \def\ExtraCustomAbbreviationFields{ }%
```

Initialise accessibility settings if required.

```
7395   \@gls@initaccesskeys
```

Get the category.

```
7396   \def\glscategorylabel{abbreviation}%
```

Ignore the shortplural and longplural keys.

```
7397 \setkeys*{glxtrabbrv}[shortplural,longplural]{#1}%
```

Set the abbreviation style.

```
7398 \ifcsdef{@glxabbrv@current@glscategorylabel}%  
7399 {%
```

Warning should already have been issued.

```
7400 \let@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle  
7401 \let\GlsXtrWarnDeprecatedAbbrStyle@gobbletwo  
7402 \glxtr@applyabbrvstyle{\csname @glxabbrv@current@glscategorylabel\endcsname}%  
7403 \let\GlsXtrWarnDeprecatedAbbrStyle@glsxtr@orgwarndep  
7404 }%  
7405 {%
```

If no style has been associated with this category, fallback on the style for the abbreviation category.

```
7406 \glxtr@applyabbrvstyle{@glxabbrv@current@abbreviation}%  
7407 }%
```

Set the default long plural

```
7408 \def@gls@longpl{#4\glspluralsuffix}%  
7409 \let@gls@default@longpl@gls@longpl
```

Has the markwords attribute been set?

```
7410 \glsifcategoryattribute{glscategorylabel}{markwords}{true}%  
7411 {%  
7412 \glxtr@markwordseps@gls@long{#4}%  
7413 \expandafter\def\expandafter@gls@longpl\expandafter  
7414 {\@gls@long\glspluralsuffix}%  
7415 \let@gls@default@longpl@gls@longpl
```

Update \glslongtok.

```
7416 \expandafter\glslongtok\expandafter{@gls@long}%  
7417 }%  
7418 {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
7419 \glsifcategoryattribute{glscategorylabel}{markshortwords}{true}%  
7420 {%  
7421 \glxtr@markwordseps@gls@short{#3}%  
7422 }%  
7423 {%
```

Has the insertdots attribute been set?

```
7424 \glsifcategoryattribute{glscategorylabel}{insertdots}{true}%  
7425 {%  
7426 \glxtr@insertdots@gls@short{#3}%  
  
7427 \appto@gls@short{\@}%  
7428 }%  
7429 {\def@gls@short{#3}}%  
7430 }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
7431 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%  
7432 {%  
7433   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short  
7434     '\abbrvpluralsuffix}%  
7435 }%  
7436 {%
```

Has the noshortplural attribute been set?

```
7437 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%  
7438 {%  
7439   \let\@gls@shortpl\@gls@short  
7440 }%  
7441 {%  
7442   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short  
7443     '\abbrvpluralsuffix}%  
7444 }%  
7445 }%
```

Update \glsshorttok:

```
7446 \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
7447 \glxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
7448 \setkeys*{glxtrabbrv}[category]{#1}%
```

Save in case required.

```
7449 \let\@gls@org@longpl\@gls@longpl  
7450 \let\@gls@org@shortpl\@gls@shortpl
```

Has the plural been explicitly set?

```
7451 \ifx\@gls@default@longpl\@gls@longpl  
7452 \else
```

Has the markwords attribute been set?

```
7453 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%  
7454 {%  
7455   \expandafter\@glxtr@keywordseps\expandafter\@gls@longpl\expandafter  
7456     {\@gls@longpl}%  
7457 }%  
7458 {}%  
7459 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
7460 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%  
7461 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
7462 \@gls@setup@default@access
```


Do any extra setup provided by hook:

7463 \newabbreviationhook

Define this entry:

```
7464 \protected@edef\@do@newglossaryentry{%  
7465   \noexpand\newglossaryentry{\the\glslabeltok}%  
7466   {%  
7467     type=\glstrabbrvtype,%  
7468     category=abbreviation,%  
7469     short={\the\glsshorttok},%  
7470     shortplural={\the\glsshortpltok},%  
7471     long={\the\glslongtok},%  
7472     longplural={\the\glslongpltok},%  
7473     name={\the\glsshorttok},%  
7474     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

7475 \ExtraCustomAbbreviationFields

Any explicit fields set in the optional argument override all other settings.

```
7476   \the\glskeylisttok  
7477   }%  
7478 }%  
7479 \@do@newglossaryentry
```

Obtain the type and add it to the list of abbreviations.

```
7480 \@glstr@addabbreviationlist{\glstrytype{\the\glslabeltok}}%  
7481 \GlsXtrPostNewAbbreviation  
7482 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

7483 \newcommand*{\glstrnewabbspresetkeyhook}[3]{}

NewAbbreviation Hook used by abbreviation styles.

7484 \newcommand*{\GlsXtrPostNewAbbreviation}{}

bbreviationhook Hook for use with \newabbreviation.

7485 \newcommand*{\newabbreviationhook}{}

reviationFields

7486 \newcommand*{\CustomAbbreviationFields}{}

\glstrparen For the parenthetical styles.

7487 \newcommand*{\glstrparen}[1]{(#1)}

lsxtrfullformat Full format without case change.

```
7488 \newcommand*{\glstrfullformat}[2]{%  
7489   \glsfirstlongfont{\glsaccesslong{#1}}#2\glstrfullsep{#1}%  
7490   \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%  
7491 }
```

`\lsxtrfullformat` Full format with case change.

```
7492 \newcommand*{\Glsxtrfullformat}[2]{%
7493   \glsfirstlongfont{\Glsaccesslong{#1}}#2\lsxtrfullsep{#1}%
7494   \lsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
7495 }
```

`\xtrfullplformat` Plural full format without case change.

```
7496 \newcommand*{\glsxtrfullplformat}[2]{%
7497   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\lsxtrfullsep{#1}%
7498   \lsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
7499 }
```

`\xtrfullplformat` Plural full format with case change.

```
7500 \newcommand*{\Glsxtrfullplformat}[2]{%
7501   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\lsxtrfullsep{#1}%
7502   \lsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
7503 }
```

`\glsxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```
7504 \newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

`\nlinefullformat` Full format without case change.

```
7505 \newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}
```

`\nlinefullformat` Full format with case change.

```
7506 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\xtrfullplformat` Plural full format without case change.

```
7507 \newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}
```

`\inefullplformat` Plural full format with case change.

```
7508 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glsentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glsxtrfull` set of commands instead.

`\glsentryfull`

```
7509 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
7510 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glsentryfullpl`

```
7511 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

7512 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

7513 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

7514 `\newcommand*{\glsabbrvdefaultfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use. This is redefined by the abbreviation styles, as appropriate.

7515 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`

7516 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.

7517 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.

7518 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.

7519 `\newcommand*{\glsfirstlongfont}[1]{\glsfirstlongfont{#1}}`

`\glsfirstlongdefaultfont`

7520 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glsfirstlongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

7521 `\newcommand*{\glsbrvpluralsuffix}{\glspluralsuffix}`

`\glsbrvpluralsuffix` Default plural suffix.

7522 `\newcommand*{\glsbrvpluralsuffix}{\glsbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).

7523 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\@ns@glsxtrfull}`

7524 `\newcommand*{\@ns@glsxtrfull}[2][\@gls@hyp@opt\@ns@glsxtrfull]{\@gls@hyp@opt\@ns@glsxtrfull}`

7525 `\new@ifnextchar[\@gls@hyp@opt\@ns@glsxtrfull]{\@gls@hyp@opt\@ns@glsxtrfull}`

7526 `\@gls@hyp@opt\@ns@glsxtrfull`

7527 `\@gls@hyp@opt\@ns@glsxtrfull`

`\@glsxtr@full` Low-level macro:

7528 `\def\@glsxtr@full#1#2[#3]{\@gls@hyp@opt\@ns@glsxtrfull#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).

```

7529 \@glxtr@record{#1}{#2}{glslink}%
7530 \glsdoifexists{#2}%
7531 {%
7532   \glsssetabbrvfmt{\glscategory{#2}}%
7533   \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7534   \let\glslifplural\@secondoftwo
7535   \let\glscapscase\@firstofthree
7536   \let\glslinsert\@empty
7537   \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.

```

7538   \glxtrsetupfulldefs
7539   \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7540 }%
7541 \glspostlinkhook
7542 }

```

trsetupfulldefs

```

7543 \newcommand*{\glxtrsetupfulldefs}{%
7544   \let\glxtrifwasfirstuse\@firstoftwo
7545 }

```

\Glsxtrfull Full form (first letter uppercase).

```

7546 \newrobustcmd*{\Glsxtrfull}{\@glsl@hyp@opt\@ns@Glsxtrfull}
7547 \newcommand*\ns@Glsxtrfull[2][{}]{%
7548   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
7549   {\@Glsxtr@full{#1}{#2}[]}%
7550 }

```

\@Glsxtr@full Low-level macro:

```

7551 \def\@Glsxtr@full#1#2[#3]{%
7552   \glsdoifexists{#2}%
7553   {%
7554     \glsssetabbrvfmt{\glscategory{#2}}%
7555     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7556     \let\glslifplural\@secondoftwo
7557     \let\glscapscase\@secondofthree
7558     \let\glslinsert\@empty
7559     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
7560     \glxtrsetupfulldefs
7561     \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7562   }%
7563   \glspostlinkhook
7564 }

```

`\GLSxtrfull` Full form (all uppercase).

```
7565 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
7566 \newcommand*\ns@GLSxtrfull[2][\%
7567 \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{%
7568     {\@GLSxtr@full{#1}{#2}[]}%
7569 }
```

`\@GLSxtr@full` Low-level macro:

```
7570 \def\@GLSxtr@full#1#2[#3]{%
7571 \glsdoifexists{#2}%
7572 {%
7573 \glssetabbrvfmt{\glscategory{#2}}%
7574 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7575 \let\glsifplural\@secondoftwo
7576 \let\glsupcase\@thirdofthree
7577 \let\glsinsert\@empty
7578 \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
7579 \glsxtrsetupfulldefs
7580 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7581 }%
7582 \glspostlinkhook
7583 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
7584 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
7585 \newcommand*\ns@glsxtrfullpl[2][\%
7586 \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{%
7587     {\@glsxtr@fullpl{#1}{#2}[]}%
7588 }
```

`\@glsxtr@fullpl` Low-level macro:

```
7589 \def\@glsxtr@fullpl#1#2[#3]{%
```

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

```
7590 \@glsxtr@record{#1}{#2}{\glslink}%
7591 \glsdoifexists{#2}%
7592 {%
7593 \glssetabbrvfmt{\glscategory{#2}}%
7594 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7595 \let\glsifplural\@firstoftwo
7596 \let\glsupcase\@firstofthree
7597 \let\glsinsert\@empty
7598 \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
7599 \glsxtrsetupfulldefs
7600 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7601 }%
7602 \glspostlinkhook
7603 }
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
7604 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
7605 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
7606   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}]{%
7607     {\@Glsxtr@fullpl{#1}{#2}[]}%
7608 }
```

`\@Glsxtr@fullpl` Low-level macro:

```
7609 \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).
7610   \@glsxtr@record{#1}{#2}{glslink}%
7611   \glsdoifexists{#2}%
7612   {%
7613     \glssetabbrvfmt{\gls@category{#2}}%
7614     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7615     \let\glsifplural\@firstoftwo
7616     \let\gls@scapscase\@secondofthree
7617     \let\glsinsert\@empty
7618     \def\gls@customtext{\Glsxtrinlinefullplformat{#2}{#3}}%
7619     \glsxtrsetupfulldefs
7620     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7621   }%
7622   \gls@postlinkhook
7623 }
```

`\GLSxtrfullpl` Plural full form (all upper case).

```
7624 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
7625 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
7626   \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}]{%
7627     {\@GLSxtr@fullpl{#1}{#2}[]}%
7628 }
```

`\@GLSxtr@fullpl` Low-level macro:

```
7629 \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).
7630   \@glsxtr@record{#1}{#2}{glslink}%
7631   \glsdoifexists{#2}%
7632   {%
7633     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7634     \let\glsifplural\@firstoftwo
7635     \let\gls@scapscase\@thirdofthree
7636     \let\glsinsert\@empty
7637     \def\gls@customtext{%
7638       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
7639     \glsxtrsetupfulldefs
```

```

7640 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7641 }%
7642 \glspostlinkhook
7643 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

7644 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

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

```

7645 \newcommand*{\ns@glsxtrshort}[2] [] {%
7646 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2} []}%
7647 }

```

Read in the final optional argument:

```

7648 \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).

```

7649 \glsxtr@record{#1}{#2}{glslink}%
7650 \glsdoifexists{#2}%
7651 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

7652 \glssetabbrvfmt{\glscategory{#2}}%
7653 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7654 \let\glsxtrifwasfirstuse\@secondoftwo
7655 \let\glsifplural\@secondoftwo
7656 \let\glscapscase\@firstofthree
7657 \let\glsinsert\@empty
7658 \def\glscustomtext{%
7659 \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7660 \ifglsxtrininsertinside\else#3\fi
7661 }%
7662 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7663 }%
7664 \glspostlinkhook
7665 }

```

`\Glsxtrshort`

```

7666 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

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

```

7667 \newcommand*{\ns@Glsxtrshort}[2] [] {%
7668 \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
7669 }

```

Read in the final optional argument:

```

7670 \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).

```

7671 \@glsxtr@record{#1}{#2}{glslink}%
7672 \glsdoifexists{#2}%
7673 {%
7674   \glssetabbrvfmt{\glscategory{#2}}%
7675   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7676   \let\glsxtrifwasfirstuse\@secondoftwo
7677   \let\glsifplural\@secondoftwo
7678   \let\glscapscase\@secondofthree
7679   \let\glsinsert\@empty
7680   \def\glscustomtext{%
7681     \glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7682     \ifglsxtrininsertinside\else#3\fi
7683   }%
7684   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7685 }%
7686 \glspostlinkhook
7687 }

```

\GLSxtrshort

```

7688 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}

```

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

```

7689 \newcommand*{\ns@GLSxtrshort}[2][\@ns@GLSxtrshort]{%
7690   \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[\@ns@GLSxtrshort]}%
7691 }

```

Read in the final optional argument:

```

7692 \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).

```

7693 \@glsxtr@record{#1}{#2}{glslink}%
7694 \glsdoifexists{#2}%
7695 {%
7696   \glssetabbrvfmt{\glscategory{#2}}%
7697   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7698   \let\glsxtrifwasfirstuse\@secondoftwo
7699   \let\glsifplural\@secondoftwo
7700   \let\glscapscase\@thirdofthree
7701   \let\glsinsert\@empty
7702   \def\glscustomtext{%
7703     \mfirstucMakeUppercase
7704     {\glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
7705     \ifglsxtrininsertinside\else#3\fi
7706   }%
7707 }%
7708 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7709 }%

```



```

7710 \glspostlinkhook
7711 }

```

\glsxtrlong

```

7712 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7713 \newcommand*{\ns@glxtrlong}[2] [] {%
7714   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
7715 }

```

Read in the final optional argument:

```

7716 \def\@glxtrlong#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7717   \@glxtr@record{#1}{#2}{glslink}%
7718   \glsdofexists{#2}%
7719   {%
7720     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7721     \let\glxtrifwasfirstuse\@secondoftwo
7722     \let\glxtrifplural\@secondoftwo
7723     \let\glxtrifscase\@firstofthree
7724     \let\glxtrinsert\@empty
7725     \def\glxtrcustomtext{%
7726       \glslongfont{\glssaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7727       \ifglxtrinsertinside\else#3\fi
7728     }%
7729     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7730   }%
7731   \glspostlinkhook
7732 }

```

\Glsxtrlong

```

7733 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
7734 \newcommand*{\ns@Glsxtrlong}[2] [] {%
7735   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
7736 }

```

Read in the final optional argument:

```

7737 \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).
7738   \@glxtr@record{#1}{#2}{glslink}%
7739   \glsdofexists{#2}%
7740   {%
7741     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7742     \let\glxtrifwasfirstuse\@secondoftwo

```

```

7743 \let\glsifplural\@secondoftwo
7744 \let\glsifscapscase\@secondofthree
7745 \let\glsinsert\@empty
7746 \def\glscustomtext{%
7747   \glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7748   \ifglstrinsertinside\else#3\fi
7749 }%
7750 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7751 }%
7752 \glspostlinkhook
7753 }

```

\GLSxtrlong

```

7754 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}

```

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

```

7755 \newcommand*{\ns@GLSxtrlong}[2] [] {%
7756   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
7757 }

```

Read in the final optional argument:

```

7758 \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).

```

7759   \@glsxtr@record{#1}{#2}{\glslink}%
7760   \glsdoifexists{#2}%
7761   {%
7762     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7763     \let\glsxtrifwasfirstuse\@secondoftwo
7764     \let\glsifplural\@secondoftwo
7765     \let\glsifscapscase\@thirdofthree
7766     \let\glsinsert\@empty
7767     \def\glscustomtext{%
7768       \mfirstucMakeUppercase
7769       {\glsfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
7770       \ifglstrinsertinside\else#3\fi
7771     }%
7772     }%
7773     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7774   }%
7775   \glspostlinkhook
7776 }

```

Plural short forms:

\glsxtrshortpl

```

7777 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

```

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

```

7778 \newcommand*{\ns@glsxtrshortpl}[2] [] {%

```

```

7779 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7780 }

```

Read in the final optional argument:

```

7781 \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).

```

7782 \@glsxtr@record{#1}{#2}{glslink}%
7783 \glsdoifexists{#2}%
7784 {%
7785 \glssetabbrvfmt{\glscategory{#2}}%
7786 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7787 \let\glsxtrifwasfirstuse\@secondoftwo
7788 \let\glsifplural\@firstoftwo
7789 \let\glscapscase\@firstofthree
7790 \let\glsinsert\@empty
7791 \def\glscustomtext{%
7792 \glsabbrvfont{\glssaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7793 \ifglsxtrininsertinside\else#3\fi
7794 }%
7795 \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7796 }%
7797 \glspostlinkhook
7798 }

```

\Glsxtrshortpl

```

7799 \newrobustcmd*{\Glsxtrshortpl}{\@glsl@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

7800 \newcommand*{\@ns@Glsxtrshortpl}[2] []{%
7801 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7802 }

```

Read in the final optional argument:

```

7803 \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).

```

7804 \@glsxtr@record{#1}{#2}{glslink}%
7805 \glsdoifexists{#2}%
7806 {%
7807 \glssetabbrvfmt{\glscategory{#2}}%
7808 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7809 \let\glsxtrifwasfirstuse\@secondoftwo
7810 \let\glsifplural\@firstoftwo
7811 \let\glscapscase\@secondofthree
7812 \let\glsinsert\@empty
7813 \def\glscustomtext{%
7814 \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7815 \ifglsxtrininsertinside\else#3\fi

```

```

7816 }%
7817 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7818 }%
7819 \glspostlinkhook
7820 }

```

\GLSxtrshortpl

```

7821 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7822 \newcommand*{\ns@GLSxtrshortpl}[2][{}]{%
7823   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
7824 }

```

Read in the final optional argument:

```

7825 \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).
7826 \@glsxtr@record{#1}{#2}{glslink}%
7827 \glsdoifexists{#2}%
7828 {%
7829   \glssetabbrvfmt{\glscategory{#2}}%
7830   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7831   \let\glsxtrifwasfirstuse\@secondoftwo
7832   \let\glsifplural\@firstoftwo
7833   \let\glscapscase\@thirdofthree
7834   \let\glsinsert\@empty
7835   \def\glscustomtext{%
7836     \mfirstucMakeUppercase
7837     {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7838     \ifglsxtrininsertinside\else#3\fi
7839   }%
7840 }%
7841 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7842 }%
7843 \glspostlinkhook
7844 }

```

Plural long forms:

\glsxtrlongpl

```

7845 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7846 \newcommand*{\ns@glsxtrlongpl}[2][{}]{%
7847   \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
7848 }
    Read in the final optional argument:
7849 \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).

```

7850 \@glstr@record{#1}{#2}{glslink}%
7851 \glsoifexists{#2}%
7852 {%
7853 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7854 \let\glstrifwasfirstuse\@secondoftwo
7855 \let\gl@ifplural\@firstoftwo
7856 \let\glscapscase\@firstofthree
7857 \let\glinsert\@empty
7858 \def\glscustomtext{%
7859 \glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7860 \ifglstrinsertinside\else#3\fi
7861 }%
7862 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7863 }%
7864 \glspostlinkhook
7865 }

```

\Glsxtrlongpl

```

7866 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}

```

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

```

7867 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7868 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7869 }

```

Read in the final optional argument:

```

7870 \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).

```

7871 \@glstr@record{#1}{#2}{glslink}%
7872 \glsoifexists{#2}%
7873 {%
7874 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7875 \let\glstrifwasfirstuse\@secondoftwo
7876 \let\gl@ifplural\@firstoftwo
7877 \let\glscapscase\@secondofthree
7878 \let\glinsert\@empty
7879 \def\glscustomtext{%
7880 \glslongfont{\Glsaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7881 \ifglstrinsertinside\else#3\fi
7882 }%
7883 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7884 }%
7885 \glspostlinkhook
7886 }

```

\GLSxtrlongpl

```

7887 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7888 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7889   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7890 }

    Read in the final optional argument:
7891 \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).
7892   \@glsxtr@record{#1}{#2}{glslink}%
7893   \glsdoifexists{#2}%
7894   {%
7895     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7896     \let\glsxtrifwasfirstuse\@secondoftwo
7897     \let\glsifplural\@firstoftwo
7898     \let\glsapscase\@thirdofthree
7899     \let\glsinsert\@empty
7900     \def\glscustomtext{%
7901       \mfirstucMakeUppercase
7902       {\glsfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7903       \ifglsxtrininsertinside\else#3\fi
7904     }%
7905     }%
7906     \@gls@link{#1}{#2}{\csname gls@glstype @entryfmt\endcsname}%
7907   }%
7908   \glspostlinkhook
7909 }

```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

7910 \newcommand*{\glssetabbrvfmt}[1]{%
7911   \ifcsdef{\glsabbrv@current@#1}%
7912   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7913   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7914 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7915 \newrobustcmd*{\glsuseabbrvfont}[2]{{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```

7916 \newrobustcmd*{\glsuselongfont}[2]{{\glssetabbrvfmt{#2}\glsfont{#1}}}

```

sxtrgenabbrvfmt Similar to \glsacfm, but for abbreviations.

```

7917 \newcommand*{\glsxtrgenabbrvfmt}{%
7918   \ifdefempty\glscustomtext
7919   {%
7920     \ifglsused\glslabel
7921     {%

```

Subsequent use:

7922 \glsifplural
7923 {%

Subsequent plural form:

7924 \glscapscase
7925 {%

Subsequent plural form, don't adjust case:

7926 \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7927 }%
7928 {%

Subsequent plural form, make first letter upper case:

7929 \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7930 }%
7931 {%

Subsequent plural form, all caps:

7932 \mfirstucMakeUppercase
7933 {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7934 }%
7935 }%
7936 {%

Subsequent singular form

7937 \glscapscase
7938 {%

Subsequent singular form, don't adjust case:

7939 \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7940 }%
7941 {%

Subsequent singular form, make first letter upper case:

7942 \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7943 }%
7944 {%

Subsequent singular form, all caps:

7945 \mfirstucMakeUppercase
7946 {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7947 }%
7948 }%
7949 }%
7950 {%

First use:

7951 \glsifplural
7952 {%

First use plural form:

7953 \glscapscase
7954 {%

First use plural form, don't adjust case:

```
7955      \glstrfullplformat{\glslabel}{\glsinsert}%  
7956      }%  
7957      {%
```

First use plural form, make first letter upper case:

```
7958      \Glsxtrfullplformat{\glslabel}{\glsinsert}%  
7959      }%  
7960      {%
```

First use plural form, all caps:

```
7961      \mfirstucMakeUppercase  
7962      {\glstrfullplformat{\glslabel}{\glsinsert}}%  
7963      }%  
7964      }%  
7965      {%
```

First use singular form

```
7966      \glscapscase  
7967      {%
```

First use singular form, don't adjust case:

```
7968      \glstrfullformat{\glslabel}{\glsinsert}%  
7969      }%  
7970      {%
```

First use singular form, make first letter upper case:

```
7971      \Glsxtrfullformat{\glslabel}{\glsinsert}%  
7972      }%  
7973      {%
```

First use singular form, all caps:

```
7974      \mfirstucMakeUppercase  
7975      {\glstrfullformat{\glslabel}{\glsinsert}}%  
7976      }%  
7977      }%  
7978      }%  
7979      }%  
7980      {%
```

User supplied text.

```
7981      \glscustomtext  
7982      }%  
7983 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7984 \newcommand*{\glstrsubsequentfmt}[2]{%  
7985   \glabbrvfont{\glsaccessshort{#1}\ifglstrinsertinside #2\fi}%  
7986   \ifglstrinsertinside \else#2\fi  
7987 }  
7988 \let\glstrdefaultsubsequentfmt\glstrsubsequentfmt
```



```

subsequentplfmt Subsequent use format (plural no case change).
7989 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7990   \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7991   \ifglxtrininsertinside \else#2\fi
7992 }
7993 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
7994 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7995   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7996   \ifglxtrininsertinside \else#2\fi
7997 }
7998 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).
7999 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
8000   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
8001   \ifglxtrininsertinside \else#2\fi
8002 }
8003 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

```

breviationstyle
8004 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
8005   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
8006   {%
8007     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
8008   }%
8009   {%
      Have abbreviations already been defined for this category?
8010     \ifcsstring{@glsabbrv@current@#1}{#2}%
8011     {%
      Style already set.
8012     }%
8013     {%
8014       \def\@glxtr@dostylewarn{%
8015         \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
8016         {%
8017           \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
8018             style has been switched \MessageBreak
8019             for category ‘#1’, \MessageBreak
8020             but there have already been entries \MessageBreak
8021             defined for this category. Unwanted \MessageBreak
8022             side-effects may result}}%
8023           \@endfortrue
8024         }%
8025         \@glxtr@dostylewarn

```

Set up the style for the given category.

```
8026      \csdef{@glsabbrv@current@#1}{#2}%
8027      \edef\glscategorylabel{#1}%
8028      \glsxtr@applyabbrvstyle{#2}%
8029      }%
8030      }%
8031 }
```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```
8032 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
8033   \csuse{@glsabbrv@dispstyle@setup@#1}%
8034   \csuse{@glsabbrv@dispstyle@fmts@#1}%
8035 }
```

`\r@applyabbrvfmt` Only apply the style formats.

```
8036 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
8037   \csuse{@glsabbrv@dispstyle@fmts@#1}%
8038 }
```

`\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.

```
8039 \newcommand*{\newabbreviationstyle}[3]{%
8040   \ifcsdef{@glsabbrv@dispstyle@setup@#1}%
8041   {%
8042     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
8043       defined}{}%
8044   }%
8045   {%
8046     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```
8047     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
8048     #2}%
8049     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```
8050     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
8051     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
8052     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
8053     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```
8054     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
8055     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
8056     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
8057     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
8058     #3}%
8059   }%
8060 }
```

renewabbreviationstyle

```
8061 \newcommand*{\renewabbreviationstyle}[3]{%
8062   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
8063   {%
8064     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
8065   }%
8066   {%
8067     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
8068       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
8069       #2}%
8070     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
8071       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
8072       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
8073       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
8074       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
8075       #3}%
8076     }%
8077 }
```

renewabbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```
8078 \newcommand*{\letabbreviationstyle}[2]{%
8079   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
8080   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
8081 }
```

renewdeprecated@abbrstyle

`\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```
8082 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
8083   \csdef{@glsabbrv@dispstyle@setup@#1}{%
8084     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
8085     \csuse{@glsabbrv@dispstyle@setup@#2}%
8086   }%
8087   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
8088 }
```

renewdeprecatedAbbrStyle Generate warning for deprecated style use.

```
8089 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
8090   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
8091     use ‘#2’ instead}%
8092 }
```

eAbbrStyleSetup

```

8093 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
8094   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
8095   {%
8096     \PackageError{glossaries-extra}%
8097     {Unknown abbreviation style definitions ‘#1’}{}%
8098   }%
8099   {%
8100     \csname @glsabbrv@dispstyle@setup@#1\endcsname
8101   }%
8102 }

```

seAbbrStyleFmts

```

8103 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
8104   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
8105   {%
8106     \PackageError{glossaries-extra}%
8107     {Unknown abbreviation style formats ‘#1’}{}%
8108   }%
8109   {%
8110     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
8111   }%
8112 }

```

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.

```

8113 \newif\ifglsxtrinsertinside
8114 \glsxtrinsertinsidefalse

```

trlongshortname

```

8115 \newcommand*{\glsxtrlongshortname}{%
8116   \protect\glsabbrvfont{\the\glsshorttok}%
8117 }

```

long-short

```

8118 \newabbreviationstyle{long-short}%
8119 {%

```

Set accessibility attributes if enabled.

```
8120 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
8121 \renewcommand*{\CustomAbbreviationFields}{%
8122   name={\glxtrlongshortname},
8123   sort={\the\glsshorttok},
8124   first={\protect\glsfirstlongfont{\the\glslongtok}%
8125     \protect\glxtrfullsep{\the\glslabeltok}%
8126     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
8127   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
8128     \protect\glxtrfullsep{\the\glslabeltok}%
8129     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

8130   plural={\protect\glsabbrvfont{\the\glsshortpltok}}},%
8131   text={\protect\glsabbrvfont{\the\glsshorttok}}},%
8132   description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
8133 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8134   \glshasattribute{\the\glslabeltok}{regular}%
8135   {%
8136     \glissetattribute{\the\glslabeltok}{regular}{false}%
8137   }%
8138   {}%
8139 }%
8140}%
8141{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
8142 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8143 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8144 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8145 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8146 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8147 \renewcommand*{\glxtrfullformat}[2]{%
8148   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8149   \ifglxtrininsertinside\else##2\fi
8150   \glxtrfullsep{##1}%
8151   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8152 }%
8153 \renewcommand*{\glxtrfullplformat}[2]{%
8154   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8155   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8156   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8157 }%
8158 \renewcommand*{\Glsxtrfullformat}[2]{%
8159   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8160   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

8161 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8162 }%
8163 \renewcommand*{\Glsxtrfullplformat}[2]{%
8164 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8165 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8166 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8167 }%
8168 }

```

Set this as the default style for general abbreviations:

```
8169 \setabbreviationstyle{long-short}
```

ngshortdescsort

```

8170 \newcommand*{\glsxtrlongshortdescsort}{%
8171 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
8172 }

```

ngshortdescname

```

8173 \newcommand*{\glsxtrlongshortdescname}{%
8174 \protect\glslongfont{\the\glslongtok}
8175 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
8176 }

```

long-short-desc User supplies description. The long form is included in the name.

```

8177 \newabbreviationstyle{long-short-desc}%
8178 {%

```

Set accessibility attributes if enabled.

```
8179 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```

8180 \renewcommand*{\CustomAbbreviationFields}{%
8181 name={\glsxtrlongshortdescname},
8182 sort={\glsxtrlongshortdescsort},%
8183 first={\protect\glsfirstlongfont{\the\glslongtok}%
8184 \protect\glsxtrfullsep{\the\glslabeltok}%
8185 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
8186 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
8187 \protect\glsxtrfullsep{\the\glslabeltok}%
8188 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

8189 text={\protect\glsabbrvfont{\the\glsshorttok}},%
8190 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
8191 }%

```

Unset the regular attribute if it has been set.

```

8192 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8193 \glshasattribute{\the\glslabeltok}{regular}%

```

```

8194    {%
8195        \glsssetattribute{\the\glslabelltok}{regular}{false}%
8196    }%
8197    {}%
8198 }%
8199 }%
8200 {%
8201     \GlsXtrUseAbbrStyleFmts{long-short}%
8202 }

```

trshortlongname

```

8203 \newcommand*{\glxtrshortlongname}{%
8204     \protect\glsabbrvfont{\the\glsshorttok}%
8205 }

```

short-long Short form followed by long form in parenthesis on first use.

```

8206 \newabbreviationstyle{short-long}%
8207 {%

```

Set accessibility attributes if enabled.

```

8208     \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

8209     \renewcommand*{\CustomAbbreviationFields}{%
8210         name={\glxtrshortlongname},
8211         sort={\the\glsshorttok},
8212         description={\the\glslongtok},%
8213         first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
8214             \protect\glxtrfullsep{\the\glslabelltok}%
8215             \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
8216         firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8217             \protect\glxtrfullsep{\the\glslabelltok}%
8218             \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
8219         text={\protect\glsabbrvfont{\the\glsshorttok}},%
8220         plural={\protect\glsabbrvfont{\the\glsshortpltok}}}

```

Unset the regular attribute if it has been set.

```

8221     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8222         \glshasattribute{\the\glslabelltok}{regular}%
8223         {%
8224             \glsssetattribute{\the\glslabelltok}{regular}{false}%
8225         }%
8226         {}%
8227     }%
8228 }%
8229 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8230     \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8231     \renewcommand*{\glsabbrvfont[1]}{\glsabbrvdefaultfont{##1}}%

```

```

8232 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8233 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8234 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8235 \renewcommand*{\glsxtrfullformat}[2]{%
8236   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8237   \ifglsxtrinsertinside\else##2\fi
8238   \glsxtrfullsep{##1}%
8239   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8240 }%
8241 \renewcommand*{\glsxtrfullplformat}[2]{%
8242   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8243   \ifglsxtrinsertinside\else##2\fi
8244   \glsxtrfullsep{##1}%
8245   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8246 }%
8247 \renewcommand*{\Glsxtrfullformat}[2]{%
8248   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8249   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8250   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8251 }%
8252 \renewcommand*{\Glsxtrfullplformat}[2]{%
8253   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8254   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8255   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8256 }%
8257 }

```

ortlongdescsort

```

8258 \newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

8259 \newcommand*{\glsxtrshortlongdescname}{%
8260   \protect\glsabbrvfont{\the\glsshorttok}
8261   \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
8262 }

```

short-long-desc User supplies description. The long form is included in the name.

```

8263 \newabbreviationstyle{short-long-desc}%
8264 {%

```

Set accessibility attributes if enabled.

```

8265 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

8266 \renewcommand*{\CustomAbbreviationFields}{%
8267   name={\glsxtrshortlongdescname},
8268   sort={\glsxtrshortlongdescsort},
8269   first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%

```



```

8270 \protect\glxtrfullsep{\the\glslabeltok}%
8271 \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
8272 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8273 \protect\glxtrfullsep{\the\glslabeltok}%
8274 \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%

8275 text={\protect\glsabbrvfont{\the\glsshortttok}}},%

8276 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
8277 }%

Unset the regular attribute if it has been set.
8278 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8279 \glshasattribute{\the\glslabeltok}{regular}%
8280 {%
8281 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8282 }%
8283 {}%
8284 }%
8285 }%
8286 {%
8287 \GlsXtrUseAbbrStyleFmts{short-long}%
8288 }

```

`\glslongfootnotefont` Only used by the “footnote” styles.

```
8289 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

`\glsshortfootnotefont` Only used by the “footnote” styles.

```
8290 \newcommand*{\glsshortfootnotefont}[1]{\glsshortfootnotefont{#1}}%
```

`\glstrabbrvfootnote`

`\glstrabbrvfootnote{<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`).

```
8291 \newcommand*{\glstrabbrvfootnote}[2]{\footnote{#2}}
```

`\glstrfootnotename`

```

8292 \newcommand*{\glstrfootnotename}{%
8293 \protect\glsabbrvfont{\the\glsshortttok}%
8294 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

8295 \newabbreviationstyle{footnote}%
8296 {}

```

Set accessibility attributes if enabled. (Add firstshortaccess since long form is hidden in a footnote on first use.)

```
8297 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
8298 \renewcommand*{\CustomAbbreviationFields}{%
8299   name={\glxtrfootnotename},
8300   sort={\the\glsshorttok},
8301   description={\the\glslongtok},%

8302   first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
8303     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8304     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8305   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8306     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8307     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

8308   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8309   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.
```

```
8310 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8311   \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%
8312   \glshasattribute{\the\glslabeltok}{regular}%
8313   {%
8314     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8315   }%
8316   {}%
8317 }%
8318 }%
8319 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
8320 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8321 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8322 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8323 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8324 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
8325 \renewcommand*{\glxtrfullformat}[2]{%
8326   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8327   \ifglxtrininsertinside\else##2\fi
8328   \protect\glxtrabbrvfootnote{##1}%
8329   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8330 }%
8331 \renewcommand*{\glxtrfullplformat}[2]{%
8332   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8333   \ifglxtrininsertinside\else##2\fi
8334   \protect\glxtrabbrvfootnote{##1}%
```

```

8335     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8336 }%
8337 \renewcommand*{\Glsxtrfullformat}[2]{%
8338   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8339   \ifglxtrinsertinside\else##2\fi
8340   \protect\glxtrabbrvfootnote{##1}%
8341   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8342 }%
8343 \renewcommand*{\Glsxtrfullplformat}[2]{%
8344   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8345   \ifglxtrinsertinside\else##2\fi
8346   \protect\glxtrabbrvfootnote{##1}%
8347   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8348 }%

```

The first use full form and the inline full form use the short (long) style.

```

8349 \renewcommand*{\glxtrinlinefullformat}[2]{%
8350   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8351   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8352   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8353 }%
8354 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8355   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8356   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8357   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8358 }%
8359 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8360   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8361   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8362   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8363 }%
8364 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8365   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8366   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8367   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8368 }%
8369 }

```

short-footnote

```

8370 \letabbreviationstyle{short-footnote}{footnote}

```

footnotedesname

```

8371 \newcommand*{\glxtrfootnotedesname}{%
8372   \protect\glsabbrvfont{\the\glsshorttok}%
8373   \protect\glxtrfullsep{\the\glslabeltok}%
8374   \protect\glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
8375 }

```

footnotedescsort

```

8376 \newcommand*{\glxtrfootnotedescsort}{\the\glsshorttok}

```

t-footnote-desc Like short-footnote but with user supplied description.

```
8377 \newabbreviationstyle{short-footnote-desc}%  
8378 {%
```

Set accessibility attributes if enabled

```
8379 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
8380 \renewcommand*{\CustomAbbreviationFields}{%  
8381   name={\glxtrfootnotedesname},  
8382   sort={\glxtrfootnotedesort},  
8383   first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%  
8384   \protect\glxtrabbrvfootnote{\the\glslabeltok}}%  
8385   {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%  
8386   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%  
8387   \protect\glxtrabbrvfootnote{\the\glslabeltok}}%  
8388   {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%  
8389   text={\protect\glsabrvfont{\the\glsshorttok}}},%  
8390   plural={\protect\glsabrvfont{\the\glsshortpltok}}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
8391 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
8392   \glsetattribute{\the\glslabeltok}{nohyperfirst}{true}%  
8393   \glshasattribute{\the\glslabeltok}{regular}%  
8394   {%  
8395     \glsetattribute{\the\glslabeltok}{regular}{false}%  
8396   }%  
8397   {}%  
8398 }%  
8399 }%  
8400 {%  
8401   \GlsXtrUseAbbrStyleFmts{footnote}%  
8402 }
```

footnote-desc Synonym.

```
8403 \letabbreviationstyle{footnote-desc}{short-footnote-desc}
```

postfootnote Similar to footnote 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.

```
8404 \newabbreviationstyle{postfootnote}%  
8405 {%
```

Set accessibility attributes if enabled. (Add firstshortaccess since long form is hidden in a footnote on first use.)

```
8406 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
8407 \renewcommand*{\CustomAbbreviationFields}{%
```

```

8408   name={\glxtrfootnotename},
8409   sort={\the\glsshorttok},
8410   description={\the\glslongtok},%
8411   first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
8412   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%

8413   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8414   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.

```

8415   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8416     \csdef{glsxtrpostlink\glscategorylabel}{%
8417       \glxtrifwasfirstuse
8418       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8419         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
8420         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8421     }%
8422   }%
8423 }%
8424   \glshasattribute{\the\glslabeltok}{regular}%
8425   {%
8426     \glssetattribute{\the\glslabeltok}{regular}{false}%
8427   }%
8428   {}%
8429 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8430   \renewcommand*{\glxtrsetupfulldefs}{%
8431     \let\glxtrifwasfirstuse\@secondoftwo
8432   }%
8433 }%
8434 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8435   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8436   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8437   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8438   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8439   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8440   \renewcommand*{\glxtrfullformat}[2]{%
8441     \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8442     \ifglxtrininsertinside\else##2\fi
8443   }%
8444   \renewcommand*{\glxtrfullplformat}[2]{%

```

```

8445 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8446 \ifglxtrinsertinside\else##2\fi
8447 }%
8448 \renewcommand*{\Glsxtrfullformat}[2]{%
8449 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8450 \ifglxtrinsertinside\else##2\fi
8451 }%
8452 \renewcommand*{\Glsxtrfullplformat}[2]{%
8453 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8454 \ifglxtrinsertinside\else##2\fi
8455 }%

```

The first use full form and the inline full form use the short (long) style.

```

8456 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8457 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8458 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8459 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8460 }%
8461 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8462 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8463 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8464 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8465 }%
8466 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8467 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8468 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8469 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8470 }%
8471 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8472 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8473 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8474 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8475 }%
8476 }

```

rt-postfootnote

```
8477 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

stfootnote-desc Like short-postfootnote but with user supplied description.

```

8478 \newabbreviationstyle{short-postfootnote-desc}%
8479 {%

```

Set accessibility attributes if enabled.

```
8480 \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```

8481 \renewcommand*{\CustomAbbreviationFields}{%
8482 name={\glsxtrfootnotedesname},
8483 sort={\glsxtrfootnotedesort},
8484 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
8485 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%

```

```

8486   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8487   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.

```

8488   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8489     \csdef{glsxtrpostlink\glscategorylabel}{%
8490       \glsxtrifwasfirstuse
8491       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8492         \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
8493         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8494     }%
8495   }%
8496 }%
8497 \glshasattribute{\the\glslabeltok}{regular}%
8498 {%
8499   \glssetattribute{\the\glslabeltok}{regular}{false}%
8500 }%
8501 {}%
8502 }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

8503   \renewcommand*{\glsxtrsetupfulldefs}{%
8504     \let\glsxtrifwasfirstuse\@secondoftwo
8505   }%
8506 }%
8507 {%
8508   \GlsXtrUseAbbrStyleFmts{postfootnote}%
8509 }

```

stfootnote-desc

```

8510 \letabbreviationstyle{postfootnote-desc}{short-postfootnote-desc}

```

shortnolongname

```

8511 \newcommand*{\glsxtrshortnolongname}{%
8512   \protect\glsabbrvfont{\the\glsshorttok}}%
8513 }

```

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.

```

8514 \newabbreviationstyle{short}%
8515 {%

```

Set accessibility attributes if enabled.

```
8516 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
8517 \renewcommand*{\CustomAbbreviationFields}{%
8518   name={\glxtrshortnolongname},
8519   sort={\the\glsshorttok},
8520   first={\protect\glfirstabbrvfont{\the\glsshorttok}},
8521   firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}},
8522   text={\protect\glabbrvfont{\the\glsshorttok}},
8523   plural={\protect\glabbrvfont{\the\glsshortpltok}},
8524   description={\the\glslongtok}}%
8525 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8526   \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8527 }%
8528 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
8529 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8530 \renewcommand*{\glabbrvfont}[1]{\glabbrvdefaultfont{##1}}%
8531 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvdefaultfont{##1}}%
8532 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8533 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
8534 \renewcommand*{\glxtrinlinefullformat}[2]{%
8535   \protect\glfirstabbrvfont{\glaccessshort{##1}}%
8536   \ifglxtrininsertinside##2\fi}%
8537 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8538 \glxtrparen{\glfirstlongfont{\glaccesslong{##1}}}%
8539 }%
8540 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8541   \protect\glfirstabbrvfont{\glaccessshortpl{##1}}%
8542   \ifglxtrininsertinside##2\fi}%
8543 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8544 \glxtrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
8545 }%
8546 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8547   \protect\glfirstabbrvfont{\glaccessshort{##1}}%
8548   \ifglxtrininsertinside##2\fi}%
8549 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8550 \glxtrparen{\glfirstlongfont{\glaccesslong{##1}}}%
8551 }%
8552 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8553   \protect\glfirstabbrvfont{\glaccessshortpl{##1}}%
8554   \ifglxtrininsertinside##2\fi}%
8555 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8556 \glxtrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
8557 }%
```


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.

```

8558 \renewcommand*{\glxtrfullformat}[2]{%
8559   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8560   \ifglxtrinsertinside\else##2\fi
8561 }%
8562 \renewcommand*{\glxtrfullplformat}[2]{%
8563   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8564   \ifglxtrinsertinside\else##2\fi
8565 }%
8566 \renewcommand*{\Glsxtrfullformat}[2]{%
8567   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8568   \ifglxtrinsertinside\else##2\fi
8569 }%
8570 \renewcommand*{\Glsxtrfullplformat}[2]{%
8571   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8572   \ifglxtrinsertinside\else##2\fi
8573 }%
8574 }

```

Set this as the default style for acronyms:

```

8575 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

8576 \letabbreviationstyle{short-nolong}{short}

```

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

8577 \newabbreviationstyle{short-nolong-noreg}%
8578 {%
8579   \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

8580 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8581   \glshasattribute{\the\glslabeltok}{regular}%
8582   {%
8583     \glssetattribute{\the\glslabeltok}{regular}{false}%
8584   }%
8585   {}}%
8586 }%
8587 }%
8588 {%
8589   \GlsXtrUseAbbrStyleFmts{short-nolong}%
8590 }

```

trshortdescname

```

8591 \newcommand*{\glxtrshortdescname}{%
8592   \protect\glsabbrvfont{\the\glsshorttok}%
8593   \protect\glxtrfullsep{\the\glslabeltok}%
8594   \protect\glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
8595 }

```

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.

```
8596 \newabbreviationstyle{short-desc}%
8597 {%
```

Set accessibility attributes if enabled.

```
8598 \glstrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
8599 \renewcommand*{\CustomAbbreviationFields}{%
8600   name={\glstrshortdescname},
8601   sort={\the\glsshorttok},
8602   first={\protect\glfirstabbrvfont{\the\glsshorttok}},
8603   firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}},
8604   text={\protect\glabbrvfont{\the\glsshorttok}},
8605   plural={\protect\glabbrvfont{\the\glsshortpltok}}}%
8606 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8607   \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8608 }%
8609 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
8610 \renewcommand*{\abbrvpluralsuffix}{\glstrabbrvpluralsuffix}%
8611 \renewcommand*{\glabbrvfont[1]{\glabbrvdefaultfont{##1}}}%
8612 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvdefaultfont{##1}}%
8613 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8614 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8615 \renewcommand*{\glxtrinlinefullformat}[2]{%
8616   \glfirstabbrvfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8617   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8618   \glxtrparen{\glfirstlongfont{\glaccesslong{##1}}}%
8619 }%
8620 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8621   \glfirstabbrvfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8622   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8623   \glxtrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
8624 }%
8625 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8626   \glfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8627   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8628   \glxtrparen{\glfirstlongfont{\glaccesslong{##1}}}%
8629 }%
8630 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8631   \glfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8632   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8633   \glxtrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
8634 }%
```

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.

```

8635 \renewcommand*{\glxtrfullformat}[2]{%
8636   \glsfirstabbrvfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8637   \ifglxtrinsertinside\else##2\fi
8638 }%
8639 \renewcommand*{\glxtrfullplformat}[2]{%
8640   \glsfirstabbrvfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8641   \ifglxtrinsertinside\else##2\fi
8642 }%
8643 \renewcommand*{\Glsxtrfullformat}[2]{%
8644   \glsfirstabbrvfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8645   \ifglxtrinsertinside\else##2\fi
8646 }%
8647 \renewcommand*{\Glsxtrfullplformat}[2]{%
8648   \glsfirstabbrvfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8649   \ifglxtrinsertinside\else##2\fi
8650 }%
8651 }

```

short-nolong-desc

```

8652 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

8653 \newabbreviationstyle{short-nolong-desc-noreg}%
8654 {%
8655   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
8656   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8657     \glshasattribute{\the\glslabeltok}{regular}%
8658     {%
8659       \glsssetattribute{\the\glslabeltok}{regular}{false}%
8660     }%
8661     {}%
8662   }%
8663 }%
8664 {%
8665   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
8666 }

```

Unset the regular attribute if it has been set.

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

8667 \newabbreviationstyle{nolong-short}%
8668 {%
8669   \GlsXtrUseAbbrStyleSetup{short-nolong}%
8670 }%
8671 {%
8672   \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8673 \renewcommand*{\glxtrinlinefullformat}[2]{%
8674 \protect\glsfirstlongfont{\glsaccesslong{##1}%
8675 \ifglxtrinsertinside##2\fi}%
8676 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8677 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8678 }%
8679 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8680 \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8681 \ifglxtrinsertinside##2\fi}%
8682 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8683 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8684 }%
8685 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8686 \protect\glsfirstlongfont{\glsaccesslong{##1}%
8687 \ifglxtrinsertinside##2\fi}%
8688 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8689 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8690 }%
8691 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8692 \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8693 \ifglxtrinsertinside##2\fi}%
8694 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8695 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8696 }%
8697 }

```

`long-short-noreg` Like `long-short` but doesn't set the regular attribute.

```

8698 \newabbreviationstyle{long-short-noreg}%
8699 {%
8700 \GlsXtrUseAbbrStyleSetup{long-short}%

```

Unset the regular attribute if it has been set.

```

8701 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8702 \glshasattribute{\the\glslabeltok}{regular}%
8703 {%
8704 \glssetattribute{\the\glslabeltok}{regular}{false}%
8705 }%
8706 {}%
8707 }%
8708 }%
8709 {%
8710 \GlsXtrUseAbbrStyleFmts{long-short}%
8711 }

```

`noshortdescname`

```

8712 \newcommand*{\glxtrlongnoshortdescname}{%
8713 \protect\glslongfont{\the\glslongtok}%
8714 }

```

`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. The accessibility attributes don’t need setting here.

```

8715 \newabbreviationstyle{long-desc}%
8716 {%
8717   \renewcommand*{\CustomAbbreviationFields}{%
8718     name={\glxtrlongnoshortdescname},
8719     sort={\the\glslongtok},
8720     first={\protect\glslongfont{\the\glslongtok}},
8721     firstplural={\protect\glslongfont{\the\glslongpltok}},
8722     text={\glslongfont{\the\glslongtok}},
8723     plural={\glslongfont{\the\glslongpltok}}}%
8724   }%
8725   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8726     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8727 }%
8728 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8729 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8730 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
8731 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
8732 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8733 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8734 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8735   \glslongfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8736   \ifglsxtrininsertinside \else##2\fi
8737 }%
8738 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8739   \glslongfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8740   \ifglsxtrininsertinside \else##2\fi
8741 }%
8742 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8743   \glslongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8744   \ifglsxtrininsertinside \else##2\fi
8745 }%
8746 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8747   \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8748   \ifglsxtrininsertinside \else##2\fi
8749 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8750 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8751   \glssfirstlongfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8752   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8753   \glsxtrparen{\protect\glssfirstabbrvfont{\glssaccessshort{##1}}}%
8754 }%

```

```

8755 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8756   \glsfirslongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8757   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8758   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8759 }%
8760 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8761   \glsfirslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8762   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8763   \glxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8764 }%
8765 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8766   \glsfirslongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8767   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8768   \glxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8769 }%

```

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.

```

8770 \renewcommand*{\glxtrfullformat}[2]{%
8771   \glsfirslongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8772   \ifglxtrinsertinside\else##2\fi
8773 }%
8774 \renewcommand*{\glxtrfullplformat}[2]{%
8775   \glsfirslongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8776   \ifglxtrinsertinside\else##2\fi
8777 }%
8778 \renewcommand*{\Glsxtrfullformat}[2]{%
8779   \glsfirslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8780   \ifglxtrinsertinside\else##2\fi
8781 }%
8782 \renewcommand*{\Glsxtrfullplformat}[2]{%
8783   \glsfirslongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8784   \ifglxtrinsertinside\else##2\fi
8785 }%
8786 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```
8787 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```

8788 \newabbreviationstyle{long-noshort-desc-noreg}%
8789 {%
8790   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

Unset the regular attribute if it has been set.

```

8791 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8792   \glshasattribute{\the\glslabeltok}{regular}%
8793   {%
8794     \glissetattribute{\the\glslabeltok}{regular}{false}%
8795   }%

```

```

8796     {}%
8797   }%
8798 }%
8799 {%
8800   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8801 }

```

longnoshortname

```

8802 \newcommand*{\glxtrlongnoshortname}{%
8803   \protect\glsabbrvfont{\the\glsshorttok}%
8804 }

```

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.

```

8805 \newabbreviationstyle{long}%
8806 {%

```

Set accessibility attributes if enabled.

```

8807   \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

8808   \renewcommand*{\CustomAbbreviationFields}{%
8809     name={\glxtrlongnoshortname},
8810     sort={\the\glsshorttok},
8811     first={\protect\glsfirstlongfont{\the\glslongtok}},
8812     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8813     text={\glslongfont{\the\glslongtok}},
8814     plural={\glslongfont{\the\glslongpltok}},%
8815     description={\the\glslongtok}%
8816   }%
8817   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8818     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8819 }%
8820 {%
8821   \GlsXtrUseAbbrStyleFmts{long-desc}%
8822 }

```

long-noshort Provide a synonym that matches similar styles.

```

8823 \letabbreviationstyle{long-noshort}{long}

```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```

8824 \newabbreviationstyle{long-noshort-noreg}%
8825 {%
8826   \GlsXtrUseAbbrStyleSetup{long-noshort}%

```

Unset the regular attribute if it has been set.

```

8827   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8828     \glshassetAttribute{\the\glslabeltok}{regular}%
8829   }%

```

```

8830     \glsetattribute{\the\glslabeltok}{regular}{false}%
8831     }%
8832     {}%
8833     }%
8834 }%
8835 {%
8836   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8837 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
8838 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glssabbrvscfont` Added for consistent naming.

```
8839 \newcommand*{\glssabbrvscfont}{\glsxtrscfont}
```

`\glsxtrfirstscfont` Maintained for backward-compatibility.

```
8840 \newcommand*{\glsxtrfirstscfont}[1]{\glssabbrvscfont{#1}}
```

`\glssfirstabbrvscfont` Added for consistent naming.

```
8841 \newcommand*{\glssfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix` `\protect` needs to come inside `\s` to avoid interfering with all caps.

```
8842 \newcommand*{\glsxtrscsuffix}{\protect\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
8843 \newabbreviationstyle{long-short-sc}%
8844 {%

```

Set accessibility attributes if enabled.

```
8845   \glsxtrAccSuppAbbrSetFirstLongAttr{\glscategorylabel}
```

Setup the default fields.

```

8846   \renewcommand*{\CustomAbbreviationFields}{%
8847     name={\glsxtrlongshortname},
8848     sort={\the\glsshorttok},
8849     first={\protect\glssfirstlongdefaultfont{\the\glslongtok}%
8850       \protect\glsxtrfullsep{\the\glslabeltok}%
8851       \glsxtrparen{\protect\glssfirstabbrvscfont{\the\glsshorttok}}},%
8852     firstplural={\protect\glssfirstlongdefaultfont{\the\glslongpltok}%
8853       \protect\glsxtrfullsep{\the\glslabeltok}%
8854       \glsxtrparen{\protect\glssfirstabbrvscfont{\the\glsshortpltok}}},%
8855     text={\protect\glssabbrvscfont{\the\glsshorttok}},%
8856     plural={\protect\glssabbrvscfont{\the\glsshortpltok}},%

```



```

8857     description={\the\glslongtok}}%
8858 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8859     \glshasattribute{\the\glslabeltok}{regular}%
8860     {%
8861         \glsetattribute{\the\glslabeltok}{regular}{false}%
8862     }%
8863     {}%
8864 }%
8865 }%
8866 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8867 \renewcommand*\abbrvpluralsuffix}{\glsxtrscsuffix}%
8868 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8869 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

8870 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8871 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8872 \renewcommand*\glsxtrfullformat}[2]{%
8873     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8874     \ifglsxtrininsertinside\else##2\fi
8875     \glsxtrfullsep{##1}%
8876     \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8877 }%
8878 \renewcommand*\glsxtrfullplformat}[2]{%
8879     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8880     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8881     \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8882 }%
8883 \renewcommand*\Glsxtrfullformat}[2]{%
8884     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8885     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8886     \glsxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8887 }%
8888 \renewcommand*\Glsxtrfullplformat}[2]{%
8889     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8890     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8891     \glsxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8892 }%
8893 }

```

g-short-sc-desc

```

8894 \newabbreviationstyle{long-short-sc-desc}%
8895 {%

```

Set accessibility attributes if enabled.

```

8896 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```
8897 \renewcommand*{\CustomAbbreviationFields}{%
8898   name={\glxtrlongshortdescname},
8899   sort={\glxtrlongshortdescsort},%
8900   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8901     \protect\glxtrfullsep{\the\glslabeltok}%
8902     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8903   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8904     \protect\glxtrfullsep{\the\glslabeltok}%
8905     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8906   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8907   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8908 }%
```

Unset the regular attribute if it has been set.

```
8909 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8910   \glshasattribute{\the\glslabeltok}{regular}%
8911   {%
8912     \glissetattribute{\the\glslabeltok}{regular}{false}%
8913   }%
8914   {}%
8915 }%
8916 }%
8917 {%
```

As long-short-sc style:

```
8918 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8919 }
```

short-sc-long Now the short (long) version

```
8920 \newabbreviationstyle{short-sc-long}%
8921 {%
```

Set accessibility attributes if enabled.

```
8922 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
8923 \renewcommand*{\CustomAbbreviationFields}{%
8924   name={\glxtrshortlongname},
8925   sort={\the\glsshorttok},
8926   description={\the\glslongtok},%
8927   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8928     \protect\glxtrfullsep{\the\glslabeltok}%
8929     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8930   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8931     \protect\glxtrfullsep{\the\glslabeltok}%
8932     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8933   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8934   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8935 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8936   \glshasattribute{\the\glslabelltok}{regular}}%
8937   {%
8938     \glissetattribute{\the\glslabelltok}{regular}{false}}%
8939   }%
8940   {}%
8941 }%
8942 }%
8943 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8944 \renewcommand*{\abbrvpluralsuffix}{\glsxtrscsuffix}%
8945 \renewcommand*{\glssabrvfont[1]{\glssabrvscfont{##1}}}%
8946 \renewcommand*{\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}}%
8947 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8948 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8949 \renewcommand*{\glsxtrfullformat}[2]{%
8950   \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8951   \ifglsxtrinsertinside\else##2\fi
8952   \glsxtrfullsep{##1}%
8953   \glsxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8954 }%
8955 \renewcommand*{\glsxtrfullplformat}[2]{%
8956   \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8957   \ifglsxtrinsertinside\else##2\fi
8958   \glsxtrfullsep{##1}%
8959   \glsxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8960 }%
8961 \renewcommand*{\Glsxtrfullformat}[2]{%
8962   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8963   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8964   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8965 }%
8966 \renewcommand*{\Glsxtrfullplformat}[2]{%
8967   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8968   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8969   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8970 }%
8971 }

```

rt-sc-long-desc As before but user provides description

```

8972 \newabbreviationstyle{short-sc-long-desc}%
8973 {%

```

Set accessibility attributes if enabled.

```

8974 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

8975 \renewcommand*{\CustomAbbreviationFields}{%

```

```

8976   name={\glxtrshortlongdescname},
8977   sort={\glxtrshortlongdescsort},
8978   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8979     \protect\glxtrfullsep{\the\glslabeltok}%
8980     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8981   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8982     \protect\glxtrfullsep{\the\glslabeltok}%
8983     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8984   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8985   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8986 }%

```

Unset the regular attribute if it has been set.

```

8987 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8988   \glshasattribute{\the\glslabeltok}{regular}%
8989   {%
8990     \glissetattribute{\the\glslabeltok}{regular}{false}%
8991   }%
8992   {}%
8993 }%
8994 }%
8995 {%

```

As short-sc-long style:

```

8996 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8997 }

```

short-sc

```

8998 \newabbreviationstyle{short-sc}%
8999 {%

```

Set accessibility attributes if enabled.

```

9000 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9001 \renewcommand*{\CustomAbbreviationFields}{%
9002   name={\glxtrshortnolongname},
9003   sort={\the\glsshorttok},
9004   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
9005   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
9006   text={\protect\glsabbrvscfont{\the\glsshorttok}},
9007   plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
9008   description={\the\glslongtok}}%
9009 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9010   \glissetattribute{\the\glslabeltok}{regular}{true}}%
9011 }%
9012 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9013 \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
9014 \renewcommand*{\glsabbrvfont[1]}{\glsabbrvscfont{##1}}%

```

```

9015 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
9016 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9017 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9018 \renewcommand*\glsxtrinlinefullformat}[2]{%
9019   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
9020   \ifglsxtrininsertinside##2\fi}%
9021   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9022   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9023 }%
9024 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9025   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
9026   \ifglsxtrininsertinside##2\fi}%
9027   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9028   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9029 }%

9030 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9031   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
9032   \ifglsxtrininsertinside##2\fi}%
9033   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9034   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9035 }%
9036 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9037   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}%
9038   \ifglsxtrininsertinside##2\fi}%
9039   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9040   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9041 }%

```

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.

```

9042 \renewcommand*\glsxtrfullformat}[2]{%
9043   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9044   \ifglsxtrininsertinside\else##2\fi
9045 }%
9046 \renewcommand*\glsxtrfullplformat}[2]{%
9047   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9048   \ifglsxtrininsertinside\else##2\fi
9049 }%
9050 \renewcommand*\Glsxtrfullformat}[2]{%
9051   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9052   \ifglsxtrininsertinside\else##2\fi
9053 }%
9054 \renewcommand*\Glsxtrfullplformat}[2]{%
9055   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9056   \ifglsxtrininsertinside\else##2\fi
9057 }%
9058 }

```

short-sc-nolong

```
9059 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
9060 \newabbreviationstyle{short-sc-desc}{%
```

```
9061 {%
```

Set accessibility attributes if enabled.

```
9062 \glxstrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
9063 \renewcommand*{\CustomAbbreviationFields}{%
```

```
9064 name={\glxstrshortdescname},
```

```
9065 sort={\the\glsshorttok},
```

```
9066 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
```

```
9067 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
```

```
9068 text={\protect\glssabbrvscfont{\the\glsshorttok}},
```

```
9069 plural={\protect\glssabbrvscfont{\the\glsshortpltok}}}%
```

```
9070 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
9071 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
```

```
9072 }%
```

```
9073 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
9074 \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
```

```
9075 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
```

```
9076 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
```

```
9077 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
```

```
9078 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
9079 \renewcommand*{\glxtrinlinefullformat}[2]{%
```

```
9080 \glsfirstabbrvscfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
```

```
9081 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
9082 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}}%
```

```
9083 }%
```

```
9084 \renewcommand*{\glxtrinlinefullplformat}[2]{%
```

```
9085 \glsfirstabbrvscfont{\glssaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
```

```
9086 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
9087 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}}%
```

```
9088 }%
```

```
9089 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
```

```
9090 \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
```

```
9091 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
9092 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}}%
```

```
9093 }%
```

```
9094 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
```

```
9095 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
```

```
9096 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
```

```
9097 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}}%
```

```
9098 }%
```

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.

```

9099 \renewcommand*{\glxtrfullformat}[2]{%
9100   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9101   \ifglxtrinsertinside\else##2\fi
9102 }%
9103 \renewcommand*{\glxtrfullplformat}[2]{%
9104   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9105   \ifglxtrinsertinside\else##2\fi
9106 }%
9107 \renewcommand*{\Glsxtrfullformat}[2]{%
9108   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9109   \ifglxtrinsertinside\else##2\fi
9110 }%
9111 \renewcommand*{\Glsxtrfullplformat}[2]{%
9112   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9113   \ifglxtrinsertinside\else##2\fi
9114 }%
9115 }

```

-sc-nolong-desc

```

9116 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

9117 \newabbreviationstyle{nolong-short-sc}%
9118 {%
9119   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
9120 }%
9121 {%
9122   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9123 \renewcommand*{\glxtrinlinefullformat}[2]{%
9124   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9125   \ifglxtrinsertinside##2\fi}%
9126   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9127   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9128 }%
9129 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9130   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9131   \ifglxtrinsertinside##2\fi}%
9132   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9133   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9134 }%
9135 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9136   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9137   \ifglxtrinsertinside##2\fi}%
9138   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9139   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%

```

```

9140 }%
9141 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9142   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9143     \ifglxtrinsertinside##2\fi}%
9144   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9145   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
9146 }%
9147 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`. No accessibility attributes needed here.

```

9148 \newabbreviationstyle{long-noshort-sc}%
9149 {%
9150   \renewcommand*{\CustomAbbreviationFields}{%
9151     name={\glxtrlongnoshortname},
9152     sort={\the\glsshorttok},
9153     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9154     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9155     text={\protect\glslongdefaultfont{\the\glslongtok}},
9156     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9157     description={\the\glslongtok}%
9158   }%
9159   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9160     \glsetattribute{\the\glslabeltok}{regular}{true}}%
9161 }%
9162 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9163   \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
9164   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
9165   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
9166   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9167   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9168   \renewcommand*{\glxtrsubsequentfmt}[2]{%
9169     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9170     \ifglxtrinsertinside \else##2\fi
9171   }%
9172   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9173     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9174     \ifglxtrinsertinside \else##2\fi
9175   }%
9176   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9177     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9178     \ifglxtrinsertinside \else##2\fi
9179   }%
9180   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9181     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9182     \ifglxtrinsertinside \else##2\fi

```


9183 }%

The inline full form displays the long format followed by the short form in parentheses.

```
9184 \renewcommand*{\glxstrinlinefullformat}[2]{%
9185   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9186   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9187   \glxstrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9188 }%
9189 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9190   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9191   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9192   \glxstrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9193 }%
9194 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9195   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9196   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9197   \glxstrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9198 }%
9199 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9200   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9201   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9202   \glxstrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9203 }%
```

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.

```
9204 \renewcommand*{\glxstrfullformat}[2]{%
9205   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9206   \ifglxstrinsertinside\else##2\fi
9207 }%
9208 \renewcommand*{\glxstrfullplformat}[2]{%
9209   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9210   \ifglxstrinsertinside\else##2\fi
9211 }%
9212 \renewcommand*{\Glsxtrfullformat}[2]{%
9213   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9214   \ifglxstrinsertinside\else##2\fi
9215 }%
9216 \renewcommand*{\Glsxtrfullplformat}[2]{%
9217   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9218   \ifglxstrinsertinside\else##2\fi
9219 }%
9220 }
```

long-sc Backward compatibility:

```
9221 \@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.

```
9222 \newabbreviationstyle{long-noshort-sc-desc}%
```

```

9223 {%
9224   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9225 }%
9226 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9227   \renewcommand*\abbrvpluralsuffix{\glstrscsuffix}%
9228   \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
9229   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
9230   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9231   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9232   \renewcommand*\glstrsubsequentfmt}[2]{%
9233     \glslongdefaultfont{\glsaccesslong{##1}\ifglstrinsertinside ##2\fi}%
9234     \ifglstrinsertinside \else##2\fi
9235   }%
9236   \renewcommand*\glstrsubsequentplfmt}[2]{%
9237     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglstrinsertinside ##2\fi}%
9238     \ifglstrinsertinside \else##2\fi
9239   }%
9240   \renewcommand*\Glsxtrsubsequentfmt}[2]{%
9241     \glslongdefaultfont{\Glsaccesslong{##1}\ifglstrinsertinside ##2\fi}%
9242     \ifglstrinsertinside \else##2\fi
9243   }%
9244   \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
9245     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglstrinsertinside ##2\fi}%
9246     \ifglstrinsertinside \else##2\fi
9247   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9248   \renewcommand*\glstrinlinefullformat}[2]{%
9249     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
9250     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9251     \glstrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9252   }%
9253   \renewcommand*\glstrinlinefullplformat}[2]{%
9254     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9255     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9256     \glstrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9257   }%
9258   \renewcommand*\Glsxtrinlinefullformat}[2]{%
9259     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglstrinsertinside##2\fi}%
9260     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9261     \glstrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9262   }%
9263   \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9264     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9265     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9266     \glstrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9267   }%

```

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.

```

9268 \renewcommand*{\glxtrfullformat}[2]{%
9269   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9270   \ifglxtrinsertinside\else##2\fi
9271 }%
9272 \renewcommand*{\glxtrfullplformat}[2]{%
9273   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9274   \ifglxtrinsertinside\else##2\fi
9275 }%
9276 \renewcommand*{\Glsxtrfullformat}[2]{%
9277   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9278   \ifglxtrinsertinside\else##2\fi
9279 }%
9280 \renewcommand*{\Glsxtrfullplformat}[2]{%
9281   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9282   \ifglxtrinsertinside\else##2\fi
9283 }%
9284 }

```

long-desc-sc Backward compatibility:

```

9285 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

9286 \newabbreviationstyle{short-sc-footnote}%
9287 {%

```

Set accessibility attributes if enabled.

```

9288 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9289 \renewcommand*{\CustomAbbreviationFields}{%
9290   name={\glxtrfootnotename},
9291   sort={\the\glsshorttok},
9292   description={\the\glslongtok},%
9293   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
9294     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9295     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9296   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
9297     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9298     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9299   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
9300   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.

```

9301 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9302   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9303   \glsattribute{\the\glslabeltok}{regular}%
9304 }%

```

```

9305     \glsetattribute{\the\glslabeltok}{regular}{false}%
9306   }%
9307   {}%
9308 }%
9309}%
9310{%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9311 \renewcommand*\abbrvpluralsuffix{\glxtrscsuffix}%
9312 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
9313 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
9314 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9315 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9316 \renewcommand*\glsxtrfullformat}[2]{%
9317   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9318   \ifglsxtrininsertinside\else##2\fi
9319   \protect\glsxtrabbrvfootnote{##1}%
9320   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9321 }%
9322 \renewcommand*\glsxtrfullplformat}[2]{%
9323   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9324   \ifglsxtrininsertinside\else##2\fi
9325   \protect\glsxtrabbrvfootnote{##1}%
9326   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9327 }%
9328 \renewcommand*\Glsxtrfullformat}[2]{%
9329   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9330   \ifglsxtrininsertinside\else##2\fi
9331   \protect\glsxtrabbrvfootnote{##1}%
9332   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9333 }%
9334 \renewcommand*\Glsxtrfullplformat}[2]{%
9335   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9336   \ifglsxtrininsertinside\else##2\fi
9337   \protect\glsxtrabbrvfootnote{##1}%
9338   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9339 }%

```

The first use full form and the inline full form use the short (long) style.

```

9340 \renewcommand*\glsxtrininlinefullformat}[2]{%
9341   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9342   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9343   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9344 }%
9345 \renewcommand*\glsxtrininlinefullplformat}[2]{%
9346   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9347   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9348   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9349 }%

```

```

9350 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9351   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9352   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9353   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9354 }%
9355 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9356   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9357   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9358   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9359 }%
9360 }

```

footnote-sc Backward compatibility:

```

9361 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

c-footnote-desc Like short-sc-footnote but with user supplied description.

```

9362 \newabbreviationstyle{short-sc-footnote-desc}%
9363 {}%

```

Set accessibility attributes if enabled.

```

9364 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

9365 \renewcommand*{\CustomAbbreviationFields}{%
9366   name={\glxtrfootnotedesname},
9367   sort={\glxtrfootnotedesort},
9368   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
9369     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9370     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9371   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
9372     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9373     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9374   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
9375   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.

```

9376 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9377   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9378   \glshasattribute{\the\glslabeltok}{regular}%
9379   {%
9380     \glsssetattribute{\the\glslabeltok}{regular}{false}%
9381   }%
9382   {}%
9383 }%
9384 }%
9385 {}%
9386 \GlsXtrUseAbbrStyleFmts{short-sc-footnote}%
9387 }

```

sc-postfootnote

```
9388 \newabbreviationstyle{short-sc-postfootnote}%  
9389 {%
```

Set accessibility attributes if enabled.

```
9390 \glxstrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
9391 \renewcommand*{\CustomAbbreviationFields}{%  
9392   name={\glxstrfootnotename},  
9393   sort={\the\glsshorttok},  
9394   description={\the\glslongtok},%  
9395   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%  
9396   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%  
9397   text={\protect\glssabbrvscfont{\the\glsshorttok}},%  
9398   plural={\protect\glssabbrvscfont{\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.

```
9399 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
9400   \csdef{glxstrpostlink\glscategorylabel}{%  
9401     \glxstrifwasfirstuse  
9402     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
9403       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%  
9404       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%  
9405       }%  
9406       {}%  
9407       }%  
9408       \glshasattribute{\the\glslabeltok}{regular}%  
9409       {%  
9410       \glissetattribute{\the\glslabeltok}{regular}{false}%  
9411       }%  
9412       {}%  
9413   }%
```

The footnote needs to be suppressed in the inline form, so \glxstrfull must set the first use switch off.

```
9414 \renewcommand*{\glxstrsetupfulldefs}{%  
9415   \let\glxstrifwasfirstuse\@secondoftwo  
9416   }%  
9417 }%  
9418 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
9419 \renewcommand*{\abbrvpluralsuffix}{\glxstrscsuffix}%  
9420 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%  
9421 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%  
9422 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%  
9423 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```

9424 \renewcommand*{\glxtrfullformat}[2]{%
9425   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9426   \ifglxtrinsertinside\else##2\fi
9427 }%
9428 \renewcommand*{\glxtrfullplformat}[2]{%
9429   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9430   \ifglxtrinsertinside\else##2\fi
9431 }%
9432 \renewcommand*{\Glsxtrfullformat}[2]{%
9433   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9434   \ifglxtrinsertinside\else##2\fi
9435 }%
9436 \renewcommand*{\Glsxtrfullplformat}[2]{%
9437   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9438   \ifglxtrinsertinside\else##2\fi
9439 }%

```

The first use full form and the inline full form use the short (long) style.

```

9440 \renewcommand*{\glxtrinlinefullformat}[2]{%
9441   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9442   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9443   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9444 }%
9445 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9446   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9447   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9448   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9449 }%
9450 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9451   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9452   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9453   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9454 }%
9455 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9456   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9457   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9458   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9459 }%
9460 }

```

postfootnote-sc Backward compatibility:

```

9461 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

stfootnote-desc Like short-sc-footnote but with user supplied description.

```

9462 \newabbreviationstyle{short-sc-postfootnote-desc}%
9463 {}

```

Set accessibility attributes if enabled.

```

9464 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

9465 \renewcommand*{\CustomAbbreviationFields}{%
9466   name={\glxtrfootnotedescname},
9467   sort={\glxtrfootnotedescsort},
9468   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
9469   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
9470   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
9471   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.

```

9472 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9473   \csdef{glxtrpostlink\glscategorylabel}{%
9474     \glxtrifwasfirstuse
9475     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9476       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9477       {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}}%
9478     }%
9479   }%
9480 }%
9481 \glshasattribute{\the\glslabeltok}{regular}%
9482 {%
9483   \glissetattribute{\the\glslabeltok}{regular}{false}%
9484   }%
9485   {}%
9486 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9487 \renewcommand*{\glxtrsetupfulldefs}{%
9488   \let\glxtrifwasfirstuse\@secondoftwo
9489 }%
9490 }%
9491 {%
9492   \GlsXtrUseAbbrStyleFmts{short-sc-postfootnote}%
9493 }

```

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.

```

9494 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

```

`\glsabbrvsmfont` Added for consistent naming.

```

9495 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

```


sxtrfirstsmfont Maintained for backward compatibility.

```
9496 \newcommand*{\glxsxtrfirstsmfont}[1]{\glssabrvsmfont{#1}}
```

irstabrvsmfont Added for consistent naming.

```
9497 \newcommand*{\glssfirstabrvsmfont}{\glxsxtrfirstsmfont}
```

and for the default short form suffix:

```
\glxsxtrsmsuffix
9498 \newcommand*{\glxsxtrsmsuffix}{\glsxtrabrvpluralsuffix}
```

long-short-sm

```
9499 \newabbreviationstyle{long-short-sm}%
9500 {%
    Set accessibility attributes if enabled.
9501   \glxsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
    Setup the default fields.
9502   \renewcommand*{\CustomAbbreviationFields}{%
9503     name={\glsxtrlongshortname},
9504     sort={\the\glssshorttok},
9505     first={\protect\glssfirstlongdefaultfont{\the\glslongtok}%
9506       \protect\glxsxtrfullsep{\the\glslabeltok}%
9507       \glxsxtrparen{\protect\glssfirstabrvsmfont{\the\glssshorttok}}},%
9508     firstplural={\protect\glssfirstlongdefaultfont{\the\glslongpltok}%
9509       \protect\glxsxtrfullsep{\the\glslabeltok}%
9510       \glxsxtrparen{\protect\glssfirstabrvsmfont{\the\glssshortpltok}}},%
9511     text={\protect\glssabrvsmfont{\the\glssshorttok}},%
9512     plural={\protect\glssabrvsmfont{\the\glssshortpltok}},%
9513     description={\the\glslongtok}}%
9514   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9515     \glshasattribute{\the\glslabeltok}{regular}%
9516     {%
9517       \glsssetattribute{\the\glslabeltok}{regular}{false}%
9518     }%
9519   }%
9520 }%
9521 }%
9522 {%
9523   \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
9524   \renewcommand*\glssfirstabrvfont[1]{\glssfirstabrvsmfont{##1}}%
9525   \renewcommand*{\abrvpluralsuffix}{\glxsxtrsmsuffix}%
    Use the default long fonts.
9526   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9527   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline form are the same for this style.

```

9528 \renewcommand*{\glxtrfullformat}[2]{%
9529   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9530   \ifglxtrinsertinside\else##2\fi
9531   \glxtrfullsep{##1}%
9532   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9533 }%
9534 \renewcommand*{\glxtrfullplformat}[2]{%
9535   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9536   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9537   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9538 }%
9539 \renewcommand*{\Glsxtrfullformat}[2]{%
9540   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9541   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9542   \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
9543 }%
9544 \renewcommand*{\Glsxtrfullplformat}[2]{%
9545   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9546   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9547   \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
9548 }%
9549 }

```

g-short-sm-desc

```

9550 \newabbreviationstyle{long-short-sm-desc}%
9551 {%

```

Set accessibility attributes if enabled.

```

9552 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

9553 \renewcommand*{\CustomAbbreviationFields}{%
9554   name={\glxtrlongshortdescname},
9555   sort={\glxtrlongshortdescsort},%
9556   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9557     \protect\glxtrfullsep{\the\glslabeltok}%
9558     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
9559   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9560     \protect\glxtrfullsep{\the\glslabeltok}%
9561     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
9562   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9563   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
9564 }%

```

Unset the regular attribute if it has been set.

```

9565 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9566   \glshasattribute{\the\glslabeltok}{regular}%
9567   {%
9568     \glissetattribute{\the\glslabeltok}{regular}{false}%

```

```

9569 }%
9570 {}%
9571 }%
9572 }%
9573 {%

```

As long-short-sm style:

```

9574 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
9575 }

```

short-sm-long Now the short (long) version

```

9576 \newabbreviationstyle{short-sm-long}%
9577 {%

```

Set accessibility attributes if enabled.

```

9578 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

9579 \renewcommand*{\CustomAbbreviationFields}{%
9580   name={\glxtrshortlongname},
9581   sort={\the\glsshorttok},
9582   description={\the\glslongtok},%
9583   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9584     \protect\glxtrfullsep{\the\glslabeltok}%
9585     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9586   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9587     \protect\glxtrfullsep{\the\glslabeltok}%
9588     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9589   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9590   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9591 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9592   \glshasattribute{\the\glslabeltok}{regular}%
9593   {%
9594     \glissetattribute{\the\glslabeltok}{regular}{false}%
9595   }%
9596   {}%
9597 }%
9598 }%
9599 {%

```

```

9600 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9601 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9602 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
9603 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9604 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9605 \renewcommand*{\glxtrfullformat}[2]{%
9606   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%

```

```

9607 \ifglxtrinsertinside\else##2\fi
9608 \glxtrfullsep{##1}%
9609 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9610 }%
9611 \renewcommand*{\glxtrfullplformat}[2]{%
9612 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9613 \ifglxtrinsertinside\else##2\fi
9614 \glxtrfullsep{##1}%
9615 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9616 }%
9617 \renewcommand*{\Glsxtrfullformat}[2]{%
9618 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9619 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9620 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9621 }%
9622 \renewcommand*{\Glsxtrfullplformat}[2]{%
9623 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9624 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9625 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9626 }%
9627 }

```

rt-sm-long-desc As before but user provides description

```

9628 \newabbreviationstyle{short-sm-long-desc}%
9629 {%

```

Set accessibility attributes if enabled.

```
9630 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```

9631 \renewcommand*{\CustomAbbreviationFields}{%
9632 name={\glxtrshortlongdescname},
9633 sort={\glxtrshortlongdescsort},
9634 first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}}%
9635 \protect\glxtrfullsep{\the\glslabeltok}}%
9636 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9637 firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}}%
9638 \protect\glxtrfullsep{\the\glslabeltok}}%
9639 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9640 text={\protect\glsabbrvsmfont{\the\glssshorttok}},%
9641 plural={\protect\glsabbrvsmfont{\the\glssshortpltok}}}%
9642 }%

```

Unset the regular attribute if it has been set.

```

9643 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9644 \glshasattribute{\the\glslabeltok}{regular}}%
9645 {%
9646 \glissetattribute{\the\glslabeltok}{regular}{false}}%
9647 }%
9648 {}%

```

```

9649 }%
9650 }%
9651 {%

```

As short-sm-long style:

```

9652 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
9653 }

```

short-sm

```

9654 \newabbreviationstyle{short-sm}%
9655 {%

```

Set accessibility attributes if enabled.

```

9656 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

9657 \renewcommand*\CustomAbbreviationFields{%
9658   name={\glxtrshortnolongname},
9659   sort={\the\glsshorttok},
9660   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
9661   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
9662   text={\protect\glsabbrvsmfont{\the\glsshorttok}},
9663   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
9664   description={\the\glslongtok}}%
9665 \renewcommand*\GlsXtrPostNewAbbreviation{%
9666   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9667 }%
9668 {%

9669 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9670 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9671 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
9672 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9673 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9674 \renewcommand*\glxtrinlinefullformat[2]{%
9675   \protect\glsfirstabbrvsmfont{\glsaccesssshort{##1}}%
9676   \ifglxtrininsertinside##2\fi}%
9677 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9678 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9679 }%
9680 \renewcommand*\glxtrinlinefullplformat[2]{%
9681   \protect\glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}%
9682   \ifglxtrininsertinside##2\fi}%
9683 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9684 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9685 }%

9686 \renewcommand*\Glsxtrinlinefullformat[2]{%
9687   \protect\glsfirstabbrvsmfont{\Glsaccesssshort{##1}}%

```

```

9688     \ifglxtrinsertinside##2\fi}%
9689     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9690     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9691 }%
9692 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9693     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
9694         \ifglxtrinsertinside##2\fi}%
9695     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9696     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9697 }%

```

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.

```

9698 \renewcommand*{\glxtrfullformat}[2]{%
9699     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9700     \ifglxtrinsertinside\else##2\fi
9701 }%
9702 \renewcommand*{\glxtrfullplformat}[2]{%
9703     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9704     \ifglxtrinsertinside\else##2\fi
9705 }%
9706 \renewcommand*{\Glsxtrfullformat}[2]{%
9707     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9708     \ifglxtrinsertinside\else##2\fi
9709 }%
9710 \renewcommand*{\Glsxtrfullplformat}[2]{%
9711     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9712     \ifglxtrinsertinside\else##2\fi
9713 }%
9714 }

```

short-sm-nolong

```

9715 \letabbreviationstyle{short-sm-nolong}{short-sm}

```

short-sm-desc

```

9716 \newabbreviationstyle{short-sm-desc}{%
9717 {%

```

Set accessibility attributes if enabled.

```

9718 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9719 \renewcommand*{\CustomAbbreviationFields}{%
9720     name={\glxtrshortdescname},
9721     sort={\the\glssshorttok},
9722     first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}},
9723     firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}},
9724     text={\protect\glssabbrvsmfont{\the\glssshorttok}},
9725     plural={\protect\glssabbrvsmfont{\the\glssshortpltok}}}%
9726 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

9727 \glsssetattribute{\the\glslabelltok}{regular}{true}}%
9728 }%
9729 {%

```

```

9730 \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
9731 \renewcommand*\glssfirstabrvfont[1]{\glssfirstabrvsmfont{##1}}%
9732 \renewcommand*\glssabrvpluralsuffix{\glssxtrmsuffix}%
9733 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
9734 \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9735 \renewcommand*\glssxtrinlinefullformat}[2]{%
9736 \glssfirstabrvsmfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9737 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9738 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9739 }%
9740 \renewcommand*\glssxtrinlinefullplformat}[2]{%
9741 \glssfirstabrvsmfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9742 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9743 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9744 }%
9745 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9746 \glssfirstabrvsmfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9747 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9748 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9749 }%
9750 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9751 \glssfirstabrvsmfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9752 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9753 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9754 }%

```

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.

```

9755 \renewcommand*\glssxtrfullformat}[2]{%
9756 \glssfirstabrvsmfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9757 \ifglssxtrininsertinside\else##2\fi
9758 }%
9759 \renewcommand*\glssxtrfullplformat}[2]{%
9760 \glssfirstabrvsmfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9761 \ifglssxtrininsertinside\else##2\fi
9762 }%
9763 \renewcommand*\Glsxtrfullformat}[2]{%
9764 \glssfirstabrvsmfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9765 \ifglssxtrininsertinside\else##2\fi
9766 }%
9767 \renewcommand*\Glsxtrfullplformat}[2]{%
9768 \glssfirstabrvsmfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9769 \ifglssxtrininsertinside\else##2\fi
9770 }%

```

9771 }

-sm-nolong-desc

9772 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

nolong-short-sm

```
9773 \newabbreviationstyle{nolong-short-sm}%
9774 {%
9775   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9776 }%
9777 {%
9778   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
9779   \renewcommand*{\glxtrinlinefullformat}[2]{%
9780     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9781       \ifglxtrininsertinside##2\fi}%
9782     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9783     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9784   }%
9785   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9786     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9787       \ifglxtrininsertinside##2\fi}%
9788     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9789     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9790   }%
9791   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9792     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9793       \ifglxtrininsertinside##2\fi}%
9794     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9795     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9796   }%
9797   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9798     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9799       \ifglxtrininsertinside##2\fi}%
9800     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9801     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9802   }%
9803 }
```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
9804 \newabbreviationstyle{long-noshort-sm}%
9805 {%
```

Set accessibility attributes if enabled.

```
9806   \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
9807   \renewcommand*{\CustomAbbreviationFields}{%
```



```

9808   name={\glxtrlongnosshortname},
9809   sort={\the\glsshorttok},
9810   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9811   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9812   text={\protect\glslongdefaultfont{\the\glslongtok}},
9813   plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9814   description={\the\glslongtok}%
9815 }%
9816 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9817   \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9818 }%
9819 {%

9820 \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
9821 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9822 \renewcommand*{\abbrvpluralsuffix}{\glxtrsmsuffix}%
9823 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9824 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9825 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9826   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9827   \ifglxtrininsertinside \else##2\fi
9828 }%
9829 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9830   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9831   \ifglxtrininsertinside \else##2\fi
9832 }%
9833 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9834   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9835   \ifglxtrininsertinside \else##2\fi
9836 }%
9837 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9838   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9839   \ifglxtrininsertinside \else##2\fi
9840 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9841 \renewcommand*{\glxtrininlinefullformat}[2]{%
9842   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9843   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9844   \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
9845 }%
9846 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9847   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9848   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9849   \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
9850 }%
9851 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9852   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%

```

```

9853     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9854     \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
9855 }%
9856 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9857     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9858     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9859     \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
9860 }%

```

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.

```

9861 \renewcommand*{\glxtrfullformat}[2]{%
9862     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9863     \ifglxtrinsertinside\else##2\fi
9864 }%
9865 \renewcommand*{\glxtrfullplformat}[2]{%
9866     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9867     \ifglxtrinsertinside\else##2\fi
9868 }%
9869 \renewcommand*{\Glsxtrfullformat}[2]{%
9870     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9871     \ifglxtrinsertinside\else##2\fi
9872 }%
9873 \renewcommand*{\Glsxtrfullplformat}[2]{%
9874     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9875     \ifglxtrinsertinside\else##2\fi
9876 }%
9877 }

```

long-sm Backward compatibility:

```

9878 \@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`.

```

9879 \newabbreviationstyle{long-noshort-sm-desc}%
9880 {%
9881     \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9882 }%
9883 {%
9884     \renewcommand*{\glssabbrvfont}[1]{\glssabbrvsmfont{##1}}%
9885     \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9886     \renewcommand*{\abbrvpluralsuffix}{\glsxtrsmsuffix}%
9887     \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9888     \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9889 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9890     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9891     \ifglxtrinsertinside \else##2\fi

```

```

9892 }%
9893 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9894   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9895   \ifglxtrinsertinside \else##2\fi
9896 }%
9897 \renewcommand*{\Glsxtrsubsequentffmt}[2]{%
9898   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9899   \ifglxtrinsertinside \else##2\fi
9900 }%
9901 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9902   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9903   \ifglxtrinsertinside \else##2\fi
9904 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9905 \renewcommand*{\glxtrinlinefullformat}[2]{%
9906   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9907   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9908   \glxtrparen{\protect\glslfirstabbrvsmfont{\glssaccessshort{##1}}}%
9909 }%
9910 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9911   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9912   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9913   \glxtrparen{\protect\glslfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
9914 }%
9915 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9916   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9917   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9918   \glxtrparen{\protect\glslfirstabbrvsmfont{\glssaccessshort{##1}}}%
9919 }%
9920 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9921   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9922   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9923   \glxtrparen{\protect\glslfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
9924 }%

```

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.

```

9925 \renewcommand*{\glxtrfullformat}[2]{%
9926   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9927   \ifglxtrinsertinside\else##2\fi
9928 }%
9929 \renewcommand*{\glxtrfullplformat}[2]{%
9930   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9931   \ifglxtrinsertinside\else##2\fi
9932 }%
9933 \renewcommand*{\Glsxtrfullformat}[2]{%
9934   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9935   \ifglxtrinsertinside\else##2\fi
9936 }%

```

```

9937 \renewcommand*{\Glsxtrfullplformat}[2]{%
9938   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9939   \ifglxtrinsertinside\else##2\fi
9940 }%
9941 }

```

long-desc-sm Backward compatibility:

```

9942 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9943 \newabbreviationstyle{short-sm-footnote}%
9944 {%

```

Set accessibility attributes if enabled.

```

9945 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9946 \renewcommand*{\CustomAbbreviationFields}{%
9947   name={\glxtrfootnotename},
9948   sort={\the\glsshorttok},
9949   description={\the\glslongtok},%
9950   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9951     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9952     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9953   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9954     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9955     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9956   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9957   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.

```

9958 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9959   \glsetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9960   \glshasattribute{\the\glslabeltok}{regular}%
9961   {%
9962     \glsetattribute{\the\glslabeltok}{regular}{false}%
9963   }%
9964   {}%
9965 }%
9966 }%
9967 {%

```

```

9968 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9969 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9970 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
9971 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9972 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9973 \renewcommand*{\glxtrfullformat}[2]{%
9974   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9975   \ifglxtrininsertinside\else##2\fi
9976   \protect\glxtrabbrvfootnote{##1}%
9977   {\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9978 }%
9979 \renewcommand*{\glxtrfullplformat}[2]{%
9980   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9981   \ifglxtrininsertinside\else##2\fi
9982   \protect\glxtrabbrvfootnote{##1}%
9983   {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9984 }%
9985 \renewcommand*{\Glsxtrfullformat}[2]{%
9986   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9987   \ifglxtrininsertinside\else##2\fi
9988   \protect\glxtrabbrvfootnote{##1}%
9989   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9990 }%
9991 \renewcommand*{\Glsxtrfullplformat}[2]{%
9992   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9993   \ifglxtrininsertinside\else##2\fi
9994   \protect\glxtrabbrvfootnote{##1}%
9995   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9996 }%

```

The first use full form and the inline full form use the short (long) style.

```

9997 \renewcommand*{\glxtrininlinefullformat}[2]{%
9998   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9999   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10000   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10001 }%
10002 \renewcommand*{\glxtrininlinefullplformat}[2]{%
10003   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10004   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10005   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10006 }%
10007 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
10008   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10009   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10010   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10011 }%
10012 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
10013   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10014   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10015   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10016 }%
10017 }

```

footnote-sm Backward compatibility:

```

10018 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

m-footnote-desc Like short-footnote but with user supplied description.

```
10019 \newabbreviationstyle{short-sm-footnote-desc}%  
10020 {%
```

Set accessibility attributes if enabled.

```
10021 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
10022 \renewcommand*{\CustomAbbreviationFields}{%  
10023   name={\glxtrfootnotedesname},  
10024   sort={\glxtrfootnotedesort},  
10025   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%  
10026     \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
10027     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%  
10028   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%  
10029     \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
10030     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%  
10031   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%  
10032   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.

```
10033 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
10034   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%  
10035   \glshasattribute{\the\glslabeltok}{regular}%  
10036   {%  
10037     \glsssetattribute{\the\glslabeltok}{regular}{false}%  
10038   }%  
10039   {}%  
10040 }%  
10041 }%  
10042 {%  
10043   \GlsXtrUseAbbrStyleFmts{short-sm-footnote}%  
10044 }
```

sm-postfootnote

```
10045 \newabbreviationstyle{short-sm-postfootnote}%  
10046 {%
```

Set accessibility attributes if enabled.

```
10047 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
10048 \renewcommand*{\CustomAbbreviationFields}{%  
10049   name={\glxtrfootnotename},  
10050   sort={\the\glsshorttok},  
10051   description={\the\glslongtok},%  
10052   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%  
10053   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%  
10054   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%  
10055   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.

```
10056 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10057 \csdef{glxstrpostlink\glscategorylabel}{%
10058 \glxtrifwasfirstuse
10059 {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
10060 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10061 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
10062 }%
10063 {}%
10064 }%
10065 \glshasattribute{\the\glslabeltok}{regular}%
10066 {%
10067 \glissetattribute{\the\glslabeltok}{regular}{false}%
10068 }%
10069 {}%
10070 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
10071 \renewcommand*{\glxtrsetupfulldefs}{%
10072 \let\glxtrifwasfirstuse\@secondoftwo
10073 }%
10074 }%
10075 {%
```

```
10076 \renewcommand*\glssabbrvfont[1]{\glssabbrvsmfont{##1}}%
10077 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
10078 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
10079 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
10080 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
10081 \renewcommand*{\glxtrfullformat}[2]{%
10082 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10083 \ifglxtrininsertinside\else##2\fi
10084 }%
10085 \renewcommand*{\glxtrfullplformat}[2]{%
10086 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10087 \ifglxtrininsertinside\else##2\fi
10088 }%
10089 \renewcommand*{\Glsxtrfullformat}[2]{%
10090 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10091 \ifglxtrininsertinside\else##2\fi
10092 }%
10093 \renewcommand*{\Glsxtrfullplformat}[2]{%
10094 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10095 }
```

```

10095 \ifglxtrinsertinside\else##2\fi
10096 }%

```

The first use full form and the inline full form use the short (long) style.

```

10097 \renewcommand*{\glxtrinlinefullformat}[2]{%
10098 \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10099 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10100 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
10101 }%
10102 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10103 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10104 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10105 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10106 }%
10107 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10108 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10109 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10110 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10111 }%
10112 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10113 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10114 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10115 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
10116 }%
10117 }

```

postfootnote-sm Backward compatibility:

```

10118 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

stfootnote-desc Like short-sm-postfootnote but with user supplied description.

```

10119 \newabbreviationstyle{short-sm-postfootnote-desc}%
10120 {%

```

Set accessibility attributes if enabled.

```

10121 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

10122 \renewcommand*{\CustomAbbreviationFields}{%
10123 name={\glxtrfootnotedesname},
10124 sort={\glxtrfootnotedesort},
10125 first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}},%
10126 firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}},%
10127 text={\protect\glssabbrvsmfont{\the\glssshorttok}},%
10128 plural={\protect\glssabbrvsmfont{\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.

```

10129 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10130 \csdef{glxtrpostlink\glscategorylabel}{%
10131 \glxtrifwasfirstuse
10132 {%

```


Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

10133      \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10134      {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
10135      }%
10136      {}%
10137      }%
10138      \glshasattribute{\the\glslabeltok}{regular}%
10139      {%
10140      \glissetattribute{\the\glslabeltok}{regular}{false}%
10141      }%
10142      {}%
10143      }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

10144  \renewcommand*{\glxtrsetupfulldefs}{%
10145  \let\glxtrifwasfirstuse\@secondoftwo
10146  }%
10147  }%
10148  {%
10149  \GlsXtrUseAbbrStyleFmts{short-sm-postfootnote}%
10150  }

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

10151 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

```

`\glfirstabbrvemfont`

```

10152 \newcommand*{\glfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

```

The default short form suffix:

`\glxxtremsuffix`

```

10153 \newcommand*{\glxxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`\glfirstlongemfont` Only used by the “long-em” styles.

```

10154 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%

```

`\glslongemfont` Only used by the “long-em” styles.

```

10155 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

```

`\long-short-em` The long form is just set in the default long font.

```

10156 \newabbreviationstyle{long-short-em}%
10157 {}%

```

Set accessibility attributes if enabled.

```
10158 \glxstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
10159 \renewcommand*{\CustomAbbreviationFields}{%
10160   name={\glxstrlongshortname},
10161   sort={\the\glsshorttok},
10162   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
10163     \protect\glxstrfullsep{\the\glslabeltok}%
10164     \glxstrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10165   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
10166     \protect\glxstrfullsep{\the\glslabeltok}%
10167     \glxstrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10168   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10169   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
10170   description={\the\glslongtok}}%
10171 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10172   \glshalattribute{\the\glslabeltok}{regular}%
10173   {%
10174     \glissetattribute{\the\glslabeltok}{regular}{false}%
10175   }%
10176   {}%
10177 }%
10178 }%
10179 {%
```

```
10180 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
10181 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
10182 \renewcommand*\abbrvpluralsuffix{\glxstremsuffix}%
```

Use the default long fonts.

```
10183 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
10184 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
10185 \renewcommand*\glxstrfullformat[2]{%
10186   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10187   \ifglxtrinsertinside\else##2\fi
10188   \glxstrfullsep{##1}%
10189   \glxstrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10190 }%
10191 \renewcommand*\glxstrfullplformat[2]{%
10192   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10193   \ifglxtrinsertinside\else##2\fi\glxstrfullsep{##1}%
10194   \glxstrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10195 }%
10196 \renewcommand*\Glsxtrfullformat[2]{%
10197   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10198   \ifglxtrinsertinside\else##2\fi\glxstrfullsep{##1}%
10199   \glxstrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%

```

```

10200 }%
10201 \renewcommand*{\Glsxtrfullplformat}[2]{%
10202   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10203   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10204   \glxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10205 }%
10206 }

```

g-short-em-desc

```

10207 \newabbreviationstyle{long-short-em-desc}%
10208 {%

```

Set accessibility attributes if enabled.

```

10209 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

10210 \renewcommand*{\CustomAbbreviationFields}{%
10211   name={\glxtrlongshortdescname},
10212   sort={\glxtrlongshortdescsort},%
10213   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
10214     \protect\glxtrfullsep{\the\glslabeltok}%
10215     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10216   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
10217     \protect\glxtrfullsep{\the\glslabeltok}%
10218     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10219   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10220   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10221 }%

```

Unset the regular attribute if it has been set.

```

10222 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10223   \glshasattribute{\the\glslabeltok}{regular}%
10224   {%
10225     \glissetattribute{\the\glslabeltok}{regular}{false}%
10226   }%
10227   {}%
10228 }%
10229 }%
10230 {%

```

As long-short-em style:

```

10231 \GlsXtrUseAbbrStyleFmts{long-short-em}%
10232 }

```

ong-em-short-em

```

10233 \newabbreviationstyle{long-em-short-em}%
10234 {%

```

Set accessibility attributes if enabled.

```

10235 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields. `\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```

10236 \renewcommand*{\CustomAbbreviationFields}{%
10237     name={\glsxtrlongshortname},
10238     sort={\the\glsshorttok},
10239     first={\protect\glsfirstlongemfont{\the\glslongtok}%
10240         \protect\glsxtrfullsep{\the\glslabeltok}%
10241         \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10242     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
10243         \protect\glsxtrfullsep{\the\glslabeltok}%
10244         \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

10245     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10246     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
10247     description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10248 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10249     \glshasattribute{\the\glslabeltok}{regular}%
10250     {%
10251         \glissetattribute{\the\glslabeltok}{regular}{false}%
10252     }%
10253     {}%
10254 }%
10255 }%
10256 {%

10257 \renewcommand*{\abbrvpluralsuffix}{\glsxtremsuffix}%
10258 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10259 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10260 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10261 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10262 \renewcommand*{\glsxtrfullformat}[2]{%
10263     \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10264     \ifglsxtrininsertinside\else##2\fi
10265     \glsxtrfullsep{##1}%
10266     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10267 }%
10268 \renewcommand*{\glsxtrfullplformat}[2]{%
10269     \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10270     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10271     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10272 }%
10273 \renewcommand*{\Glsxtrfullformat}[2]{%
10274     \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10275     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10276     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10277 }%

```

```

10278 \renewcommand*{\Glsxtrfullplformat}[2]{%
10279   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10280   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10281   \glxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10282 }%
10283 }

```

m-short-em-desc

```

10284 \newabbreviationstyle{long-em-short-em-desc}%
10285 {%
    Set accessibility attributes if enabled.
10286   \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

10287 \renewcommand*{\CustomAbbreviationFields}{%
10288   name={\glxtrlongshortdescname},
10289   sort={\glxtrlongshortdescsort},%
10290   first={\protect\glsfirstlongemfont{\the\glslongtok}%
10291     \protect\glxtrfullsep{\the\glslabeltok}%
10292     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10293   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
10294     \protect\glxtrfullsep{\the\glslabeltok}%
10295     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10296   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10297   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10298 }%

```

Unset the regular attribute if it has been set.

```

10299 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10300   \glshasattribute{\the\glslabeltok}{regular}%
10301   {%
10302     \glissetattribute{\the\glslabeltok}{regular}{false}%
10303   }%
10304   {}%
10305 }%
10306 }%
10307 {%
10308   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
10309 }

```

short-em-long Now the short (long) version

```

10310 \newabbreviationstyle{short-em-long}%
10311 {%
    Set accessibility attributes if enabled.
10312   \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
    Setup the default fields.
10313   \renewcommand*{\CustomAbbreviationFields}{%
10314     name={\glxtrshortlongname},

```

```

10315 sort={\the\glsshorttok},
10316 description={\the\glslongtok},%
10317 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10318 \protect\glsxtrfullsep{\the\glslabeltok}%
10319 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
10320 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10321 \protect\glsxtrfullsep{\the\glslabeltok}%
10322 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
10323 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10324 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10325 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10326 \glsattribute{\the\glslabeltok}{regular}%
10327 {%
10328 \glssetattribute{\the\glslabeltok}{regular}{false}%
10329 }%
10330 {}%
10331 }%
10332 }%
10333 {%

```

Mostly as short-long style:

```

10334 \renewcommand*{\abbrvpluralsuffix}{\glstremssuffix}%
10335 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10336 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10337 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10338 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10339 \renewcommand*{\glsxtrfullformat}[2]{%
10340 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10341 \ifglxtrinsertinside\else##2\fi
10342 \glsxtrfullsep{##1}%
10343 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
10344 }%
10345 \renewcommand*{\glsxtrfullplformat}[2]{%
10346 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10347 \ifglxtrinsertinside\else##2\fi
10348 \glsxtrfullsep{##1}%
10349 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
10350 }%
10351 \renewcommand*{\Glsxtrfullformat}[2]{%
10352 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10353 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10354 \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
10355 }%
10356 \renewcommand*{\Glsxtrfullplformat}[2]{%
10357 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10358 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10359 \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%

```

```

10360 }%
10361 }

```

rt-em-long-desc As before but user provides description

```

10362 \newabbreviationstyle{short-em-long-desc}%
10363 {%

```

Set accessibility attributes if enabled.

```

10364 \glstrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

10365 \renewcommand*{\CustomAbbreviationFields}{%
10366   name={\glstrshortlongdescname},
10367   sort={\glstrshortlongdescsort},
10368   first={\protect\glstrfirstabrvfont{\the\glsshorttok}}%
10369   \protect\glstrfullsep{\the\glslabeltok}}%
10370   \glstrparen{\protect\glstrfirstlongdefaultfont{\the\glslongtok}}},%
10371   firstplural={\protect\glstrfirstabrvfont{\the\glsshortpltok}}%
10372   \protect\glstrfullsep{\the\glslabeltok}}%
10373   \glstrparen{\protect\glstrfirstlongdefaultfont{\the\glslongpltok}}},%
10374   text={\protect\glstrabrvfont{\the\glsshorttok}}},%
10375   plural={\protect\glstrabrvfont{\the\glsshortpltok}}}%
10376 }%

```

Unset the regular attribute if it has been set.

```

10377 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10378   \glshasattribute{\the\glslabeltok}{regular}}%
10379   {%
10380     \glissetattribute{\the\glslabeltok}{regular}{false}}%
10381   }%
10382   {}%
10383 }%
10384 }%
10385 {%
10386   \GlsXtrUseAbbrStyleFmts{short-em-long}%
10387 }

```

hort-em-long-em

```

10388 \newabbreviationstyle{short-em-long-em}%
10389 {%

```

Set accessibility attributes if enabled.

```

10390 \glstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields. \glslongemfont is used in the description since \glsdsc doesn't set the style.

```

10391 \renewcommand*{\CustomAbbreviationFields}{%
10392   name={\glstrshortlongname},
10393   sort={\the\glsshorttok},
10394   description={\protect\glslongemfont{\the\glslongtok}}},%
10395   first={\protect\glstrfirstabrvfont{\the\glsshorttok}}%

```

```

10396 \protect\glxtrfullsep{\the\glslabeltok}%
10397 \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
10398 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10399 \protect\glxtrfullsep{\the\glslabeltok}%
10400 \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%

10401 text={\protect\glsabbrvemfont{\the\glsshorttok}}},%
10402 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%

```

Unset the regular attribute if it has been set.

```

10403 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10404 \glshasattribute{\the\glslabeltok}{regular}%
10405 {%
10406 \glissetattribute{\the\glslabeltok}{regular}{false}%
10407 }%
10408 {}%
10409 }%
10410}%
10411}%

```

```

10412 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10413 \renewcommand*{\glsabrvfont}[1]{\glsabrvemfont{##1}}%
10414 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10415 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10416 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10417 \renewcommand*{\glxtrfullformat}[2]{%
10418 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10419 \ifglxtrininsertinside\else##2\fi
10420 \glxtrfullsep{##1}%
10421 \glxtrparen{\glsfirstlongemfont{\glssaccesslong{##1}}}%
10422}%
10423 \renewcommand*{\glxtrfullplformat}[2]{%
10424 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10425 \ifglxtrininsertinside\else##2\fi
10426 \glxtrfullsep{##1}%
10427 \glxtrparen{\glsfirstlongemfont{\glssaccesslongpl{##1}}}%
10428}%
10429 \renewcommand*{\Glsxtrfullformat}[2]{%
10430 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10431 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10432 \glxtrparen{\glsfirstlongemfont{\Glsaccesslong{##1}}}%
10433}%
10434 \renewcommand*{\Glsxtrfullplformat}[2]{%
10435 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10436 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10437 \glxtrparen{\glsfirstlongemfont{\Glsaccesslongpl{##1}}}%
10438}%
10439}

```


em-long-em-desc

```
10440 \newabbreviationstyle{short-em-long-em-desc}%
10441 {%
    Set accessibility attributes if enabled.
10442   \glxstrAccSuppAbbrSetTextShortAttrs\glscategorylabel
    Setup the default fields.
10443   \renewcommand*{\CustomAbbreviationFields}{%
10444     name={\glxstrshortlongdescname},%
10445     sort={\glxstrshortlongdescsort},%
10446     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10447       \protect\glxstrfullsep{\the\glslabeltok}%
10448       \glxstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
10449     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10450       \protect\glxstrfullsep{\the\glslabeltok}%
10451       \glxstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
10452     text={\protect\glssabbrvemfont{\the\glsshorttok}},%
10453     plural={\protect\glssabbrvemfont{\the\glsshortpltok}}}%
10454   }%
    Unset the regular attribute if it has been set.
10455   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10456     \glshasattribute{\the\glslabeltok}{regular}%
10457     {%
10458       \glissetattribute{\the\glslabeltok}{regular}{false}%
10459     }%
10460     {}%
10461   }%
10462 }%
10463 {%
10464   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
10465 }
```

short-em

```
10466 \newabbreviationstyle{short-em}%
10467 {%
    Set accessibility attributes if enabled.
10468   \glxstrAccSuppAbbrSetNoLongAttrs\glscategorylabel
    Setup the default fields.
10469   \renewcommand*{\CustomAbbreviationFields}{%
10470     name={\glxstrshortnolongname},
10471     sort={\the\glsshorttok},
10472     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
10473     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
10474     text={\protect\glssabbrvemfont{\the\glsshorttok}},
10475     plural={\protect\glssabbrvemfont{\the\glsshortpltok}},
10476     description={\the\glslongtok}}%
10477   \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

10478 \glsssetAttribute{\the\glslabelltok}{regular}{true}}%
10479 }%
10480 {%

```

```

10481 \renewcommand*{\abbrvpluralsuffix}{\glstremssuffix}%
10482 \renewcommand*{\glssabrvfont}[1]{\glssabrvfont{##1}}%
10483 \renewcommand*{\glssfirstabrvfont}[1]{\glssfirstabrvfont{##1}}%
10484 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
10485 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

10486 \renewcommand*{\glssxtrinlinefullformat}[2]{%
10487 \protect\glssfirstabrvfont{\glssaccesssshort{##1}}%
10488 \ifglssxtrininsertinside##2\fi}%
10489 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10490 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
10491 }%
10492 \renewcommand*{\glssxtrinlinefullplformat}[2]{%
10493 \protect\glssfirstabrvfont{\glssaccesssshortpl{##1}}%
10494 \ifglssxtrininsertinside##2\fi}%
10495 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10496 \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
10497 }%

10498 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10499 \protect\glssfirstabrvfont{\Glsaccesssshort{##1}}%
10500 \ifglssxtrininsertinside##2\fi}%
10501 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10502 \glssxtrparen{\glssfirstlongdefaultfont{\Glsaccesslong{##1}}}%
10503 }%
10504 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10505 \protect\glssfirstabrvfont{\Glsaccesssshortpl{##1}}%
10506 \ifglssxtrininsertinside##2\fi}%
10507 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10508 \glssxtrparen{\glssfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
10509 }%

```

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.

```

10510 \renewcommand*{\glssxtrfullformat}[2]{%
10511 \glssfirstabrvfont{\glssaccesssshort{##1}}\ifglssxtrininsertinside##2\fi}%
10512 \ifglssxtrininsertinside\else##2\fi
10513 }%
10514 \renewcommand*{\glssxtrfullplformat}[2]{%
10515 \glssfirstabrvfont{\glssaccesssshortpl{##1}}\ifglssxtrininsertinside##2\fi}%
10516 \ifglssxtrininsertinside\else##2\fi
10517 }%
10518 \renewcommand*{\Glsxtrfullformat}[2]{%
10519 \glssfirstabrvfont{\glssaccesssshort{##1}}\ifglssxtrininsertinside##2\fi}%
10520 \ifglssxtrininsertinside\else##2\fi

```

```

10521 }%
10522 \renewcommand*{\Glsxtrfullplformat}[2]{%
10523   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10524   \ifglxtrinsertinside\else##2\fi
10525 }%
10526 }

```

short-em-nolong

```

10527 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

10528 \newabbreviationstyle{short-em-desc}%
10529 {%

```

Set accessibility attributes if enabled. The default name includes the long form but `\glsxtrshortdescname` could be modified to omit the long form, so include the `nameshortaccess` attribute.

```

10530 \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

10531 \renewcommand*{\CustomAbbreviationFields}{%
10532   name={\glsxtrshortdescname},
10533   sort={\the\glsshorttok},
10534   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
10535   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
10536   text={\protect\glsabbrvemfont{\the\glsshorttok}},
10537   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10538 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10539   \glssetattribute{\the\glslabeltok}{regular}{true}}%
10540 }%
10541 {%

```

```

10542 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10543 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10544 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10545 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10546 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

10547 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10548   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10549   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10550   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
10551 }%
10552 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10553   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10554   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10555   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
10556 }%
10557 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10558   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%

```

```

10559 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10560 \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
10561 }%
10562 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10563 \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10564 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10565 \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
10566 }%

```

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.

```

10567 \renewcommand*{\glxtrfullformat}[2]{%
10568 \glsfirstabbrvemfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10569 \ifglxtrinsertinside\else##2\fi
10570 }%
10571 \renewcommand*{\glxtrfullplformat}[2]{%
10572 \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10573 \ifglxtrinsertinside\else##2\fi
10574 }%
10575 \renewcommand*{\Glsxtrfullformat}[2]{%
10576 \glsfirstabbrvemfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10577 \ifglxtrinsertinside\else##2\fi
10578 }%
10579 \renewcommand*{\Glsxtrfullplformat}[2]{%
10580 \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10581 \ifglxtrinsertinside\else##2\fi
10582 }%
10583 }

```

-em-nolong-desc

```

10584 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

10585 \newabbreviationstyle{nolong-short-em}%
10586 {%
10587 \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
10588 }%
10589 {%
10590 \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

10591 \renewcommand*{\glxtrinlinefullformat}[2]{%
10592 \protect\glsfirstlongdefaultfont{\glaccesslong{##1}%
10593 \ifglxtrinsertinside##2\fi}%
10594 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10595 \glxtrparen{\glsfirstabbrvemfont{\glaccessshort{##1}}}%
10596 }%
10597 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10598 \protect\glsfirstlongdefaultfont{\glaccesslongpl{##1}%
10599 \ifglxtrinsertinside##2\fi}%

```

```

10600 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10601 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10602 }%
10603 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10604 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
10605 \ifglxtrinsertinside##2\fi}%
10606 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10607 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10608 }%
10609 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10610 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
10611 \ifglxtrinsertinside##2\fi}%
10612 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10613 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10614 }%
10615 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

10616 \newabbreviationstyle{long-noshort-em}%
10617 {%

```

Set accessibility attributes if enabled.

```
10618 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```

10619 \renewcommand*{\CustomAbbreviationFields}{%
10620 name={\glxtrlongnoshortname},
10621 sort={\the\glsshorttok},
10622 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
10623 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
10624 text={\protect\glslongdefaultfont{\the\glslongtok}},
10625 plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
10626 description={\the\glslongtok}%
10627 }%
10628 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10629 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
10630 }%
10631 {%

```

```

10632 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10633 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
10634 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10635 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10636 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10637 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10638 \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10639 \ifglxtrinsertinside \else##2\fi
10640 }%
10641 \renewcommand*{\glxtrsubsequentplfmt}[2]{%

```

```

10642 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10643 \ifglxtrinsertinside \else##2\fi
10644 }%
10645 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10646 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10647 \ifglxtrinsertinside \else##2\fi
10648 }%
10649 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10650 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10651 \ifglxtrinsertinside \else##2\fi
10652 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10653 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10654 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10655 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10656 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10657 }%
10658 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10659 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10660 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10661 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10662 }%
10663 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10664 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10665 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10666 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10667 }%
10668 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10669 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10670 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10671 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10672 }%

```

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.

```

10673 \renewcommand*{\glsxtrfullformat}[2]{%
10674 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10675 \ifglxtrinsertinside\else##2\fi
10676 }%
10677 \renewcommand*{\glsxtrfullplformat}[2]{%
10678 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10679 \ifglxtrinsertinside\else##2\fi
10680 }%
10681 \renewcommand*{\Glsxtrfullformat}[2]{%
10682 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10683 \ifglxtrinsertinside\else##2\fi
10684 }%
10685 \renewcommand*{\Glsxtrfullplformat}[2]{%
10686 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%

```

```

10687 \ifglxtrinsertinside\else##2\fi
10688 }%
10689 }

```

long-em Backward compatibility:

```

10690 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

10691 \newabbreviationstyle{long-em-noshort-em}%
10692 {%

```

Set accessibility attributes if enabled.

```

10693 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

10694 \renewcommand*{\CustomAbbreviationFields}{%
10695   name={\glxtrlongnoshortname},
10696   sort={\the\glsshorttok},
10697   first={\protect\glfirstlongemfont{\the\glslongtok}},
10698   firstplural={\protect\glfirstlongemfont{\the\glslongpltok}},
10699   text={\protect\glslongemfont{\the\glslongtok}},
10700   plural={\protect\glslongemfont{\the\glslongpltok}},%
10701   description={\protect\glslongemfont{\the\glslongtok}}%
10702 }%
10703 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10704   \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
10705 }%
10706 {%

```

```

10707 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10708 \renewcommand*{\glabbrvfnt[1]{\glabbrvemfont{##1}}}%
10709 \renewcommand*{\glfirstabbrvfnt[1]{\glfirstabbrvemfont{##1}}}%
10710 \renewcommand*{\glfirstlongfont[1]{\glfirstlongemfont{##1}}}%
10711 \renewcommand*{\glslongfont[1]{\glslongemfont{##1}}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10712 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10713   \glslongemfont{\glaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10714   \ifglxtrinsertinside \else##2\fi
10715 }%
10716 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10717   \glslongemfont{\glaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10718   \ifglxtrinsertinside \else##2\fi
10719 }%
10720 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10721   \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10722   \ifglxtrinsertinside \else##2\fi
10723 }%
10724 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10725   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%

```

```

10726 \ifglxtrinsertinside \else##2\fi
10727 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10728 \renewcommand*{\glxtrinlinefullformat}[2]{%
10729 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10730 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10731 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10732 }%
10733 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10734 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10735 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10736 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10737 }%
10738 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10739 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10740 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10741 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
10742 }%
10743 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10744 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10745 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10746 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10747 }%

```

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.

```

10748 \renewcommand*{\glxtrfullformat}[2]{%
10749 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10750 \ifglxtrinsertinside\else##2\fi
10751 }%
10752 \renewcommand*{\glxtrfullplformat}[2]{%
10753 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10754 \ifglxtrinsertinside\else##2\fi
10755 }%
10756 \renewcommand*{\Glsxtrfullformat}[2]{%
10757 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10758 \ifglxtrinsertinside\else##2\fi
10759 }%
10760 \renewcommand*{\Glsxtrfullplformat}[2]{%
10761 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10762 \ifglxtrinsertinside\else##2\fi
10763 }%
10764 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

10765 \newabbreviationstyle{long-em-noshort-em-noreg}%
10766 {}%

```

Set accessibility attributes if enabled.

```

10767 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```


Setup the default fields.

```
10768 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```
10769 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
10770 \glshasattribute{\the\glslabeltok}{regular}}%  
10771 {%  
10772 \glissetattribute{\the\glslabeltok}{regular}{false}}%  
10773 }%  
10774 {}%  
10775 }%  
10776 }%  
10777 {%  
10778 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%  
10779 }
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
10780 \newabbreviationstyle{long-noshort-em-desc}%  
10781 {%  
10782 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%  
10783 }%  
10784 {%
```

```
10785 \renewcommand*{\abbrvpluralsuffix}{\glstremsuffix}%  
10786 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%  
10787 \renewcommand*{\glssfirstabrvfont}[1]{\glssfirstabrvemfont{##1}}%  
10788 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%  
10789 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
10790 \renewcommand*{\glsxtrsubsequentfmt}[2]{%  
10791 \glsslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%  
10792 \ifglsxtrininsertinside \else##2\fi  
10793 }%  
10794 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%  
10795 \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%  
10796 \ifglsxtrininsertinside \else##2\fi  
10797 }%  
10798 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%  
10799 \glsslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%  
10800 \ifglsxtrininsertinside \else##2\fi  
10801 }%  
10802 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%  
10803 \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%  
10804 \ifglsxtrininsertinside \else##2\fi  
10805 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
10806 \renewcommand*{\glsxtrininlinefullformat}[2]{%
```

```

10807 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10808 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10809 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10810 }%
10811 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10812 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10813 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10814 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10815 }%
10816 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10817 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10818 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10819 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10820 }%
10821 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10822 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10823 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10824 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10825 }%

```

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.

```

10826 \renewcommand*{\glxtrfullformat}[2]{%
10827 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10828 \ifglxtrinsertinside\else##2\fi
10829 }%
10830 \renewcommand*{\glxtrfullplformat}[2]{%
10831 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10832 \ifglxtrinsertinside\else##2\fi
10833 }%
10834 \renewcommand*{\Glsxtrfullformat}[2]{%
10835 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10836 \ifglxtrinsertinside\else##2\fi
10837 }%
10838 \renewcommand*{\Glsxtrfullplformat}[2]{%
10839 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10840 \ifglxtrinsertinside\else##2\fi
10841 }%
10842 }

```

long-desc-em Backward compatibility:

```
10843 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glxtrshort`. The long form is emphasized. No accessibility attributes need to be set.

```

10844 \newabbreviationstyle{long-em-noshort-em-desc}%
10845 {%
10846 \renewcommand*{\CustomAbbreviationFields}{%
10847 name={\glxtrlongnoshortdescname},

```

```

10848     sort={\the\glslongtok},
10849     first={\protect\glsfirstlongemfont{\the\glslongtok}},
10850     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10851     text={\glslongemfont{\the\glslongtok}},
10852     plural={\glslongemfont{\the\glslongpltok}}}%
10853 }%
10854 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10855   \glssetattribute{\the\glslabeltok}{regular}{true}}%
10856 }%
10857 {%

10858 \renewcommand*{\abbrvpluralsuffix}{\glsxtremsuffix}%
10859 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10860 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10861 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10862 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10863 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10864   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10865   \ifglsxtrininsertinside \else##2\fi
10866 }%
10867 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10868   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10869   \ifglsxtrininsertinside \else##2\fi
10870 }%
10871 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10872   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10873   \ifglsxtrininsertinside \else##2\fi
10874 }%
10875 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10876   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10877   \ifglsxtrininsertinside \else##2\fi
10878 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10879 \renewcommand*{\glsxtrininlinefullformat}[2]{%
10880   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10881   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10882   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10883 }%
10884 \renewcommand*{\glsxtrininlinefullplformat}[2]{%
10885   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10886   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10887   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10888 }%
10889 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
10890   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10891   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10892   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%

```

```

10893 }%
10894 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10895   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10896   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10897   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10898 }%

```

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.

```

10899 \renewcommand*{\glxtrfullformat}[2]{%
10900   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10901   \ifglxtrinsertinside\else##2\fi
10902 }%
10903 \renewcommand*{\glxtrfullplformat}[2]{%
10904   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10905   \ifglxtrinsertinside\else##2\fi
10906 }%
10907 \renewcommand*{\Glsxtrfullformat}[2]{%
10908   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10909   \ifglxtrinsertinside\else##2\fi
10910 }%
10911 \renewcommand*{\Glsxtrfullplformat}[2]{%
10912   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10913   \ifglxtrinsertinside\else##2\fi
10914 }%
10915 }

```

`t-em-desc-noreg` Like `long-em-noshort-em-desc` but doesn't set the regular attribute.

```

10916 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10917 {%
10918   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

10919 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10920   \glshasattribute{\the\glslabeltok}{regular}%
10921   {%
10922     \glissetattribute{\the\glslabeltok}{regular}{false}%
10923   }%
10924   {}}%
10925 }%
10926 }%
10927 {%
10928   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10929 }

```

`ort-em-footnote`

```

10930 \newabbreviationstyle{short-em-footnote}%
10931 {%

```

Set accessibility attributes if enabled.

```

10932   \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

10933 \renewcommand*{\CustomAbbreviationFields}{%
10934   name={\glxtrfootnotename},
10935   sort={\the\glsshorttok},
10936   description={\the\glslongtok},%
10937   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10938     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10939     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
10940   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10941     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10942     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10943   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10944   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.

```

10945 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10946   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
10947   \glsattribute{\the\glslabeltok}{regular}%
10948   {%
10949     \glssetattribute{\the\glslabeltok}{regular}{false}%
10950   }%
10951   {}%
10952 }%
10953 }%
10954 {%

10955 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10956 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10957 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10958 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
10959 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10960 \renewcommand*{\glxtrfullformat}[2]{%
10961   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10962   \ifglxtrininsertinside\else##2\fi
10963   \protect\glxtrabbrvfootnote{##1}%
10964   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10965 }%
10966 \renewcommand*{\glxtrfullplformat}[2]{%
10967   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10968   \ifglxtrininsertinside\else##2\fi
10969   \protect\glxtrabbrvfootnote{##1}%
10970   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10971 }%
10972 \renewcommand*{\Glsxtrfullformat}[2]{%
10973   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10974   \ifglxtrininsertinside\else##2\fi
10975   \protect\glxtrabbrvfootnote{##1}%

```

```

10976     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10977 }%
10978 \renewcommand*{\Glsxtrfullplformat}[2]{%
10979   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10980   \ifglxtrinsertinside\else##2\fi
10981   \protect\glxtrabbrvfootnote{##1}%
10982   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10983 }%

```

The first use full form and the inline full form use the short (long) style.

```

10984 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10985   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10986   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10987   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10988 }%
10989 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10990   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10991   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10992   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10993 }%
10994 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10995   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10996   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
10997   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10998 }%
10999 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11000   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11001   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
11002   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
11003 }%
11004 }

```

footnote-em Backward compatibility:

```

11005 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

m-footnote-desc Like short-em-footnote but with user supplied description.

```

11006 \newabbreviationstyle{short-em-footnote-desc}%
11007 {%

```

Set accessibility attributes if enabled.

```

11008 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

11009 \renewcommand*{\CustomAbbreviationFields}{%
11010   name={\glxtrfootnotedesname},
11011   sort={\glxtrfootnotedesort},
11012   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
11013     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
11014     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
11015   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%

```

```

11016 \protect\glstrabbrvfootnote{\the\glslabeltok}%
11017 {\protect\glslfirstlongfootnotefont{\the\glslongpltok}}},%
11018 text={\protect\glstrabbrvemfont{\the\glsshorttok}},%
11019 plural={\protect\glstrabbrvemfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

11020 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11021 \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%
11022 \glshasattribute{\the\glslabeltok}{regular}%
11023 {%
11024 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
11025 }%
11026 {}%
11027 }%
11028 }%
11029 {%
11030 \GlsXtrUseAbbrStyleFmts{short-em-footnote}%
11031 }

```

em-postfootnote

```

11032 \newabbreviationstyle{short-em-postfootnote}%
11033 {%

```

Set accessibility attributes if enabled.

```

11034 \glstrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

11035 \renewcommand*{\CustomAbbreviationFields}{%
11036 name={\glstrfootnotename},
11037 sort={\the\glsshorttok},
11038 description={\the\glslongtok},%
11039 first={\protect\glslfirstabbrvemfont{\the\glsshorttok}},%
11040 firstplural={\protect\glslfirstabbrvemfont{\the\glsshortpltok}},%
11041 text={\protect\glstrabbrvemfont{\the\glsshorttok}},%
11042 plural={\protect\glstrabbrvemfont{\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.

```

11043 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11044 \csdef{glstrpostlink\glscategorylabel}{%
11045 \glstrifwasfirstuse
11046 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

11047 \glstrdopostpunc{\protect\glstrabbrvfootnote{\glslabel}%
11048 {\glslfirstlongfootnotefont{\glstrylong{\glslabel}}}}}%
11049 }%
11050 {}%
11051 }%

```

```

11052 \glshasattribute{\the\glslabeltok}{regular}%
11053 {%
11054 \glissetattribute{\the\glslabeltok}{regular}{false}%
11055 }%
11056 {}%
11057 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

11058 \renewcommand*\glxtrsetupfulldefs{%
11059 \let\glxtrifwasfirstuse\@secondoftwo
11060 }%
11061}%
11062{%

11063 \renewcommand*\glxtrpluralsuffix{\glxtremsuffix}%
11064 \renewcommand*\glxabbrvfont[1]{\glxabbrvemfont{##1}}%
11065 \renewcommand*\glxfirstabbrvfont[1]{\glxfirstabbrvemfont{##1}}%
11066 \renewcommand*\glxfirstlongfont[1]{\glxfirstlongfootnotefont{##1}}%
11067 \renewcommand*\glxlongfont[1]{\glxlongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

11068 \renewcommand*\glxtrfullformat[2]{%
11069 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11070 \ifglxtrinsertinside\else##2\fi
11071 }%
11072 \renewcommand*\glxtrfullplformat[2]{%
11073 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11074 \ifglxtrinsertinside\else##2\fi
11075 }%
11076 \renewcommand*\Glsxtrfullformat[2]{%
11077 \glxfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11078 \ifglxtrinsertinside\else##2\fi
11079 }%
11080 \renewcommand*\Glsxtrfullplformat[2]{%
11081 \glxfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11082 \ifglxtrinsertinside\else##2\fi
11083 }%

```

The first use full form and the inline full form use the short (long) style.

```

11084 \renewcommand*\glxtrininlinefullformat[2]{%
11085 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11086 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
11087 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslong{##1}}}%
11088 }%
11089 \renewcommand*\glxtrininlinefullplformat[2]{%
11090 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11091 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
11092 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslongpl{##1}}}%
11093 }%
11094 \renewcommand*\Glsxtrininlinefullformat[2]{%

```



```

11095 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11096 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
11097 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
11098 }%
11099 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11100 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11101 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
11102 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
11103 }%
11104 }

```

postfootnote-em Backward compatibility:

```

11105 \@glxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}

```

stfootnote-desc Like short-em-postfootnote but with user supplied description.

```

11106 \newabbreviationstyle{short-em-postfootnote-desc}%
11107 {%

```

Set accessibility attributes if enabled.

```

11108 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

11109 \renewcommand*{\CustomAbbreviationFields}{%
11110 name={\glxtrfootnotedesname},
11111 sort={\glxtrfootnotedesort},
11112 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
11113 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
11114 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
11115 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

11116 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11117 \csdef{glxtrpostlink\glscategorylabel}{%
11118 \glxtrifwasfirstuse
11119 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

11120 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
11121 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
11122 }%
11123 {}%
11124 }%
11125 \glshasattribute{\the\glslabeltok}{regular}%
11126 {%
11127 \glissetattribute{\the\glslabeltok}{regular}{false}%
11128 }%
11129 {}%
11130 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

11131 \renewcommand*{\glxtrsetupfulldefs}{%
11132   \let\glxtrifwasfirstuse\@secondoftwo
11133 }%
11134 }%
11135 {%
11136   \GlsXtrUseAbbrStyleFmts{short-em-postfootnote}%
11137 }

```

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.

```

11138 \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.

```

11139 \ifdef\glscurrentfieldvalue
11140 {
11141   \newcommand*{\glxtruserparen}[2]{%
11142     \glxtrfullsep{#2}%
11143     \glxtrparen
11144     {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
11145   }
11146 }
11147 {
11148   \newcommand*{\glxtruserparen}[2]{%
11149     \glxtrfullsep{#2}%
11150     \glxtrparen
11151     {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
11152   }
11153 }

```

Font used for short form:

`\lsabbrvuserfont`

```

11154 \newcommand*{\lsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

```

Font used for short form on first use:

`\stabbrvuserfont`

```

11155 \newcommand*{\stabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
11156 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
11157 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
11158 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
11159 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
11160 \newabbreviationstyle{long-short-user}{%
```

```
11161 {%
```

Set accessibility attributes if enabled.

```
11162 \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
11163 \renewcommand*{\CustomAbbreviationFields}{%
```

```
11164 name={\glsxtrlongshortname},
```

```
11165 sort={\the\glsshorttok},
```

```
11166 first={\protect\glsfirstlonguserfont{\the\glslongtok}}%
```

```
11167 \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
```

```
11168 {\the\glslabeltok}},%
```

```
11169 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}}%
```

```
11170 \protect\glsxtruserparen
```

```
11171 {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
```

```
11172 text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
```

```
11173 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
```

```
11174 description={\protect\glsuserdescription{\the\glslongtok}}%
```

```
11175 {\the\glslabeltok}}}%
```

Unset the regular attribute if it has been set.

```
11176 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
11177 \glshasattribute{\the\glslabeltok}{regular}}%
```

```
11178 {%
```

```
11179 \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
11180 }%
```

```
11181 {}%
```

```
11182 }%
```

```
11183 }%
```

```
11184 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

11185 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
11186 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
11187 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
11188 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11189 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11190 \renewcommand*{\glxtrfullformat}[2]{%
11191   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11192   \ifglxtrininsertinside\else##2\fi
11193   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
11194 }%
11195 \renewcommand*{\glxtrfullplformat}[2]{%
11196   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11197   \ifglxtrininsertinside\else##2\fi
11198   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
11199 }%
11200 \renewcommand*{\Glsxtrfullformat}[2]{%
11201   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11202   \ifglxtrininsertinside\else##2\fi
11203   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshort{##1}}}{##1}%
11204 }%
11205 \renewcommand*{\Glsxtrfullplformat}[2]{%
11206   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11207   \ifglxtrininsertinside\else##2\fi
11208   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshortpl{##1}}}{##1}%
11209 }%
11210 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

11211 \newabbreviationstyle{long-postshort-user}%
11212 {%

```

Set accessibility attributes if enabled.

```

11213 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11214 \renewcommand*{\CustomAbbreviationFields}{%
11215   name={\glxtrlongshortname},
11216   sort={\the\glsshorttok},
11217   first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11218   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
11219   text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11220   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
11221   description={\protect\glsuserdescription{\the\glslongtok}}%
11222   {\the\glslabeltok}}}%
11223 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11224   \csdef{glxtrpostlink\glscategorylabel}{%

```

```

11225     \glxtrifwasfirstuse
11226     {%
11227     \glxtruserparen
11228     {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
11229     {\glslabel}%
11230     }%
11231     {}%
11232     }%
11233     \glshasattribute{\the\glslabeltok}{regular}%
11234     {%
11235     \glissetattribute{\the\glslabeltok}{regular}{false}%
11236     }%
11237     {}%
11238     }%
11239 }%
11240 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11241 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
11242 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
11243 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
11244 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11245 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

11246 \renewcommand*{\glxtrfullformat}[2]{%
11247 \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11248 \ifglxtrininsertinside\else##2\fi
11249 }%
11250 \renewcommand*{\glxtrfullplformat}[2]{%
11251 \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11252 \ifglxtrininsertinside\else##2\fi
11253 }%
11254 \renewcommand*{\Glsxtrfullformat}[2]{%
11255 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11256 \ifglxtrininsertinside\else##2\fi
11257 }%
11258 \renewcommand*{\Glsxtrfullplformat}[2]{%
11259 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11260 \ifglxtrininsertinside\else##2\fi
11261 }%

```

In-line format:

```

11262 \renewcommand*{\glxtrininlinefullformat}[2]{%
11263 \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11264 \ifglxtrininsertinside\else##2\fi
11265 \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
11266 }%
11267 \renewcommand*{\glxtrininlinefullplformat}[2]{%
11268 \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

11269 \ifglxtrinsertinside\else##2\fi
11270 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
11271 }%
11272 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11273 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11274 \ifglxtrinsertinside\else##2\fi
11275 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
11276 }%
11277 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11278 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11279 \ifglxtrinsertinside\else##2\fi
11280 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
11281 }%
11282 }

```

ortuserdescname

```

11283 \newcommand*{\glxtrlongshortuserdescname}{%
11284 \protect\glslonguserfont{\the\glslongtok}%
11285 \protect\glxtruserparen
11286 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
11287 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

11288 \newabbreviationstyle{long-postshort-user-desc}%
11289 {%

```

Set accessibility attributes if enabled.

```

11290 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11291 \renewcommand*{\CustomAbbreviationFields}{%
11292 name={\glxtrlongshortuserdescname},
11293 sort={\the\glslongtok},
11294 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11295 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

11296 text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11297 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
11298 }%
11299 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11300 \csdef{glxtrpostlink\glscategorylabel}{%
11301 \glxtrifwasfirstuse
11302 {%
11303 \glxtruserparen
11304 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
11305 {\glslabel}%
11306 }%
11307 }%
11308 }%
11309 \glshasattribute{\the\glslabeltok}{regular}%

```

```

11310   {%
11311     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
11312   }%
11313   {}%
11314 }%
11315 }%
11316 {%
11317   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
11318 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

11319 \newabbreviationstyle{short-postlong-user}%
11320 {%

```

Set accessibility attributes if enabled.

```

11321 \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11322 \renewcommand*{\CustomAbbreviationFields}{%
11323   name={\glsxtrshortlongname},
11324   sort={\the\glsshorttok},
11325   first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11326   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

11327   text={\protect\glssabrvuserfont{\the\glsshorttok}},%
11328   plural={\protect\glssabrvuserfont{\the\glsshortpltok}},%
11329   description={\protect\glssabrvuserfont{\the\glslongtok}}%
11330   {\the\glslabeltok}}%
11331 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11332   \csdef{glsxtrpostlink\glscategorylabel}{%
11333     \glsxtrifwasfirstuse
11334     {%
11335       \glsxtruserparen
11336       {\glsfirstlonguserfont{\glssentrylong{\glslabel}}}%
11337       {\glslabel}%
11338     }%
11339   }%
11340 }%
11341 \glshasattribute{\the\glslabeltok}{regular}%
11342 {%
11343   \glsssetAttribute{\the\glslabeltok}{regular}{false}%
11344 }%
11345 {}%
11346 }%
11347 }%
11348 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11349 \renewcommand*{\abrvpluralsuffix}{\glsxtrusersuffix}%
11350 \renewcommand*{\glssabrvfont}[1]{\glssabrvuserfont{##1}}%
11351 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvuserfont{##1}}%

```

```

11352 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11353 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

11354 \renewcommand*{\glsxtrfullformat}[2]{%
11355   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
11356   \ifglsxtrininsertinside\else##2\fi
11357 }%
11358 \renewcommand*{\glsxtrfullplformat}[2]{%
11359   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
11360   \ifglsxtrininsertinside\else##2\fi
11361 }%
11362 \renewcommand*{\Glsxtrfullformat}[2]{%
11363   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
11364   \ifglsxtrininsertinside\else##2\fi
11365 }%
11366 \renewcommand*{\Glsxtrfullplformat}[2]{%
11367   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
11368   \ifglsxtrininsertinside\else##2\fi
11369 }%

```

In-line format:

```

11370 \renewcommand*{\glsxtrinlinefullformat}[2]{%
11371   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
11372   \ifglsxtrininsertinside\else##2\fi
11373   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
11374 }%
11375 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
11376   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
11377   \ifglsxtrininsertinside\else##2\fi
11378   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
11379 }%
11380 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11381   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
11382   \ifglsxtrininsertinside\else##2\fi
11383   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
11384 }%
11385 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11386   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
11387   \ifglsxtrininsertinside\else##2\fi
11388   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
11389 }%
11390 }

```

onguserdesname

```

11391 \newcommand*{\glsxtrshortlonguserdesname}{%
11392   \protect\glsabbrvuserfont{\the\glsshorttok}%
11393   \protect\glsxtruserparen
11394   {\protect\glslonguserfont{\the\glslongpltok}}}%
11395   {\the\glslabeltok}%

```


11396 }

long-user-desc Like short-postlong-user but leaves the user to specify the description.

11397 \newabbreviationstyle{short-postlong-user-desc}%

11398 {%

Set accessibility attributes if enabled.

11399 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

Setup the default fields.

11400 \renewcommand*{\CustomAbbreviationFields}{%

11401 name={\glxtrshortlonguserdescname},

11402 sort={\the\glsshorttok},

11403 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%

11404 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

11405 text={\protect\glsabbrvuserfont{\the\glsshorttok}},%

11406 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%

11407 }%

11408 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

11409 \csdef{glxtrpostlink\glscategorylabel}{%

11410 \glxtrifwasfirstuse

11411 {%

11412 \glxtruserparen

11413 {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%

11414 {\glslabel}%

11415 }%

11416 {}%

11417 }%

11418 \glshasattribute{\the\glslabeltok}{regular}%

11419 {%

11420 \glissetattribute{\the\glslabeltok}{regular}{false}%

11421 }%

11422 {}%

11423 }%

11424 }%

11425 {%

11426 \GlsXtrUseAbbrStyleFmts{short-postlong-user}%

11427 }

short-user-desc

11428 \newabbreviationstyle{long-short-user-desc}%

11429 {%

Set accessibility attributes if enabled.

11430 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

Setup the default fields.

11431 \renewcommand*{\CustomAbbreviationFields}{%

11432 name={\glsxtrlongshortuserdescname},

11433 sort={\glsxtrlongshortdescsort},%

```

11434   first={\protect\glsfirstlonguserfont{\the\glslongtok}%
11435     \protect\glstruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
11436     {\the\glslabeltok}},%
11437   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
11438     \protect\glstruserparen
11439     {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
11440   text={\protect\glsabbrvfont{\the\glsshorttok}},%
11441   plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
11442 }%

```

Unset the regular attribute if it has been set.

```

11443 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11444   \glshasattribute{\the\glslabeltok}{regular}%
11445   {%
11446     \glissetattribute{\the\glslabeltok}{regular}{false}%
11447   }%
11448   {}}%
11449 }%
11450 }%
11451 {%
11452   \GlsXtrUseAbbrStyleFmts{long-short-user}%
11453 }

```

short-long-user

```

11454 \newabbreviationstyle{short-long-user}%
11455 {%

```

Set accessibility attributes if enabled.

```

11456   \glstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

11457 \renewcommand*{\CustomAbbreviationFields}{%
11458   name={\glstrshortlongname},
11459   sort={\the\glsshorttok},
11460   description={\protect\glsuserdescription{\the\glslongtok}%
11461     {\the\glslabeltok}},%
11462   first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
11463     \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
11464     {\the\glslabeltok}},%
11465   firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
11466     \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
11467     {\the\glslabeltok}},%

11468   text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11469   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

11470 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

11471 \glshasattribute{\the\glslabeltok}{regular}%
11472 {%
11473 \glissetattribute{\the\glslabeltok}{regular}{false}%
11474 }%
11475 {}%
11476 }%
11477 }%
11478 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11479 \renewcommand*\abbrvpluralsuffix{\glsxtrusersuffix}%
11480 \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
11481 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvuserfont{##1}}%
11482 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonguserfont{##1}}%
11483 \renewcommand*\glslongfont[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11484 \renewcommand*\glsxtrfullformat[2]{%
11485 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11486 \ifglxtrinsertinside\else##2\fi
11487 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
11488 }%
11489 \renewcommand*\glsxtrfullplformat[2]{%
11490 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11491 \ifglxtrinsertinside\else##2\fi
11492 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
11493 }%
11494 \renewcommand*\Glsxtrfullformat[2]{%
11495 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11496 \ifglxtrinsertinside\else##2\fi
11497 \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
11498 }%
11499 \renewcommand*\Glsxtrfullplformat[2]{%
11500 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11501 \ifglxtrinsertinside\else##2\fi
11502 \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
11503 }%
11504 }

```

-long-user-desc

```

11505 \newabbreviationstyle{short-long-user-desc}%
11506 {%

```

Set accessibility attributes if enabled.

```

11507 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11508 \renewcommand*\CustomAbbreviationFields{%
11509 name={\glsxtrshortlonguserdescname},
11510 sort={\glsxtrshortlongdescsort},%

```

```

11511 first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
11512 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
11513 {\the\glslabeltok}},%
11514 firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}%
11515 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
11516 {\the\glslabeltok}},%
11517 text={\protect\glsabbrvfont{\the\glsshorttok}},%
11518 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
11519 }%

```

Unset the regular attribute if it has been set.

```

11520 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11521 \glshasattribute{\the\glslabeltok}{regular}}%
11522 {%
11523 \glissetattribute{\the\glslabeltok}{regular}{false}}%
11524 }%
11525 {}%
11526 }%
11527 }%
11528 {%
11529 \GlsXtrUseAbbrStyleFmts{short-long-user}}%
11530 }

```

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 `\glsxtrlong` set `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

`trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```

11531 \newrobustcmd*{\glsxtrifhyphenstart}[3]{%
11532 \ifx\glsinsert#1\relax
11533 \expandafter\@glsxtrifhyphenstart#1\relax\relax
11534 \@end@glsxtrifhyphenstart{#2}{#3}}%
11535 \else
11536 \@glsxtrifhyphenstart#1\relax\relax\@end@glsxtrifhyphenstart{#2}{#3}}%
11537 \fi
11538 }

```

`trifhyphenstart`

```

11539 \def\@glsxtrifhyphenstart#1#2\@end@glsxtrifhyphenstart#3#4{%
11540 \ifx-#1\relax#3\else #4\fi
11541 }

```

`longhyphenshort`

```
\glxstrlonghyphenshort{<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.

```
11542 \newcommand*{\glxstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
11543 {%
```

If *<insert>* starts with a hyphen, redefine `\glxstrwordsep` 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 `\glxstrwordsep` if *<insert>* doesn't start with a hyphen.

```
11544 \glxstrifhyphenstart{#4}{\def\glxstrwordsep{-}}{}%
11545 \glsfirstlonghyphenfont{#2\ifglxstrinsertinside{#4}\fi}%
11546 \ifglxstrinsertinside\else{#4}\fi
11547 \glxstrfullsep{#1}%
11548 \glxstrparen{\glsfirstabbrvhyphenfont{#3\ifglxstrinsertinside{#4}\fi}%
11549 \ifglxstrinsertinside\else{#4}\fi}%
11550 }%
11551 }
```

abbrvhyphenfont

```
11552 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
11553 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
11554 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

tlonghyphenfont

```
11555 \newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
11556 \newcommand*{\glxstrhyphensuffix}{\glxstrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the `markwords` attribute.

```
11557 \newabbreviationstyle{long-hyphen-short-hyphen}%
```

```
11558 {%
```

Set accessibility attributes if enabled.

```
11559 \glxstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
11560 \renewcommand*{\CustomAbbreviationFields}{%
```

```
11561 name={\glxstrlongshortname},
```

```
11562 sort={\the\glsshorttok},
```

```

11563 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
11564 \protect\glsxtrfullsep{\the\glslabeltok}%
11565 \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
11566 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
11567 \protect\glsxtrfullsep{\the\glslabeltok}%
11568 \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
11569 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11570 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
11571 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

11572 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11573 \glsasattribute{\the\glslabeltok}{regular}%
11574 {%
11575 \glssetattribute{\the\glslabeltok}{regular}{false}%
11576 }%
11577 {}%
11578 }%
11579}%
11580{%
11581 \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
11582 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11583 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11584 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11585 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11586 \renewcommand*{\glsxtrfullformat}[2]{%
11587 \glsxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
11588}%
11589 \renewcommand*{\glsxtrfullplformat}[2]{%
11590 \glsxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
11591 {\glsaccessshortpl{##1}}{##2}%
11592}%
11593 \renewcommand*{\Glsxtrfullformat}[2]{%
11594 \glsxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
11595}%
11596 \renewcommand*{\Glsxtrfullplformat}[2]{%
11597 \glsxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
11598 {\glsaccessshortpl{##1}}{##2}%
11599}%
11600}

```

`ort-hyphen-desc` Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

11601 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
11602 {%

```

Set accessibility attributes if enabled.

```

11603 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11604 \renewcommand*{\CustomAbbreviationFields}{%
11605     name={\glxtrlongshortdescname},
11606     sort={\glxtrlongshortdescsort},
11607     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
11608         \protect\glxtrfullsep{\the\glslabeltok}%
11609         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
11610     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
11611         \protect\glxtrfullsep{\the\glslabeltok}%
11612         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
11613     text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
11614     plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
11615 }%

```

Unset the regular attribute if it has been set.

```

11616 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11617     \glshasattribute{\the\glslabeltok}{regular}%
11618     {%
11619         \glissetattribute{\the\glslabeltok}{regular}{false}%
11620     }%
11621     {}%
11622 }%
11623 }%
11624 {%
11625     \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
11626 }

```

nghyphennoshort

`\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}`

```

11627 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

11628 {%

```

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.

```

11629     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
11630     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
11631     \ifglxtrininsertinside\else{#3}\fi
11632 }%
11633 }

```

hort-desc-noreg

This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough. No accessibility attributes need to be set.

```

11634 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
11635 {%
11636   \renewcommand*{\CustomAbbreviationFields}{%
11637     name={\glxtrlongnoshortdescname},
11638     sort={\expandonce\glxtrorglong},
11639     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11640     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11641     text={\protect\glslonghyphenfont{\the\glslongtok}},%
11642     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
11643   }%

```

Unset the regular attribute if it has been set.

```

11644   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11645     \glshasattribute{\the\glslabeltok}{regular}%
11646     {%
11647       \glissetattribute{\the\glslabeltok}{regular}{false}%
11648     }%
11649   }%
11650 }%
11651 }%
11652 {%
11653   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11654   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11655   \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
11656   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
11657   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11658   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

11659   \renewcommand*{\glxtrsubsequentfmt}[2]{%
11660     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
11661   }%
11662   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11663     \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
11664   }%
11665   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11666     \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11667   }%
11668   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11669     \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11670   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

11671   \renewcommand*{\glxtrinlinefullformat}[2]{%
11672     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
11673     \glxtrfullsep{##1}%
11674     \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
11675   }%
11676   \renewcommand*{\glxtrinlinefullplformat}[2]{%

```



```

11677 \glxstrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
11678 \glxstrfullsep{##1}%
11679 \glxstrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
11680 }%
11681 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11682 \glxstrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11683 \glxstrfullsep{##1}%
11684 \glxstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
11685 }%
11686 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11687 \glxstrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11688 \glxstrfullsep{##1}%
11689 \glxstrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
11690 }%

```

The first use full form only displays the long form.

```

11691 \renewcommand*{\glxstrfullformat}[2]{%
11692 \glxstrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
11693 }%
11694 \renewcommand*{\glxstrfullplformat}[2]{%
11695 \glxstrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
11696 }%
11697 \renewcommand*{\Glsxtrfullformat}[2]{%
11698 \glxstrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11699 }%
11700 \renewcommand*{\Glsxtrfullplformat}[2]{%
11701 \glxstrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11702 }%
11703 }

```

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.

```

11704 \newabbreviationstyle{long-hyphen-noshort-noreg}%
11705 {%

```

Set accessibility attributes if enabled.

```

11706 \glxstrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

11707 \renewcommand*{\CustomAbbreviationFields}{%
11708 name={\glxstrlongnoshortname},
11709 sort={\the\glsshorttok},
11710 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11711 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11712 text={\protect\glslonghyphenfont{\the\glslongtok}},%
11713 plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
11714 description={\the\glslongtok}%
11715 }%

```

Unset the regular attribute if it has been set.

```
11716 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11717   \glshasattribute{\the\glslabeltok}{regular}%
11718   {%
11719     \glsselattribute{\the\glslabeltok}{regular}{false}%
11720   }%
11721   {}%
11722 }%
11723}%
11724%
```

```
11725 \GlsXtrUseAbbrStyleFmts{long-hyphen-noshort-desc-noreg}%
11726 }
```

lsxtrlonghyphen

`\lsxtrlonghyphen{<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.

```
11727 \newcommand*{\lsxtrlonghyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
11728   {%
11729     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
11730     \glsfirslonghyphenfont{#1}%
11731   }%
11732 }
```

posthyphenshort

`\lsxtrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\lsxtrlonghyphenshort` but omits the `<long>` part. This always uses the singular short form.

```
11733 \newcommand*{\lsxtrposthyphenshort}[2]{%
11734   {%
11735     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11736     \ifglxtrininsertinside{\glsfirslonghyphenfont{#2}}\else{#2}\fi
11737     \glxtrfullsep{#1}%
11738     \glxtrparen
11739     {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglxtrininsertinside{#2}\fi}%
11740     \ifglxtrininsertinside\else{#2}\fi
11741   }%
11742 }%
11743 }
```

ypkensubsequent

```
\glxtrposthyphensubsequent{<label>}{<insert>}
```

Format in the post-link hook for subsequent use. The label is ignored by default.

```
11744 \newcommand*{\glxtrposthyphensubsequent}[2]{%
11745   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
11746   \ifglxtrininsertinside \else{#2}\fi
11747 }
```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
11748 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
11749 {%
```

Set accessibility attributes if enabled.

```
11750 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
11751 \renewcommand*{\CustomAbbreviationFields}{%
11752   name={\glxtrlongshortname},
11753   sort={\the\glsshorttok},
11754   first={\protect\glssfirstlonghyphenfont{\the\glslongtok}},%
11755   firstplural={\protect\glssfirstlonghyphenfont{\the\glslongpltok}},%
11756   text={\protect\glssabrvhyphenfont{\the\glsshorttok}},%
11757   plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
11758   description={\protect\glsslonghyphenfont{\the\glslongtok}}}%
11759 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11760   \csdef{glxtrpostlink\glscategorylabel}{%
11761     \glxtrifwasfirstuse
11762     {%
11763       \glxtrposthyphenshort{\glslabel}{\glsinsert}%
11764     }%
11765     {%
```

Put the insertion into the post-link:

```
11766       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11767     }%
11768   }%
11769   \glshasattribute{\the\glslabeltok}{regular}%
11770   {%
11771     \glsssetattribute{\the\glslabeltok}{regular}{false}%
11772   }%
11773   {}%
11774 }%
11775 }%
11776 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
11777 \renewcommand*{\abrvpluralsuffix}{\glxtrabrvpluralsuffix}%
```

```

11778 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11779 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11780 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11781 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

11782 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
11783   \glsabbrvfont{\glsaccessshort{##1}}}%
11784 }%
11785 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
11786   \glsabbrvfont{\glsaccessshortpl{##1}}}%
11787 }%
11788 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11789   \glsabbrvfont{\Glsaccessshort{##1}}}%
11790 }%
11791 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11792   \glsabbrvfont{\Glsaccessshortpl{##1}}}%
11793 }%

```

First use full form:

```

11794 \renewcommand*{\glsxtrfullformat}[2]{%
11795   \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
11796 }%
11797 \renewcommand*{\glsxtrfullplformat}[2]{%
11798   \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
11799 }%
11800 \renewcommand*{\Glsxtrfullformat}[2]{%
11801   \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
11802 }%
11803 \renewcommand*{\Glsxtrfullplformat}[2]{%
11804   \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
11805 }%

```

In-line format.

```

11806 \renewcommand*{\glsxtrinlinefullformat}[2]{%
11807   \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
11808   \ifglsxtrininsertinside{##2}\fi}%
11809 \ifglsxtrininsertinside \else{##2}\fi
11810 }%
11811 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
11812   \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
11813   \ifglsxtrininsertinside{##2}\fi}%
11814 \ifglsxtrininsertinside \else{##2}\fi
11815 }%
11816 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11817   \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
11818   \ifglsxtrininsertinside{##2}\fi}%
11819 \ifglsxtrininsertinside \else{##2}\fi
11820 }%
11821 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11822   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%

```

```

11823     \ifglxtrinsertinside{##2}\fi}%
11824     \ifglxtrinsertinside \else{##2}\fi
11825 }%
11826 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

11827 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
11828 {%

```

Set accessibility attributes if enabled.

```

11829 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11830 \renewcommand*{\CustomAbbreviationFields}{%
11831     name={\glxtrlongshortdescname},
11832     sort={\glxtrlongshortdescsort},%
11833     first={\protect\glslonghyphenfont{\the\glslongtok}},%
11834     firstplural={\protect\glslonghyphenfont{\the\glslongpltok}},%
11835     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11836     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11837 }%
11838 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11839     \csdef{glxtrpostlink\glscategorylabel}{%
11840         \glxtrifwasfirstuse
11841         {%
11842             \glxtrposthyphenshort{\glslabel}{\glsinsert}%
11843         }%
11844         {%

```

Put the insertion into the post-link:

```

11845         \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11846     }%
11847 }%
11848 \glshasattribute{\the\glslabeltok}{regular}%
11849 {%
11850     \glissetattribute{\the\glslabeltok}{regular}{false}%
11851 }%
11852 {}%
11853 }%
11854 }%
11855 {%
11856 \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
11857 }

```

shorthyphenlong

```
\glxtrshorthyphenlong{<label>}{<short>}{<long>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

11858 \newcommand*{\glxtrshorthyphenlong}[4]{%

Grouping is needed to localise the redefinitions.

11859 {%

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.)

11860 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%

11861 \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%

11862 \ifglxtrininsertinside\else{#4}\fi

11863 \glxtrfullsep{#1}%

11864 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%

11865 \ifglxtrininsertinside\else{#4}\fi}%

11866 }%

11867 }

hen-long-hyphen Designed for use with the markwords attribute.

11868 \newabbreviationstyle{short-hyphen-long-hyphen}%

11869 {%

Set accessibility attributes if enabled.

11870 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

Setup the default fields.

11871 \renewcommand*{\CustomAbbreviationFields}{%

11872 name={\glxtrshortlongname},

11873 sort={\the\glsshorttok},

11874 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%

11875 \protect\glxtrfullsep{\the\glslabeltok}%

11876 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%

11877 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%

11878 \protect\glxtrfullsep{\the\glslabeltok}%

11879 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%

11880 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%

11881 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%

11882 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

Unset the regular attribute if it has been set.

11883 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

11884 \glshasattribute{\the\glslabeltok}{regular}%

11885 {%

11886 \glissetattribute{\the\glslabeltok}{regular}{false}%

11887 }%

11888 {}%

11889 }%

11890 }%

11891 {%

11892 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%

11893 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%

11894 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%

```

11895 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11896 \renewcommand*\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11897 \renewcommand*\glsxtrfullformat}[2]{%
11898   \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
11899 }%
11900 \renewcommand*\glsxtrfullplformat}[2]{%
11901   \glsxtrshorthyphenlong{##1}%
11902   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
11903 }%
11904 \renewcommand*\Glsxtrfullformat}[2]{%
11905   \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
11906 }%
11907 \renewcommand*\Glsxtrfullplformat}[2]{%
11908   \glsxtrshorthyphenlong{##1}%
11909   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
11910 }%
11911 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

11912 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
11913 {}%

```

Set accessibility attributes if enabled.

```

11914 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11915 \renewcommand*\CustomAbbreviationFields{%
11916   name={\glsxtrshortlongdescname},
11917   sort={\glsxtrshortlongdescsort},
11918   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
11919   \protect\glsxtrfullsep{\the\glslabeltok}}%
11920   \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11921   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
11922   \protect\glsxtrfullsep{\the\glslabeltok}}%
11923   \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11924   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
11925   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11926 }%

```

Unset the regular attribute if it has been set.

```

11927 \renewcommand*\GlsXtrPostNewAbbreviation{%
11928   \glsattribute{\the\glslabeltok}{regular}%
11929   {%
11930     \glssetattribute{\the\glslabeltok}{regular}{false}%
11931   }}%
11932 {}%
11933 }%
11934 }%
11935 {}%

```

```

11936 \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11937 }

```

sxtrshorthyphen

```
\glsxtrshorthyphen{<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.

```
11938 \newcommand*{\glsxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

11939 {%
11940   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
11941   \glsfirstabbrvhyphenfont{#1}%
11942 }%
11943 }

```

rpostthyphenlong

```
\glsxtrpostthyphenlong{<label>}{<insert>}
```

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glsxtrshorthyphenlong` but omits the *<short>* part. This always uses the singular long form.

```

11944 \newcommand*{\glsxtrpostthyphenlong}[2]{%
11945 {%
11946   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
11947   \ifglsxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11948   \glsxtrfullsep{#1}%
11949   \glsxtrparen
11950   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglsxtrininsertinside{#2}\fi}%
11951   \ifglsxtrininsertinside\else{#2}\fi
11952 }%
11953 }%
11954 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

11955 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11956 {%

```

Set accessibility attributes if enabled.

```
11957 \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```

11958 \renewcommand*{\CustomAbbreviationFields}{%
11959   name={\glsxtrshortlongname},

```



```

11960     sort={\the\glsshorttok},
11961     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11962     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11963     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11964     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
11965     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11966 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11967   \csdef{glsxtrpostlink\glscategorylabel}{%
11968     \glsxtrifwasfirstuse
11969     {%
11970       \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
11971     }%
11972   }%

```

Put the insertion into the post-link:

```

11973     \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11974   }%
11975 }%
11976 \glsattribute{\the\glslabeltok}{regular}%
11977 {%
11978   \glssetattribute{\the\glslabeltok}{regular}{false}%
11979 }%
11980 {}%
11981 }%
11982 }%
11983 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11984 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
11985 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11986 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11987 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11988 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

11989 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
11990   \glsabbrvfont{\glsaccessshort{##1}}%
11991 }%
11992 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
11993   \glsabbrvfont{\glsaccessshortpl{##1}}%
11994 }%
11995 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11996   \glsabbrvfont{\Glsaccessshort{##1}}%
11997 }%
11998 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11999   \glsabbrvfont{\Glsaccessshortpl{##1}}%
12000 }%

```

First use full form:

```

12001 \renewcommand*{\glsxtrfullformat}[2]{%
12002   \glsxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%

```

```

12003 }%
12004 \renewcommand*{\glxtrfullplformat}[2]{%
12005   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
12006 }%
12007 \renewcommand*{\Glsxtrfullformat}[2]{%
12008   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
12009 }%
12010 \renewcommand*{\Glsxtrfullplformat}[2]{%
12011   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
12012 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

12013 \renewcommand*{\glxtrinlinefullformat}[2]{%
12014   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
12015   \ifglxtrininsertinside{##2}\fi}%
12016   \ifglxtrininsertinside \else{##2}\fi
12017 }%
12018 \renewcommand*{\glxtrinlinefullplformat}[2]{%
12019   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
12020   \ifglxtrininsertinside{##2}\fi}%
12021   \ifglxtrininsertinside \else{##2}\fi
12022 }%
12023 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
12024   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
12025   \ifglxtrininsertinside{##2}\fi}%
12026   \ifglxtrininsertinside \else{##2}\fi
12027 }%
12028 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
12029   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
12030   \ifglxtrininsertinside{##2}\fi}%
12031   \ifglxtrininsertinside \else{##2}\fi
12032 }%
12033 }

```

`ong-hyphen-desc` Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

12034 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
12035 {%

```

Set accessibility attributes if enabled.

```

12036 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

12037 \renewcommand*{\CustomAbbreviationFields}{%
12038   name={\glxtrshortlongdescname},
12039   sort={\glxtrshortlongdescsort},%
12040   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
12041   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
12042   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
12043   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%

```

```

12044 }%
12045 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
12046   \csdef{glxtrpostlink\glscategorylabel}{%
12047     \glxtrifwasfirstuse
12048     {%
12049       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
12050     }%
12051     {%

```

Put the insertion into the post-link:

```

12052       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
12053     }%
12054   }%
12055   \glshasattribute{\the\glslabeltok}{regular}%
12056   {%
12057     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
12058   }%
12059   {}%
12060 }%
12061 }%
12062 {%
12063   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
12064 }

```

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

```

12065 \newcommand*{\glabbrvonlyfont}{\glabbrvdefaultfont}%

```

stabbrvonlyfont

```

12066 \newcommand*{\glfirstabbrvonlyfont}{\glabbrvonlyfont}%

```

glslongonlyfont

```

12067 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

```

rstlongonlyfont

```

12068 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%

```

The default short form suffix:

lsxtronlysuffix

```

12069 \newcommand*{\glxtronlysuffix}{\glxtrabbrvpluralsuffix}

```

\glxtronlyname The default name format for this style.

```

12070 \newcommand*{\glxtronlyname}{%
12071   \protect\glabbrvonlyfont{\the\glsshorttok}%
12072 }

```

only-short-only

```
12073 \newabbreviationstyle{long-only-short-only}%  
12074 {%
```

Set accessibility attributes if enabled.

```
12075 \glstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
12076 \renewcommand*{\CustomAbbreviationFields}{%  
12077   name={\glstronlyname},  
12078   sort={\the\glsshorttok},  
12079   first={\protect\glstrfirstlongonlyfont{\the\glslongtok}},%  
12080   firstplural={\protect\glstrfirstlongonlyfont{\the\glslongpltok}},%  
12081   text={\protect\glstrabbrvonlyfont{\the\glsshorttok}},%  
12082   plural={\protect\glstrabbrvonlyfont{\the\glsshortpltok}},%  
12083   description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
12084 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
12085   \glshasattribute{\the\glslabeltok}{regular}%  
12086   {%  
12087     \glissetattribute{\the\glslabeltok}{regular}{false}%  
12088   }%  
12089   {}%  
12090 }%  
12091 }%  
12092 {%
```

```
12093 \renewcommand*{\abbrvpluralsuffix}{\glstronlysuffix}%  
12094 \renewcommand*{\glstrabbrvfont}[1]{\glstrabbrvonlyfont{##1}}%  
12095 \renewcommand*{\glstrfirstabbrvfont}[1]{\glstrfirstabbrvonlyfont{##1}}%  
12096 \renewcommand*{\glstrfirstlongfont}[1]{\glstrfirstlongonlyfont{##1}}%  
12097 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
12098 \renewcommand*{\glstrfullformat}[2]{%  
12099   \glstrfirstlongonlyfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%  
12100   \ifglstrinsertinside\else##2\fi  
12101 }%  
12102 \renewcommand*{\glstrfullplformat}[2]{%  
12103   \glstrfirstlongonlyfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%  
12104   \ifglstrinsertinside\else##2\fi  
12105 }%  
12106 \renewcommand*{\GlsXtrfullformat}[2]{%  
12107   \glstrfirstlongonlyfont{\Glsstraccesslong{##1}\ifglstrinsertinside##2\fi}%  
12108   \ifglstrinsertinside\else##2\fi  
12109 }%  
12110 \renewcommand*{\GlsXtrfullplformat}[2]{%  
12111   \glstrfirstlongonlyfont{\Glsstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%  
12112   \ifglstrinsertinside\else##2\fi  
12113 }%
```

The inline full form does show the short form.

```

12114 \renewcommand*{\glxtrinlinefullformat}[2]{%
12115   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
12116   \ifglxtrinsertinside\else##2\fi
12117   \glxtrfullsep{##1}%
12118   \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
12119 }%
12120 \renewcommand*{\glxtrinlinefullplformat}[2]{%
12121   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
12122   \ifglxtrinsertinside\else##2\fi
12123   \glxtrfullsep{##1}%
12124   \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
12125 }%
12126 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
12127   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
12128   \ifglxtrinsertinside\else##2\fi
12129   \glxtrfullsep{##1}%
12130   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
12131 }%
12132 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
12133   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
12134   \ifglxtrinsertinside\else##2\fi
12135   \glxtrfullsep{##1}%
12136   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
12137 }%
12138 }

```

xtronlydescsort

```

12139 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

12140 \newcommand*{\glxtronlydescname}{%
12141   \protect\glslongfont{\the\glslongtok}}%
12142 }

```

short-only-desc

```

12143 \newabbreviationstyle{long-only-short-only-desc}%
12144 {}%

```

Set accessibility attributes if enabled.

```

12145 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

12146 \renewcommand*{\CustomAbbreviationFields}{%
12147   name={\glxtronlydescname},
12148   sort={\glxtronlydescsort},%
12149   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
12150   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
12151   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%

```

```

12152 plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
12153 }%
    Unset the regular attribute if it has been set.
12154 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
12155   \glshasattribute{\the\glslabeltok}{regular}%
12156   {%
12157     \glissetattribute{\the\glslabeltok}{regular}{false}%
12158   }%
12159   {}}%
12160 }%
12161 }%
12162 {%
12163   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
12164 }

```

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:

```

12165 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```

12166 \renewcommand*{\markright}[1]{%
12167   \glsxtrmarkhook

```

```

12168 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
12169 \glsxtrrestoremarkhook
12170 }

```

\markboth Save original definition:

```

12171 \let\@glsxtr@org@markboth\markboth

```

Redefine (grouping not added in case it interferes with the original code):

```

12172 \renewcommand*{\markboth}[2]{%
12173   \glsxtrmarkhook
12174   \@glsxtr@org@markboth
12175   {\@glsxtrinmark#1\@glsxtrnotinmark}%
12176   {\@glsxtrinmark#2\@glsxtrnotinmark}%
12177   \glsxtrrestoremarkhook
12178 }

```

Also do this for \@starttoc

\@starttoc Save original definition:

```

12179 \let\@glsxtr@org@@starttoc\@starttoc

```

Redefine:

```

12180 \renewcommand*{\@starttoc}[1]{%
12181   \glsxtrmarkhook
12182   \@glsxtrinmark
12183   \@glsxtr@org@@starttoc{#1}%
12184   \@glsxtrnotinmark
12185   \glsxtrrestoremarkhook
12186 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

12187 \newcommand*{\glsxtrRevertMarks}{%
12188   \let\markright\@glsxtr@org@markright
12189   \let\markboth\@glsxtr@org@markboth
12190   \let\@starttoc\@glsxtr@org@@starttoc
12191 }

```

rRevertTocMarks Just restores \@starttoc.

```

12192 \newcommand*{\glsxtrRevertTocMarks}{%
12193   \let\@starttoc\@glsxtr@org@@starttoc
12194 }

```

\glsxtrifinmark

```

12195 \newcommand*{\glsxtrifinmark}[2]{#2}

```

\@glsxtrinmark

```

12196 \newrobustcmd*{\@glsxtrinmark}{%
12197   \let\glsxtrifinmark\@firstoftwo
12198 }

```

glsxtrnotinmark

```
12199 \newrobustcmd*{\@glsxtrnotinmark}{%
12200   \let\glsxtrifinmark\@secondoftwo
12201 }
```

eorpdforheading

```
12202 \ifdef\teorpdfstring
12203 {
12204   \newcommand*{\glsxtrtitleorpdforheading}[3]{\teorpdfstring{#1}{#2}}
12205 }
12206 {
12207   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
12208 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
12209 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
12210 \let\@glsxtr@org@MakeUppercase\MakeUppercase
12211 \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
12212 \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
12213 \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
12214 \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
12215 \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
12216 \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
12217 \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
12218 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
12219 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
12220 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
12221 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
12222 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
12223 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
12224 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
12225 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
12226 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
12227 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
12228 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
12229 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
12230 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
12231 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
12232 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
12233 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
12234 \let\glsxtrifinmark\@firstoftwo
12235 \let\MakeUppercase\MakeTextUppercase
12236 \let\glsxtrtitleorpdforheading\@thirdofthree
12237 \let\glsxtrtitleshort\glsxtrheadshort
12238 \let\glsxtrtitleshortpl\glsxtrheadshortpl
```



```

12239 \let\Glsxtrtitleshort\Glsxtrheadshort
12240 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
12241 \let\Glsxtrtitlename\Glsxtrheadname
12242 \let\Glsxtrtitlename\Glsxtrheadname
12243 \let\Glsxtrtitletext\Glsxtrheadtext
12244 \let\Glsxtrtitletext\Glsxtrheadtext
12245 \let\Glsxtrtitleplural\Glsxtrheadplural
12246 \let\Glsxtrtitleplural\Glsxtrheadplural
12247 \let\Glsxtrtitlefirst\Glsxtrheadfirst
12248 \let\Glsxtrtitlefirst\Glsxtrheadfirst
12249 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
12250 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
12251 \let\Glsxtrtitlelong\Glsxtrheadlong
12252 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
12253 \let\Glsxtrtitlelong\Glsxtrheadlong
12254 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
12255 \let\Glsxtrtitlefull\Glsxtrheadfull
12256 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
12257 \let\Glsxtrtitlefull\Glsxtrheadfull
12258 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
12259 }

```

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.)

```

12260 \newcommand*{\Glsxtrrestoremarkhook}{%
12261   \let\Glsxtrifinmark\@secondoftwo
12262   \let\MakeUppercase\@Glsxtr@org@MakeUppercase
12263   \let\Glsxtrtitleorpdforheading\@Glsxtr@org@Glsxtrtitleorpdforheading
12264   \let\Glsxtrtitleshort\@Glsxtr@org@Glsxtrtitleshort
12265   \let\Glsxtrtitleshortpl\@Glsxtr@org@Glsxtrtitleshortpl
12266   \let\Glsxtrtitleshort\@Glsxtr@org@Glsxtrtitleshort
12267   \let\Glsxtrtitleshortpl\@Glsxtr@org@Glsxtrtitleshortpl
12268   \let\Glsxtrtitlename\@Glsxtr@org@Glsxtrtitlename
12269   \let\Glsxtrtitlename\@Glsxtr@org@Glsxtrtitlename
12270   \let\Glsxtrtitletext\@Glsxtr@org@Glsxtrtitletext
12271   \let\Glsxtrtitletext\@Glsxtr@org@Glsxtrtitletext
12272   \let\Glsxtrtitleplural\@Glsxtr@org@Glsxtrtitleplural
12273   \let\Glsxtrtitleplural\@Glsxtr@org@Glsxtrtitleplural
12274   \let\Glsxtrtitlefirst\@Glsxtr@org@Glsxtrtitlefirst
12275   \let\Glsxtrtitlefirst\@Glsxtr@org@Glsxtrtitlefirst
12276   \let\Glsxtrtitlefirstplural\@Glsxtr@org@Glsxtrtitlefirstplural
12277   \let\Glsxtrtitlefirstplural\@Glsxtr@org@Glsxtrtitlefirstplural
12278   \let\Glsxtrtitlelong\@Glsxtr@org@Glsxtrtitlelong
12279   \let\Glsxtrtitlelongpl\@Glsxtr@org@Glsxtrtitlelongpl
12280   \let\Glsxtrtitlelong\@Glsxtr@org@Glsxtrtitlelong
12281   \let\Glsxtrtitlelongpl\@Glsxtr@org@Glsxtrtitlelongpl
12282   \let\Glsxtrtitlefull\@Glsxtr@org@Glsxtrtitlefull

```

```

12283 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
12284 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
12285 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
12286 }

```

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.

```

12287 \newcommand*{\glsxtrheadshort}[1]{%
12288 \protect\NoCaseChange
12289 {%
12290 \glsifattribute{#1}{headuc}{true}%
12291 {%
12292 \GLSxtrshort[noindex,hyper=false]{#1}[]%
12293 }%
12294 {%
12295 \glsxtrshort[noindex,hyper=false]{#1}[]%
12296 }%
12297 }%
12298 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

12299 \newrobustcmd*{\lsxtrtitleshort}[1]{%
12300 \glsxtrshort[noindex,hyper=false]{#1}[]%
12301 }

```

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.

```

12302 \newcommand*{\glsxtrheadshortpl}[1]{%
12303 \protect\NoCaseChange
12304 {%
12305 \glsifattribute{#1}{headuc}{true}%
12306 {%
12307 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12308 }%
12309 {%
12310 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
12311 }%
12312 }%
12313 }

```

lsxtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

12314 \newrobustcmd*{\glsxtrtitleshortpl}[1]{%
12315 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
12316 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```
12317 \newcommand*{\Glsxtrheadshort}[1]{%
12318   \protect\NoCaseChange
12319   {%
12320     \glsifattribute{#1}{headuc}{true}%
12321     {%
12322       \Glsxtrshort[noindex,hyper=false]{#1}[]%
12323     }%
12324     {%
12325       \Glsxtrshort[noindex,hyper=false]{#1}[]%
12326     }%
12327   }%
12328 }
```

lSxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
12329 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
12330   \Glsxtrshort[noindex,hyper=false]{#1}[]%
12331 }
```

LSxtrtitleshort Command to display short form of abbreviation in section title and table of contents in all upper case.

```
12332 \newrobustcmd*{\GLSxtrtitleshort}[1]{%
12333   \GLSxtrshort[noindex,hyper=false]{#1}[]%
12334 }
```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```
12335 \newcommand*{\Glsxtrheadshortpl}[1]{%
12336   \protect\NoCaseChange
12337   {%
12338     \glsifattribute{#1}{headuc}{true}%
12339     {%
12340       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
12341     }%
12342     {%
12343       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
12344     }%
12345   }%
12346 }
```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
12347 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
12348   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
12349 }
```

`\xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents in all upper case.

```
12350 \newrobustcmd*{\GLSxtrtitleshortpl}[1]{%
12351   \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12352 }
```

`\glxtrheadname` As above but for the name value.

```
12353 \newcommand*{\glxtrheadname}[1]{%
12354   \protect\NoCaseChange
12355   {%
12356     \gl@ifattribute{#1}{headuc}{true}%
12357     {%
12358       \GLSname[noindex,hyper=false]{#1}[]%
12359     }%
12360     {%
12361       \glsname[noindex,hyper=false]{#1}[]%
12362     }%
12363   }%
12364 }
```

`\glxtrtitlename` Command to display name value in section title and table of contents.

```
12365 \newrobustcmd*{\glxtrtitlename}[1]{%
12366   \glsname[noindex,hyper=false]{#1}[]%
12367 }
```

`\Glsxtrheadname` First letter converted to upper case

```
12368 \newcommand*{\Glsxtrheadname}[1]{%
12369   \protect\NoCaseChange
12370   {%
12371     \gl@ifattribute{#1}{headuc}{true}%
12372     {%
12373       \GLSname[noindex,hyper=false]{#1}[]%
12374     }%
12375     {%
12376       \Glsname[noindex,hyper=false]{#1}[]%
12377     }%
12378   }%
12379 }
```

`\Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```
12380 \newrobustcmd*{\Glsxtrtitlename}[1]{%
12381   \Glsname[noindex,hyper=false]{#1}[]%
12382 }
```

`\GLSxtrtitlename` Command to display name value in section title and table of contents in all upper case.

```
12383 \newrobustcmd*{\GLSxtrtitlename}[1]{%
12384   \GLSname[noindex,hyper=false]{#1}[]%
12385 }
```

`\glsxtrheadtext` As above but for the text value.

```
12386 \newcommand*{\glsxtrheadtext}[1]{%
12387   \protect\NoCaseChange
12388   {%
12389     \glsifattribute{#1}{headuc}{true}%
12390     {%
12391       \GLStext[noindex,hyper=false]{#1}[]%
12392     }%
12393     {%
12394       \glstext[noindex,hyper=false]{#1}[]%
12395     }%
12396   }%
12397 }
```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
12398 \newrobustcmd*{\glsxtrtitletext}[1]{%
12399   \glstext[noindex,hyper=false]{#1}[]%
12400 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
12401 \newcommand*{\Glsxtrheadtext}[1]{%
12402   \protect\NoCaseChange
12403   {%
12404     \glsifattribute{#1}{headuc}{true}%
12405     {%
12406       \GLStext[noindex,hyper=false]{#1}[]%
12407     }%
12408     {%
12409       \Glstext[noindex,hyper=false]{#1}[]%
12410     }%
12411   }%
12412 }
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
12413 \newrobustcmd*{\Glsxtrtitletext}[1]{%
12414   \Glstext[noindex,hyper=false]{#1}[]%
12415 }
```

`GLSxtrtitletext` Command to display text value in section title and table of contents in all upper case.

```
12416 \newrobustcmd*{\GLSxtrtitletext}[1]{%
12417   \GLStext[noindex,hyper=false]{#1}[]%
12418 }
```

`lsxtrheadplural` As above but for the plural value.

```
12419 \newcommand*{\lsxtrheadplural}[1]{%
12420   \protect\NoCaseChange
12421   {%
```

```

12422 \glsifattribute{#1}{headuc}{true}%
12423 {%
12424 \GLSplural[noindex,hyper=false]{#1}[]%
12425 }%
12426 {%
12427 \glsplural[noindex,hyper=false]{#1}[]%
12428 }%
12429 }%
12430 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

12431 \newrobustcmd*{\glsxtrtitleplural}[1]{%
12432 \glsplural[noindex,hyper=false]{#1}[]%
12433 }

```

lsxtrheadplural Convert first letter to upper case.

```

12434 \newcommand*{\Glsxtrheadplural}[1]{%
12435 \protect\NoCaseChange
12436 {%
12437 \glsifattribute{#1}{headuc}{true}%
12438 {%
12439 \GLSplural[noindex,hyper=false]{#1}[]%
12440 }%
12441 {%
12442 \Glsplural[noindex,hyper=false]{#1}[]%
12443 }%
12444 }%
12445 }

```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

12446 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
12447 \Glsplural[noindex,hyper=false]{#1}[]%
12448 }

```

Sxtrtitleplural Command to display plural value in section title and table of contents in all upper case.

```

12449 \newrobustcmd*{\GLSxtrtitleplural}[1]{%
12450 \GLSplural[noindex,hyper=false]{#1}[]%
12451 }

```

glsxtrheadfirst As above but for the first value.

```

12452 \newcommand*{\glsxtrheadfirst}[1]{%
12453 \protect\NoCaseChange
12454 {%
12455 \glsifattribute{#1}{headuc}{true}%
12456 {%
12457 \GLSfirst[noindex,hyper=false]{#1}[]%
12458 }%

```

```

12459   {%
12460     \glsfirst[noindex,hyper=false]{#1}[]%
12461   }%
12462 }%
12463 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```

12464 \newrobustcmd*{\lsxtrtitlefirst}[1]{%
12465   \glsfirst[noindex,hyper=false]{#1}[]%
12466 }

```

Glsxtrheadfirst First letter converted to upper case

```

12467 \newcommand*{\Glsxtrheadfirst}[1]{%
12468   \protect\NoCaseChange
12469   {%
12470     \glsifattribute{#1}{headuc}{true}%
12471     {%
12472       \GLSfirst[noindex,hyper=false]{#1}[]%
12473     }%
12474     {%
12475       \Glsfirst[noindex,hyper=false]{#1}[]%
12476     }%
12477   }%
12478 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

12479 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
12480   \Glsfirst[noindex,hyper=false]{#1}[]%
12481 }

```

LSxtrtitlefirst Command to display first value in section title and table of contents in all upper case.

```

12482 \newrobustcmd*{\GLSxtrtitlefirst}[1]{%
12483   \GLSfirst[noindex,hyper=false]{#1}[]%
12484 }

```

headfirstplural As above but for the firstplural value.

```

12485 \newcommand*{\glsxtrheadfirstplural}[1]{%
12486   \protect\NoCaseChange
12487   {%
12488     \glsifattribute{#1}{headuc}{true}%
12489     {%
12490       \GLSfirstplural[noindex,hyper=false]{#1}[]%
12491     }%
12492     {%
12493       \glsfirstplural[noindex,hyper=false]{#1}[]%
12494     }%
12495   }%
12496 }

```

`itlefirstplural` Command to display firstplural value in section title and table of contents.

```
12497 \newrobustcmd*{\glxstrtitlefirstplural}[1]{%
12498   \glsfirstplural[noindex,hyper=false]{#1}[]%
12499 }
```

`headfirstplural` First letter converted to upper case

```
12500 \newcommand*{\Glsxtrheadfirstplural}[1]{%
12501   \protect\NoCaseChange
12502   {%
12503     \glusifattribute{#1}{headuc}{true}%
12504     {%
12505       \GLSfirstplural[noindex,hyper=false]{#1}[]%
12506     }%
12507     {%
12508       \Glsfirstplural[noindex,hyper=false]{#1}[]%
12509     }%
12510   }%
12511 }
```

`itlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
12512 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
12513   \Glsfirstplural[noindex,hyper=false]{#1}[]%
12514 }
```

`itlefirstplural` Command to display first value in section title and table of contents in all upper case.

```
12515 \newrobustcmd*{\GLSxtrtitlefirstplural}[1]{%
12516   \GLSfirstplural[noindex,hyper=false]{#1}[]%
12517 }
```

`\glxstrheadlong` Command used to display long form in the page header.

```
12518 \newcommand*{\glxstrheadlong}[1]{%
12519   \protect\NoCaseChange
12520   {%
12521     \glusifattribute{#1}{headuc}{true}%
12522     {%
12523       \GLSxtrlong[noindex,hyper=false]{#1}[]%
12524     }%
12525     {%
12526       \glsxtrlong[noindex,hyper=false]{#1}[]%
12527     }%
12528   }%
12529 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
12530 \newrobustcmd*{\glxstrtitlelong}[1]{%
12531   \glsxtrlong[noindex,hyper=false]{#1}[]%
12532 }
```


`\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.

```

12533 \newcommand*{\glxtrheadlongpl}[1]{%
12534   \protect\NoCaseChange
12535   {%
12536     \gl@ifattribute{#1}{headuc}{true}%
12537     {%
12538       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12539     }%
12540     {%
12541       \glxtrlongpl[noindex,hyper=false]{#1}[]%
12542     }%
12543   }%
12544 }
```

`\sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

12545 \newrobustcmd*{\glxtrtitlelongpl}[1]{%
12546   \glxtrlongpl[noindex,hyper=false]{#1}[]%
12547 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

12548 \newcommand*{\Glsxtrheadlong}[1]{%
12549   \protect\NoCaseChange
12550   {%
12551     \gl@ifattribute{#1}{headuc}{true}%
12552     {%
12553       \GLSxtrlong[noindex,hyper=false]{#1}[]%
12554     }%
12555     {%
12556       \Glsxtrlong[noindex,hyper=false]{#1}[]%
12557     }%
12558   }%
12559 }
```

`\Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12560 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
12561   \Glsxtrlong[noindex,hyper=false]{#1}[]%
12562 }
```

`\GLSxtrtitlelong` Command to display long form of abbreviation in section title and table of contents in all upper case.

```

12563 \newrobustcmd*{\GLSxtrtitlelong}[1]{%
12564   \GLSxtrlong[noindex,hyper=false]{#1}[]%
12565 }
```

`\lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
12566 \newcommand*{\Glsxtrheadlongpl}[1]{%
12567   \protect\NoCaseChange
12568   {%
12569     \glsifattribute{#1}{headuc}{true}%
12570     {%
12571       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12572     }%
12573     {%
12574       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
12575     }%
12576   }%
12577 }
```

`\sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
12578 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
12579   \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12580 }
```

`Sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents in all upper case.

```
12581 \newrobustcmd*{\GLSxtrtitlelongpl}[1]{%
12582   \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12583 }
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
12584 \newcommand*{\glsxtrheadfull}[1]{%
12585   \protect\NoCaseChange
12586   {%
12587     \glsifattribute{#1}{headuc}{true}%
12588     {%
12589       \GLSxtrfull[noindex,hyper=false]{#1}[]%
12590     }%
12591     {%
12592       \glsxtrfull[noindex,hyper=false]{#1}[]%
12593     }%
12594   }%
12595 }
```

`\glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
12596 \newrobustcmd*{\glsxtrtitlefull}[1]{%
12597   \glsxtrfull[noindex,hyper=false]{#1}[]%
12598 }
```

`\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.

```

12599 \newcommand*{\glxstrheadfullpl}[1]{%
12600   \protect\NoCaseChange
12601   {%
12602     \glusifattribute{#1}{headuc}{true}%
12603     {%
12604       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12605     }%
12606     {%
12607       \glxstrfullpl[noindex,hyper=false]{#1}[]%
12608     }%
12609   }%
12610 }

```

sxstrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

12611 \newrobustcmd*{\glxstrtitlefullpl}[1]{%
12612   \glxstrfullpl[noindex,hyper=false]{#1}[]%
12613 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

12614 \newcommand*{\Glsxtrheadfull}[1]{%
12615   \protect\NoCaseChange
12616   {%
12617     \glusifattribute{#1}{headuc}{true}%
12618     {%
12619       \GLSxtrfull[noindex,hyper=false]{#1}[]%
12620     }%
12621     {%
12622       \Glsxtrfull[noindex,hyper=false]{#1}[]%
12623     }%
12624   }%
12625 }

```

Glsxstrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12626 \newrobustcmd*{\Glsxstrtitlefull}[1]{%
12627   \Glsxtrfull[noindex,hyper=false]{#1}[]%
12628 }

```

GLSxstrtitlefull Command to display full form of abbreviation in section title and table of contents in all upper case.

```

12629 \newrobustcmd*{\GLSxstrtitlefull}[1]{%
12630   \GLSxtrfull[noindex,hyper=false]{#1}[]%
12631 }

```

lxsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

12632 \newcommand*{\Glsxtrheadfullpl}[1]{%

```

```

12633 \protect\NoCaseChange
12634 {%
12635   \glsifattribute{#1}{headuc}{true}%
12636   {%
12637     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12638   }%
12639   {%
12640     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
12641   }%
12642 }%
12643 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12644 \newrobustcmd*{\GLsxtrtitlefullpl}[1]{%
12645   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
12646 }

```

`Sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents in all upper case.

```

12647 \newrobustcmd*{\GLSxtrtitlefullpl}[1]{%
12648   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12649 }

```

`\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.

```

12650 \ifdef\texorpdfstring
12651 {
12652   \newcommand*{\glsfmtshort}[1]{%
12653     \texorpdfstring
12654       {\glsxtrtitleshort{#1}}%
12655       {\glsentryshort{#1}}%
12656   }
12657 }
12658 {
12659   \newcommand*{\glsfmtshort}[1]{%
12660     \glsxtrtitleshort{#1}}
12661 }

```

Similarly for the plural version.

```

\glsfmtshortpl
12662 \ifdef\texorpdfstring
12663 {
12664   \newcommand*{\glsfmtshortpl}[1]{%
12665     \texorpdfstring
12666       {\glsxtrtitleshortpl{#1}}%
12667       {\glsentryshortpl{#1}}%
12668   }

```

```

12669 }
12670 {
12671   \newcommand*{\glsfmtshortpl}[1]{%
12672     \glsxtrtitleshortpl{#1}}
12673 }

```

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).

```

12674 \ifdef\texorpdfstring
12675 {
12676   \newcommand*{\Glsfmtshort}[1]{%
12677     \texorpdfstring
12678       {\Glsxtrtitleshort{#1}}%
12679     {\glsentryshort{#1}}%
12680   }
12681 }
12682 {
12683   \newcommand*{\Glsfmtshort}[1]{%
12684     \Glsxtrtitleshort{#1}}
12685 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

12686 \ifdef\texorpdfstring
12687 {
12688   \newcommand*{\Glsfmtshortpl}[1]{%
12689     \texorpdfstring
12690       {\Glsxtrtitleshortpl{#1}}%
12691     {\glsentryshortpl{#1}}%
12692   }
12693 }
12694 {
12695   \newcommand*{\Glsfmtshortpl}[1]{%
12696     \Glsxtrtitleshortpl{#1}}
12697 }

```

`\glsfmtname` As above but for the name value.

```

12698 \ifdef\texorpdfstring
12699 {
12700   \newcommand*{\glsfmtname}[1]{%
12701     \texorpdfstring
12702       {\glsxtrtitlename{#1}}%
12703     {\glsentryname{#1}}%
12704   }
12705 }
12706 {
12707   \newcommand*{\glsfmtname}[1]{%
12708     \glsxtrtitlename{#1}}
12709 }

```

`\Glsfmtname` First letter converted to upper case.

```
12710 \ifdef\texorpdfstring
12711 {
12712   \newcommand*\Glsfmtname[1]{%
12713     \texorpdfstring
12714     {\Glsxtrtitlename{#1}}%
12715     {\glsentryname{#1}}%
12716   }
12717 }
12718 {
12719   \newcommand*\Glsfmtname[1]{%
12720     \Glsxtrtitlename{#1}}
12721 }
```

`\GLSfmtname` All upper case.

```
12722 \ifdef\texorpdfstring
12723 {
12724   \newcommand*\GLSfmtname[1]{%
12725     \texorpdfstring
12726     {\GLSxtrtitlename{#1}}%
12727     {\glsentryname{#1}}%
12728   }
12729 }
12730 {
12731   \newcommand*\GLSfmtname[1]{%
12732     \GLSxtrtitlename{#1}}
12733 }
```

`\glsfmttext` As above but for the text value.

```
12734 \ifdef\texorpdfstring
12735 {
12736   \newcommand*\glsfmttext[1]{%
12737     \texorpdfstring
12738     {\glsxtrtitletext{#1}}%
12739     {\glsentrytext{#1}}%
12740   }
12741 }
12742 {
12743   \newcommand*\glsfmttext[1]{%
12744     \glsxtrtitletext{#1}}
12745 }
```

`\Glsfmttext` First letter converted to upper case.

```
12746 \ifdef\texorpdfstring
12747 {
12748   \newcommand*\Glsfmttext[1]{%
12749     \texorpdfstring
12750     {\Glsxtrtitletext{#1}}%
12751     {\glsentrytext{#1}}%
```

```

12752 }
12753 }
12754 {
12755   \newcommand*{\GLsfmtext}[1]{%
12756     \GLsxtrtitletext{#1}}
12757 }

```

`\GLsfmtext` All upper case.

```

12758 \ifdef\teorpdfstring
12759 {
12760   \newcommand*{\GLsfmtext}[1]{%
12761     \teorpdfstring
12762       {\GLsxtrtitletext{#1}}%
12763       {\glstrytext{#1}}%
12764   }
12765 }
12766 {
12767   \newcommand*{\GLsfmtext}[1]{%
12768     \GLsxtrtitletext{#1}}
12769 }

```

`\glsfmplural` As above but for the plural value.

```

12770 \ifdef\teorpdfstring
12771 {
12772   \newcommand*{\glsfmplural}[1]{%
12773     \teorpdfstring
12774       {\glsxtrtitleplural{#1}}%
12775       {\glstryplural{#1}}%
12776   }
12777 }
12778 {
12779   \newcommand*{\glsfmplural}[1]{%
12780     \glsxtrtitleplural{#1}}
12781 }

```

`\Glsfmplural` First letter converted to upper case.

```

12782 \ifdef\teorpdfstring
12783 {
12784   \newcommand*{\Glsfmplural}[1]{%
12785     \teorpdfstring
12786       {\Glsxtrtitleplural{#1}}%
12787       {\glstryplural{#1}}%
12788   }
12789 }
12790 {
12791   \newcommand*{\Glsfmplural}[1]{%
12792     \Glsxtrtitleplural{#1}}
12793 }

```

`\GLSfmtplural` All upper case.

```
12794 \ifdef\txorpdfstring
12795 {
12796   \newcommand*{\GLSfmtplural}[1]{%
12797     \txorpdfstring
12798     {\GLSxtrtitleplural{#1}}%
12799     {\glentryplural{#1}}%
12800   }
12801 }
12802 {
12803   \newcommand*{\GLSfmtplural}[1]{%
12804     \GLSxtrtitleplural{#1}}
12805 }
```

`\glsfmtfirst` As above but for the first value.

```
12806 \ifdef\txorpdfstring
12807 {
12808   \newcommand*{\glsfmtfirst}[1]{%
12809     \txorpdfstring
12810     {\glsxtrtitlefirst{#1}}%
12811     {\glentryfirst{#1}}%
12812   }
12813 }
12814 {
12815   \newcommand*{\glsfmtfirst}[1]{%
12816     \glsxtrtitlefirst{#1}}
12817 }
```

`\Glsfmtfirst` First letter converted to upper case.

```
12818 \ifdef\txorpdfstring
12819 {
12820   \newcommand*{\Glsfmtfirst}[1]{%
12821     \txorpdfstring
12822     {\Glsxtrtitlefirst{#1}}%
12823     {\glentryfirst{#1}}%
12824   }
12825 }
12826 {
12827   \newcommand*{\Glsfmtfirst}[1]{%
12828     \Glsxtrtitlefirst{#1}}
12829 }
```

`\GLSfmtfirst` All upper case.

```
12830 \ifdef\txorpdfstring
12831 {
12832   \newcommand*{\GLSfmtfirst}[1]{%
12833     \txorpdfstring
12834     {\GLSxtrtitlefirst{#1}}%
12835     {\glentryfirst{#1}}%
```



```

12836 }
12837 }
12838 {
12839   \newcommand*{\GLSfmtfirst}[1]{%
12840     \Glsxtrtitlefirst{#1}}
12841 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

12842 \ifdef\teorpdfstring
12843 {
12844   \newcommand*{\glsfmtfirstpl}[1]{%
12845     \teorpdfstring
12846       {\Glsxtrtitlefirstplural{#1}}%
12847       {\glsentryfirstplural{#1}}%
12848   }
12849 }
12850 {
12851   \newcommand*{\glsfmtfirstpl}[1]{%
12852     \Glsxtrtitlefirstplural{#1}}
12853 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

12854 \ifdef\teorpdfstring
12855 {
12856   \newcommand*{\Glsfmtfirstpl}[1]{%
12857     \teorpdfstring
12858       {\Glsxtrtitlefirstplural{#1}}%
12859       {\glsentryfirstplural{#1}}%
12860   }
12861 }
12862 {
12863   \newcommand*{\Glsfmtfirstpl}[1]{%
12864     \Glsxtrtitlefirstplural{#1}}
12865 }

```

`\GLSfmtfirstpl` All upper case.

```

12866 \ifdef\teorpdfstring
12867 {
12868   \newcommand*{\GLSfmtfirstpl}[1]{%
12869     \teorpdfstring
12870       {\GLSxtrtitlefirstplural{#1}}%
12871       {\glsentryfirstplural{#1}}%
12872   }
12873 }
12874 {
12875   \newcommand*{\GLSfmtfirstpl}[1]{%
12876     \GLSxtrtitlefirstplural{#1}}
12877 }

```

`\glsfmtlong` As above but for the long value.

```
12878 \ifdef\txorpdfstring
12879 {
12880   \newcommand*\glsfmtlong[1]{%
12881     \txorpdfstring
12882     {\glsxtrtitlelong{#1}}%
12883     {\glsentrylong{#1}}%
12884   }
12885 }
12886 {
12887   \newcommand*\glsfmtlong[1]{%
12888     \glsxtrtitlelong{#1}}
12889 }
```

`\Glsfmtlong` First letter converted to upper case.

```
12890 \ifdef\txorpdfstring
12891 {
12892   \newcommand*\Glsfmtlong[1]{%
12893     \txorpdfstring
12894     {\Glsxtrtitlelong{#1}}%
12895     {\glsentrylong{#1}}%
12896   }
12897 }
12898 {
12899   \newcommand*\Glsfmtlong[1]{%
12900     \Glsxtrtitlelong{#1}}
12901 }
```

`\GLSfmtlong` All upper case.

```
12902 \ifdef\txorpdfstring
12903 {
12904   \newcommand*\GLSfmtlong[1]{%
12905     \txorpdfstring
12906     {\GLSxtrtitlelong{#1}}%
12907     {\glsentrylong{#1}}%
12908   }
12909 }
12910 {
12911   \newcommand*\GLSfmtlong[1]{%
12912     \GLSxtrtitlelong{#1}}
12913 }
```

`\glsfmtlongpl` As above but for the longplural value.

```
12914 \ifdef\txorpdfstring
12915 {
12916   \newcommand*\glsfmtlongpl[1]{%
12917     \txorpdfstring
12918     {\glsxtrtitlelongpl{#1}}%
12919     {\glsentrylongpl{#1}}%
12920   }
12921 }
```

```

12920 }
12921 }
12922 {
12923   \newcommand*{\glsfmtlongpl}[1]{%
12924     \glsxtrtitlelongpl{#1}}
12925 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

12926 \ifdef\teorpdfstring
12927 {
12928   \newcommand*{\Glsfmtlongpl}[1]{%
12929     \teorpdfstring
12930     {\Glsxtrtitlelongpl{#1}}%
12931     {\glsentrylongpl{#1}}%
12932   }
12933 }
12934 {
12935   \newcommand*{\Glsfmtlongpl}[1]{%
12936     \Glsxtrtitlelongpl{#1}}
12937 }

```

`\GLSfmtlongpl` All upper case.

```

12938 \ifdef\teorpdfstring
12939 {
12940   \newcommand*{\GLSfmtlongpl}[1]{%
12941     \teorpdfstring
12942     {\GLSxtrtitlelongpl{#1}}%
12943     {\glsentrylongpl{#1}}%
12944   }
12945 }
12946 {
12947   \newcommand*{\GLSfmtlongpl}[1]{%
12948     \GLSxtrtitlelongpl{#1}}
12949 }

```

`\glspdffmtfull` Can't use `\glsxtrinlinefullformat` in PDF bookmarks as it's not fully expandable. This command is for the PDF part of `\teorpdfstring` for the full form.

```

12950 \newcommand*{\glspdffmtfull}[1]{\glsentrylong{#1} (\glsentryshort{#1})}%

```

`glspdffmtfullpl` Likewise for plural.

```

12951 \newcommand*{\glspdffmtfullpl}[1]{\glsentrylongpl{#1} (\glsentryshortpl{#1})}%

```

`\glsfmtfull` In-line full format.

```

12952 \ifdef\teorpdfstring
12953 {
12954   \newcommand*{\glsfmtfull}[1]{%
12955     \teorpdfstring
12956     {\glsxtrtitlefull{#1}}%

```

```

12957     {\glspdffmtfull{#1}}%
12958   }
12959 }
12960 {
12961   \newcommand*{\Glsfmtfull}[1]{%
12962     \glstrtitlefull{#1}}
12963 }

```

`\Glsfmtfull` First letter converted to upper case.

```

12964 \ifdef\teorpdfstring
12965 {
12966   \newcommand*{\Glsfmtfull}[1]{%
12967     \teorpdfstring
12968     {\Glsxtrtitlefull{#1}}%
12969     {\glspdffmtfull{#1}}}%
12970   }
12971 }
12972 {
12973   \newcommand*{\Glsfmtfull}[1]{%
12974     \Glsxtrtitlefull{#1}}
12975 }

```

`\GLSfmtfull` All upper case.

```

12976 \ifdef\teorpdfstring
12977 {
12978   \newcommand*{\GLSfmtfull}[1]{%
12979     \teorpdfstring
12980     {\GLSxtrtitlefull{#1}}%
12981     {\glspdffmtfull{#1}}}%
12982   }
12983 }
12984 {
12985   \newcommand*{\GLSfmtfull}[1]{%
12986     \GLSxtrtitlefull{#1}}
12987 }

```

`\glsfmtfullpl` In-line full plural format.

```

12988 \ifdef\teorpdfstring
12989 {
12990   \newcommand*{\glsfmtfullpl}[1]{%
12991     \teorpdfstring
12992     {\glsxtrtitlefullpl{#1}}%
12993     {\glspdffmtfullpl{#1}}}%
12994   }
12995 }
12996 {
12997   \newcommand*{\glsfmtfullpl}[1]{%
12998     \glsxtrtitlefullpl{#1}}
12999 }

```

```

\Glsfmtfullpl First letter converted to upper case.
13000 \ifdef\texorpdfstring
13001 {
13002   \newcommand*\Glsfmtfullpl[1]{%
13003     \texorpdfstring
13004     {\Glsxtrtitlefullpl{#1}}%
13005     {\glspdfdfmtfullpl{#1}{}}%
13006   }
13007 }
13008 {
13009   \newcommand*\Glsfmtfullpl[1]{%
13010     \Glsxtrtitlefullpl{#1}}
13011 }

```

```

\GLSfmtfullpl All upper case.
13012 \ifdef\texorpdfstring
13013 {
13014   \newcommand*\GLSfmtfullpl[1]{%
13015     \texorpdfstring
13016     {\GLSxtrtitlefullpl{#1}}%
13017     {\glspdfdfmtfullpl{#1}{}}%
13018   }
13019 }
13020 {
13021   \newcommand*\GLSfmtfullpl[1]{%
13022     \GLSxtrtitlefullpl{#1}}
13023 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

```

seriesExtraLang
13024 \newcommand*\RequireGlossariesExtraLang[1]{%
13025   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
13026 }

```

```

seriesExtraLang
13027 \newcommand*\ProvidesGlossariesExtraLang[1]{%
13028   \ProvidesFile{glossariesxtr-#1.ldf}%
13029 }

```

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.

```
13030 \newcommand{\glxtr@loaddialect}{%
13031   \IfTrackedLanguageFileExists{\this@dialect}%
13032   {\glossariesxtr-}% prefix
13033   {\ldf}%
13034   {%
13035     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
13036   }%
13037   {}% not found
```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```
13038 \@glxtrdialecthook
13039 }
```

```
13040 \@ifpackageloaded{tracklang}
13041 {%
13042   \AnyTrackedLanguages
13043   {%
13044     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
13045   }%
13046   {}%
13047 }
13048 {}
```

Load `glossaries-extra-stylemods` if required.

```
13049 \@glxtr@redefstyles
```

and set the style:

```
13050 \@glxtr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```
13051 \NeedsTeXFormat{LaTeX2e}
13052 \ProvidesPackage{glossaries-extra-bib2gls}[2020/03/23 v1.44 (NLCT)]
```

Provide convenient shortcut commands for predefined glossary types.

`printunsrtacronyms`

```
13053 \ifglssacronym
13054   \providecommand*\printunsrtacronyms[1][\acronymtype]{%
13055     \printunsrtglossary[type=\acronymtype,#1]}%
13056 \fi
```

`printunsrtindex`

```
13057 \ifglossaryexists{index}
13058 {
```

```

13059 \providecommand*\printunsrtindex}[1][]{%
13060 \printunsrtglossary[type=index,#1]}%
13061 }{}

```

intunsrtsymbols

```

13062 \ifglossaryexists{symbols}
13063 {
13064 \providecommand*\printunsrtsymbols}[1][]{%
13065 \printunsrtglossary[type=symbols,#1]}%
13066 }{}

```

intunsrtnumbers

```

13067 \ifglossaryexists{numbers}
13068 {
13069 \providecommand*\printunsrtnumbers}[1][]{%
13070 \printunsrtglossary[type=numbers,#1]}%
13071 }{}

```

rtaabbreviations

```

13072 \ifglossaryexists{abbreviations}
13073 {
13074 \providecommand*\printunsrtaabbreviations}[1][]{%
13075 \printunsrtglossary[type=abbreviations,#1]}%
13076 }{}

```

splaynumberlist Allow \glsdisplaynumberlist and make it robust.

```

13077 \renewcommand*\glsdisplaynumberlist}[1]{%
13078 \glsdoifexists{#1}%
13079 {%
13080 {\let\bibglsglN\glsnumlistsep
13081 \let\bibglsglN\glsnumlistlastsep
13082 \glsxtrusefield{#1}{location}%
13083 }%
13084 }%
13085 }
13086 \robustify\glsdisplaynumberlist

```

entrynumberlist

```

13087 \renewcommand*\glsentrynumberlist}[1]{\glsxtrusefield{#1}{location}}

```

These are some convenient macros for use with custom rules.

\glshex

```

13088 \newcommand*\glshex{\string\u}

```

lscapturedgroup

```

13089 \newcommand*\glsapturedgroup{\string\$}

```

nZeroChildCount For use with bib2gls's save-child-count resource option.

```
13090 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
13091   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
13092 }
```

rprovidecommand For use in @preamble, this behaves like \providecommand in the document but like \renewcommand in bib2gls.

```
13093 \newcommand*{\glstrprovidecommand}{\providecommand}
```

glsrenewcommand Like \renewcommand but only generates a warning rather than an error if the command isn't defined.

```
13094 \newcommand*{\glstrenewcommand}{\@star@or@long\glstr@renewcommand}
```

tr@renewcommand

```
13095 \newcommand*{\glstr@renewcommand}[1]{%
13096   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
13097   \expandafter\@ifundefined\@gtempa
13098     {%
13099       \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
13100     }%
13101   \relax
13102   \relax
13103   \let\@ifdefinable\@rc@ifdefinable
13104   \new@command#1%
13105 }
```

lossarylocation For use with indexcounter and bib2gls.

```
13106 \newcommand*{\glstr@wrglossarylocation}[2]{#1}
```

indexCounterLink

`\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```
13107 \ifdef\hyperref
13108 {%
13109   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
13110     \glstrifhasfield{indexcounter}{#2}%
13111     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
13112     {#1}%
13113   }
13114 }
13115 {
13116   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
13117 }
```


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.

```
13118 \newcommand*{\GlsXtrDualField}{dual}
```

XtrDualBackLink

```
\GlsXtrDualBackLink{<text>}{<label>}
```

Adds a hyperlink to the dual entry.

```
13119 \newcommand*{\GlsXtrDualBackLink}[2]{%
13120   \glstrifhasfield{\GlsXtrDualField}{#2}%
13121   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
13122   {#2}%
13123 }
```

TeXEntryAliases Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
13124 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
13125   article=bibtexentry,
13126   book=bibtexentry,
13127   booklet=bibtexentry,
13128   conference=bibtexentry,
13129   inbook=bibtexentry,
13130   incollection=bibtexentry,
13131   inproceedings=bibtexentry,
13132   manual=bibtexentry,
13133   mastersthesis=bibtexentry,
13134   misc=bibtexentry,
13135   phdthesis=bibtexentry,
13136   proceedings=bibtexentry,
13137   techreport=bibtexentry,
13138   unpublished=bibtexentry
13139 }
```

ideBibTeXFields Convenient shortcut to define the standard BibTeX fields.

```
13140 \newcommand*{\GlsXtrProvideBibTeXFields}{%
13141   \glsaddstoragekey{address}{\glstrbibaddress}%
13142   \glsaddstoragekey{author}{\glstrbibauthor}%
13143   \glsaddstoragekey{booktitle}{\glstrbibbooktitle}%
13144   \glsaddstoragekey{chapter}{\glstrbibchapter}%
13145   \glsaddstoragekey{edition}{\glstrbibedition}%
13146   \glsaddstoragekey{howpublished}{\glstrbibhowpublished}%
13147   \glsaddstoragekey{institution}{\glstrbibinstitution}%

```

```

13148 \glsaddstoragekey{journal}{\glsxtrbibjournal}%
13149 \glsaddstoragekey{month}{\glsxtrbibmonth}%
13150 \glsaddstoragekey{note}{\glsxtrbibnote}%
13151 \glsaddstoragekey{number}{\glsxtrbibnumber}%
13152 \glsaddstoragekey{organization}{\glsxtrbiborganization}%
13153 \glsaddstoragekey{pages}{\glsxtrbibpages}%
13154 \glsaddstoragekey{publisher}{\glsxtrbibpublisher}%
13155 \glsaddstoragekey{school}{\glsxtrbibschooll}%
13156 \glsaddstoragekey{series}{\glsxtrbibseries}%
13157 \glsaddstoragekey{title}{\glsxtrbibtitle}%
13158 \glsaddstoragekey{bibtextype}{\glsxtrbibtype}%
13159 \glsaddstoragekey{volume}{\glsxtrbibvolume}%
13160 }

```

Multiple supplementary references are only supported with bib2gls.

`\glstisuppllocation` This is like `\glsxtrsupphypernumber` but the second argument is the external file name (which isn't obtained from the `externallocation` attribute). The third argument is the formatting (encap) control sequence *name*. This is ignored by default, but is set by `bib2gls` to the original encap in case it's required.

```

13161 \newcommand*\glsxtrmultisuppllocation[3]{%
13162   {%
13163     \def\glsxtrsuppllocationurl{#2}%
13164     \glshypernumber{#1}%
13165   }%
13166 }

```

`\glstrdisplaysupplloc`

`\glstrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```

13167 \newcommand*\glstrdisplaysupplloc[5]{%
13168   \setentrycounter[#1]{#2}%
13169   \glsxtrmultisuppllocation{#5}{#4}{#3}%
13170 }

```

`\glstrdisplaylocnameref` `\glstrdisplaylocnameref{<prefix>}{<counter>}{<format>}{<location>}{<name>}{<href>}{<hcounter>}{<external file>}` Used with the `[nameref]` record package option. The `<href>` argument was obtained from `\@currentHref` and the `<hcounter>` argument was obtained from `\theHentrycounter`, which is more reliable. If `hyperref` hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```

13171 \ifundef\hyperlink
13172 {
13173   \newcommand*\glstrdisplaylocnameref[8]{%
13174     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%

```

```

13175 }
13176 }
13177 {

```

Default action uses $\langle hcounter \rangle$. Equations and pages typically don't have a title, so check the counter name (otherwise the title may section or chapter title, which may be confusing). As from v1.42, this now checks if the control sequence $\backslash\mathrm{glstr}\langle counter \rangle\mathrm{locfmt}$ is defined.

```

13178 \newcommand*\glstrdisplaylocnameref}[8]{%
13179   \ifcsdef{glstr#2locfmt}%
13180     {\glstrnameref link{#3}{\csuse{glstr#2locfmt}{#4}{#5}}{#2.#7}{#8}}%
13181     {%
13182       \ifstrempy{#5}%
13183       {%

```

No title, so just use the location as the link text.

```

13184     \glstrnameref link{#3}{#4}{#2.#7}{#8}%
13185     }%
13186     {%
13187       \ifstrequal{#2}{page}%
13188       {\glstrnameref link{#3}{#4}{#2.#7}{#8}}%
13189       {\glstrnameref link{#3}{#5}{#2.#7}{#8}}%
13190     }%
13191   }%
13192 }
13193 }

```

re equationlocfmt

```
\glstrequationlocfmt{\langle location \rangle}{\langle title \rangle}
```

```

13194 \newcommand*\glstrequationlocfmt}[2]{(\#1)}

```

sxtrnameref link

```
\glstrfmt nameref link{\langle format \rangle}{\langle title \rangle}{\langle href \rangle}{\langle external file \rangle}
```

```

13195 \newcommand*\glstrnameref link}[4]{%

```

Locally change $\backslash\mathrm{glshypernumber}$ to $\backslash\mathrm{@firstofone}$ to remove the normal location hyperlink.

```

13196 \begingroup
13197 \let\glshypernumber\@firstofone

```

If the $\langle external file \rangle$ argument is empty, an internal link is used, otherwise an external one is needed.

```

13198   \ifstrempy{#4}%
13199   {\glstrfmt internalnameref{#3}{#1}{#2}}%

```

```

13200     {\glxtrfmtexternalnameref{#3}{#1}{#2}{#4}}%
13201   \endgroup
13202 }

```

sxtrnameloclink

```

\glxtrnamerefloclink{<prefix>}{<counter>}{<format>}{<location>}{<text>}{<external file>}

```

Like `\@gls@numberlink`, this creates a hyperlink to the target obtained from the prefix, counter and location but uses `<text>` as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

```

13203 \newcommand{\glxtrnameloclink}[6]{%
13204   \begingroup
13205   \setentrycounter[#1]{#2}%
13206   \def\glxtr@locationhypertext{#5}%
13207   \let\glshypernumber\@firstofone
13208   \def\@glsnumberformat{#3}%
13209   \def\glxtrsupplocationurl{#6}%
13210   \toks@={}%
13211   \@glxtr@bibgls@removespaces#4 \@nil
13212 \endgroup
13213 }

```

ls@removespaces

```

13214 \def\@glxtr@bibgls@removespaces#1 #2\@nil{%
13215   \toks@=\expandafter{\the\toks@#1}%
13216   \ifx\#2\%
13217     \edef\x{\the\toks@}%
13218     \ifx\x\empty
13219       \else
13220         \protected@edef\x{\glstentrycounter\@gls@counterprefix\the\toks@}%
13221         \ifdefvoid\glxtrsupplocationurl
13222           {%
13223             \expandafter\glxtrfmtinternalnameref\expandafter{\x}%
13224             {\@glsnumberformat}{\glxtr@locationhypertext}%
13225           }%
13226           {%
13227             \expandafter\glxtrfmtexternalnameref\expandafter{\x}%
13228             {\@glsnumberformat}{\glxtr@locationhypertext}{\glxtrsupplocationurl}%
13229           }%
13230         \fi
13231       \else
13232         \@gls@ReturnAfterFi{%
13233           \@glxtr@bibgls@removespaces#2\@nil
13234         }%
13235       \fi
13236 }

```

internalnameref

```
\glstrfmtinternalnameloc{<target>}{<format>}{<title>}
```

```
13237 \newcommand*{\glstrfmtinternalnameref}[3]{%
13238   \csuse{#2}{\glsdohyperlink{#1}{#3}}%
13239 }
```

externalnameref

```
\glstrfmtexternalnameloc{<target>}{<format>}{<title>}{<file>}
```

```
13240 \newcommand*{\glstrfmtexternalnameref}[4]{%
13241   \csuse{#2}{\hyperref{#4}{#1}{#3}}%
13242 }
```

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.

```
13243 \newcommand*{\glstrSetWidest}[3]{%
```

Check which style options have been provided. (The style packages may not have been loaded.)

```
13244   \ifdef\glupdatewidest
13245   {%
13246     \ifdef\glslongextraUpdateWidest
13247     {%
```

Relevant style packages all loaded. If the *<type>* has been given, append to glossary preamble.

```
13248     \ifstrempy{#1}
13249     {%
13250       \glupdatewidest[#2]{#3}%
13251       \ifnum#2=0\relax
13252       \glslongextraUpdateWidest{#3}%
13253       \else
13254       \glslongextraUpdateWidestChild{#2}{#3}%
13255       \fi
13256     }%
13257   {%
13258     \apptoglossarypreamble[#1]{\glupdatewidest[#2]{#3}}%
13259     \ifnum#2=0\relax
13260     \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
13261   \else
```

```

13262         \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
13263     \fi
13264 }%
13265 }%
13266 {%

```

Only alttree.

```

13267     \ifstrempy{#1}
13268     {%
13269         \glsupdatewidest[#2]{#3}%
13270     }%
13271     {%
13272         \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
13273     }%
13274 }%
13275 }%
13276 {%

```

\glsupdatewidest hasn't been defined. This could just mean that the glossaries-extra-stylemods package hasn't been loaded.

```

13277     \ifdef\glssetwidest
13278     {%
13279         \ifdef\glslongextraUpdateWidest
13280         {%

```

Relevant glossary-tree and glossary-longextra have been loaded. If the *<type>* has been given, append to glossary preamble.

```

13281     \ifstrempy{#1}
13282     {%
13283         \glssetwidest[#2]{#3}%
13284         \ifnum#2=0\relax
13285             \glslongextraUpdateWidest{#3}%
13286         \else
13287             \glslongextraUpdateWidestChild{#2}{#3}%
13288         \fi
13289     }%
13290     {%
13291         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
13292         \ifnum#2=0\relax
13293             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
13294         \else
13295             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
13296         \fi
13297     }%
13298 }%
13299 {%

```

Only alttree.

```

13300     \ifstrempy{#1}
13301     {%
13302         \glssetwidest[#2]{#3}%

```

```

13303     }%
13304     {%
13305         \apptoglossary preamble[#1]{\glssetwidest[#2]{#3}}%
13306     }%
13307 }%
13308 }%
13309 {%
13310     \ifdef\gls longextraUpdateWidest
13311     {%
        glossary-longextra has been loaded.
13312         \ifstrempy{#1}
13313         {%
13314             \ifnum#2=0\relax
13315                 \gls longextraUpdateWidest{#3}%
13316             \else
13317                 \gls longextraUpdateWidestChild{#2}{#3}%
13318             \fi
13319         }%
13320         {%
13321             \ifnum#2=0\relax
13322                 \apptoglossary preamble[#1]{\gls longextraUpdateWidest{#3}}%
13323             \else
13324                 \apptoglossary preamble[#1]{\gls longextraUpdateWidestChild{#2}{#3}}%
13325             \fi
13326         }%
13327     }%
        Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.
13328     }%
13329 }%
13330 }%
13331 }

```

tWidestFallback

`\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).

```

13332 \newcommand*{\glsxtrSetWidestFallback}[2]{%
13333     \ifnum#1=0\relax
13334     \ifdef\glsFindWidestTopLevelName
13335     {%
13336         \glsFindWidestTopLevelName[#2]%
13337     }%
13338     {%

```

```

13339     \GlossariesExtraWarning{You need stylemods={tree} to
13340     provide a fallback for set-widest}%
13341 }%
13342 \else
13343 \ifdef\glsFindWidestLevelTwo
13344 {%
13345     \glsFindWidestLevelTwo[#2]%
13346     \ifdef\glslongextraUpdateWidestChild
13347     {%
13348         \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
13349         \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
13350     }%
13351     {}%
13352 }%
13353 {%
13354     \GlossariesExtraWarning{You need stylemods={tree} to
13355     provide a fallback for set-widest}%
13356 }%
13357 \fi
13358 }

```

`r@labelprefixes` List of label prefixes.

```

13359 \newcommand*{\@glxstr@labelprefixes}{}

```

`arlabelprefixes` List of label prefixes.

```

13360 \newcommand*{\glxstrclearlabelprefixes}{%
13361     \renewcommand*{\@glxstr@labelprefixes}{}%
13362 }

```

`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.

```

13363 \newcommand*{\glxstraddlabelprefix}[1]{%
13364     \ifstrempy{#1}%
13365     {\glxstraddlabelprefix{\empty}}%
13366     {%
13367         \ifdefempty\@glxstr@labelprefixes
13368         {\def\@glxstr@labelprefixes{#1}}%
13369         {\appto\@glxstr@labelprefixes{,#1}}%
13370     }%
13371 }

```

`pendlabelprefix` Inserts at the start of the list.

```

13372 \newcommand*{\glxstrprependlabelprefix}[1]{%
13373     \ifstrempy{#1}%
13374     {\glxstrprependlabelprefix{\empty}}%
13375     {%
13376         \ifdefempty\@glxstr@labelprefixes

```



```

13377   {\def\@glxstr@labelprefixes{#1}}%
13378   {\preto\@glxstr@labelprefixes{#1,}}%
13379   }%
13380 }

```

labelprefixlist

`\glxstrifinlabelprefixlist{<prefix>}{<true>}{<false>}`

Test if the given prefix is in the list.

```

13381 \newcommand*{\glxstrifinlabelprefixlist}[3]{%
13382   \ifstrempy{#1}%
13383   {\glxstrifinlabelprefixlist{\empty}{#2}{#3}}%
13384   {%
13385     \DTLifinlist{#1}{\@glxstr@labelprefixes}{#2}{#3}%
13386   }%
13387 }

```

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**.

```

13388 \AtBeginDocument{%
13389   \protected@write\@auxout{}\string\providecommand{\string\@glxstr@prefixlabellist}[1]{}%
13390   \protected@write\@auxout{}\string\@glxstr@prefixlabellist{\@glxstr@labelprefixes}%
13391 }

```

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 \TeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```

13392 \newcommand*{\@glxstr@get@prefixedlabel}[1]{%
13393   \begingroup

```

Initialise to the unprefix label in the event that the list is empty.

```

13394   \edef\@gls@thislabel{#1}%
13395   \@for\@glxstr@prefix:=\@glxstr@labelprefixes\do
13396   {%
13397     \edef\@gls@thislabel{\@glxstr@prefix#1}%
13398     \ifglentryexists{\@gls@thislabel}{\@endfortrue}{}%
13399   }%
13400   \edef\x{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\x
13401 }

```

\dgl Like \gls but tries the prefixes. (Can't use \pgl as that's provided by glossaries-prefix.) Since this command is designed for bib2gls's dual entry system, the "d" stands for "dual".

```

13402 \newrobustcmd*{\dgl}{\@gls@hyp@opt\@dgl}

```

```

\@dglS
13403 \newcommand*{\@dglS}[2][\%
13404 \@glSxtr@get@prefixedlabel{#2}%
13405 \new@ifnextchar[{\@glS@{#1}{\@glS@thislabel}}{\@glS@{#1}{\@glS@thislabel}}[]}%
13406 }

\dglSpl
13407 \newrobustcmd*{\dglSpl}{\@glS@hyp@opt\dglSpl}

\dglSpl
13408 \newcommand*{\@dglSpl}[2][\%
13409 \@glSxtr@get@prefixedlabel{#2}%
13410 \new@ifnextchar[{\@glSpl@{#1}{\@glS@thislabel}}{\@glSpl@{#1}{\@glS@thislabel}}[]}%
13411 }

\dGLS
13412 \newrobustcmd*{\dGLS}{\@glS@hyp@opt\dGLS}

\dGLS
13413 \newcommand*{\@dGLS}[2][\%
13414 \@glSxtr@get@prefixedlabel{#2}%
13415 \new@ifnextchar[{\@dGLS@{#1}{\@glS@thislabel}}{\@dGLS@{#1}{\@glS@thislabel}}[]}%
13416 }

\dGLSpl
13417 \newrobustcmd*{\dGLSpl}{\@glS@hyp@opt\dGLSpl}

\dGLSpl
13418 \newcommand*{\@dGLSpl}[2][\%
13419 \@glSxtr@get@prefixedlabel{#2}%
13420 \new@ifnextchar[{\@dGLSpl@{#1}{\@glS@thislabel}}{\@dGLSpl@{#1}{\@glS@thislabel}}[]}%
13421 }

\dGLS
13422 \newrobustcmd*{\dGLS}{\@glS@hyp@opt\dGLS}

\dGLS
13423 \newcommand*{\@dGLS}[2][\%
13424 \@glSxtr@get@prefixedlabel{#2}%
13425 \new@ifnextchar[{\@dGLS@{#1}{\@glS@thislabel}}{\@dGLS@{#1}{\@glS@thislabel}}[]}%
13426 }

\dGLSpl
13427 \newrobustcmd*{\dGLSpl}{\@glS@hyp@opt\dGLSpl}

```

`\@dGLSp1`

```
13428 \newcommand*{\@dGLSp1}[2] [] {%
13429   \@glsxtr@get@prefixedlabel{#2}%
13430   \new@ifnextchar[{\@GLSp1@{#1}{\@gls@thislabel}}{\@GLSp1@{#1}{\@gls@thislabel}}[]}%
13431 }
```

`\dglslink` Like `\glslink` but tries the prefixes.

```
13432 \newrobustcmd*{\dglslink}[3] [] {%
13433   \@glsxtr@get@prefixedlabel{#2}%
13434   \glslink[#1]{\@gls@thislabel}{#3}%
13435 }
```

`\dglstdisp` Like `\glsdisp` but tries the prefixes.

```
13436 \newrobustcmd*{\dglstdisp}[3] [] {%
13437   \@glsxtr@get@prefixedlabel{#2}%
13438   \glsdisp[#1]{\@gls@thislabel}{#3}%
13439 }
```

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`

```
13440 \providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
13441 \providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
13442 \providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
13443 \providecommand*{\Zeta}{\mathrm{Z}}
```

`\Eta`

```
13444 \providecommand*{\Eta}{\mathrm{H}}
```

`\Iota`

```
13445 \providecommand*{\Iota}{\mathrm{I}}
```

`\Kappa`

```
13446 \providecommand*{\Kappa}{\mathrm{K}}
```

`\Mu`

```
13447 \providecommand*{\Mu}{\mathrm{M}}
```

```

\Nu
13448 \providecommand*\Nu{\mathrm{N}}

\Omicron
13449 \providecommand*\Omicron{\mathrm{O}}

\Rho
13450 \providecommand*\Rho{\mathrm{P}}

\Tau
13451 \providecommand*\Tau{\mathrm{T}}

\Chi
13452 \providecommand*\Chi{\mathrm{X}}

\Digamma
13453 \providecommand*\Digamma{\mathrm{F}}

\omicron
13454 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
13455 \@ifpackageloaded{upgreek}%
13456 {

\Upalpha
13457 \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
13458 \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
13459 \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
13460 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
13461 \providecommand*\Upeta{\mathrm{H}}

\Upiota
13462 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
13463 \providecommand*\Upkappa{\mathrm{K}}

```

```

\Upmu
13464 \providecommand*\Upmu{\mathrm{M}}

\Upnu
13465 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
13466 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
13467 \providecommand*\Uprho{\mathrm{P}}

\Uptau
13468 \providecommand*\Uptau{\mathrm{T}}

\Upchi
13469 \providecommand*\Upchi{\mathrm{X}}

\upomicron
13470 \providecommand*\upomicron{\mathrm{o}}

13471 }%
13472 {}% 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.

```
13473 \newcommand*{\glxtrcontrolrules}{%
13474 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
13475 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
13476 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
13477 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
13478 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
13479 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
13480 0010\string'\string=\glshex 0011
13481 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
13482 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
13483 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
13484 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
13485 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
13486 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
13487 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
13488 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
13489 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
13490 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
13491 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
13492 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
13493 }
```

lsxtrspacerules These are space characters.

```
13494 \newcommand*{\glxtrspacerules}{%
13495 \string' \string'\string;
13496 \string'\glshex 00A0\string'\string;
13497 \string'\glshex 2000\string'\string;
13498 \string'\glshex 2001\string'\string;
13499 \string'\glshex 2002\string'\string;
13500 \string'\glshex 2003\string'\string;
13501 \string'\glshex 2004\string'\string;
13502 \string'\glshex 2005\string'\string;
13503 \string'\glshex 2006\string'\string;
13504 \string'\glshex 2007\string'\string;
13505 \string'\glshex 2008\string'\string;
13506 \string'\glshex 2009\string'\string;
13507 \string'\glshex 200A\string'\string;
13508 \string'\glshex 3000\string'
13509 }
```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```
13510 \newcommand*{\glxtrnonprintablerules}{%
13511 \string'\glshex FEFF\string'\string;
13512 \string'\glshex 000A\string'\string;
13513 \string'\glshex 0009\string'\string;
```

```

13514 \string'\glshex 000C\string'\string;
13515 \string'\glshex 000B\string'
13516 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

13517 \newcommand*{\glxtrcombiningdiacriticrules}{%
13518 \glxtrcombiningdiacriticIrules\string;
13519 \glxtrcombiningdiacriticIIrules\string;
13520 \glxtrcombiningdiacriticIIIrules\string;
13521 \glxtrcombiningdiacriticIVrules
13522 }

```

diacriticIrules First set of combining diacritic marks.

```

13523 \newcommand*{\glxtrcombiningdiacriticIrules}{%
13524 \glshex 0301\string;% combining acute
13525 \glshex 0300\string;% combining grave
13526 \glshex 0306\string;% combining breve
13527 \glshex 0302\string;% combining circumflex
13528 \glshex 030C\string;% combining caron
13529 \glshex 030A\string;% combining ring
13530 \glshex 030D\string;% combining vertical line above
13531 \glshex 0308\string;% combining diaeresis
13532 \glshex 030B\string;% combining double acute
13533 \glshex 0303\string;% combining tilde
13534 \glshex 0307\string;% combining dot above
13535 \glshex 0304% combining macron
13536 }

```

diacriticIIrules Second set of combining diacritic marks.

```

13537 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
13538 \glshex 0337\string;% combining short solidus overlay
13539 \glshex 0327\string;% combining cedilla
13540 \glshex 0328\string;% combining ogonek
13541 \glshex 0323\string;% combining dot below
13542 \glshex 0332\string;% combining low line
13543 \glshex 0305\string;% combining overline
13544 \glshex 0309\string;% combining hook above
13545 \glshex 030E\string;% combining double vertical line above
13546 \glshex 030F\string;% combining double grave accent
13547 \glshex 0310\string;% combining candrabindu
13548 \glshex 0311\string;% combining inverted breve
13549 \glshex 0312\string;% combining turned comma above
13550 \glshex 0313\string;% combining comma above
13551 \glshex 0314\string;% combining reversed comma above
13552 \glshex 0315\string;% combining comma above right
13553 \glshex 0316\string;% combining grave accent below
13554 \glshex 0317% combining acute accent below
13555 }

```

acriticIIIrules Third set of combining diacritic marks.

```
13556 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
13557 \glshex 0318\string;% combining left tack below
13558 \glshex 0319\string;% combining right tack below
13559 \glshex 031A\string;% combining left angle above
13560 \glshex 031B\string;% combining horn
13561 \glshex 031C\string;% combining left half ring below
13562 \glshex 031D\string;% combining up tack below
13563 \glshex 031E\string;% combining down tack below
13564 \glshex 031F\string;% combining plus sign below
13565 \glshex 0320\string;% combining minus sign below
13566 \glshex 0321\string;% combining palatalized hook below
13567 \glshex 0322\string;% combining retroflex hook below
13568 \glshex 0324\string;% combining diaeresis below
13569 \glshex 0325\string;% combining ring below
13570 \glshex 0326\string;% combining comma below
13571 \glshex 0329\string;% combining vertical line below
13572 \glshex 032A\string;% combining bridge below
13573 \glshex 032B\string;% combining inverted double arch below
13574 \glshex 032C\string;% combining caron below
13575 \glshex 032D\string;% combining circumflex accent below
13576 \glshex 032E\string;% combining breve below
13577 \glshex 032F\string;% combining inverted breve below
13578 \glshex 0330\string;% combining tilde below
13579 \glshex 0331\string;% combining macron below
13580 \glshex 0333\string;% combining double low line
13581 \glshex 0334\string;% combining tilde overlay
13582 \glshex 0335\string;% combining short stroke overlay
13583 \glshex 0336\string;% combining long stroke overlay
13584 \glshex 0338\string;% combining long solidus overlay
13585 \glshex 0339\string;% combining combining right half ring below
13586 \glshex 033A\string;% combining inverted bridge below
13587 \glshex 033B\string;% combining square below
13588 \glshex 033C\string;% combining seagull below
13589 \glshex 033D\string;% combining x above
13590 \glshex 033E\string;% combining vertical tilde
13591 \glshex 033F\string;% combining double overline
13592 \glshex 0342\string;% combining Greek perispomeni
13593 \glshex 0344\string;% combining Greek dialytika tonos
13594 \glshex 0345\string;% combining Greek ypogegrammeni
13595 \glshex 0360\string;% combining double tilde
13596 \glshex 0361\string;% combining double inverted breve
13597 \glshex 0483\string;% combining Cyrillic titlo
13598 \glshex 0484\string;% combining Cyrillic palatalization
13599 \glshex 0485\string;% combining Cyrillic dasia pneumata
13600 \glshex 0486% combining Cyrillic psili pneumata
13601 }
```

iacriticIVrules Fourth set of combining diacritic marks.


```

13602 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
13603   \glshex 20D0\string;% combining left harpoon above
13604   \glshex 20D1\string;% combining right harpoon above
13605   \glshex 20D2\string;% combining long vertical line overlay
13606   \glshex 20D3\string;% combining short vertical line overlay
13607   \glshex 20D4\string;% combining anticlockwise arrow above
13608   \glshex 20D5\string;% combining clockwise arrow above
13609   \glshex 20D6\string;% combining left arrow above
13610   \glshex 20D7\string;% combining right arrow above
13611   \glshex 20D8\string;% combining ring overlay
13612   \glshex 20D9\string;% combining clockwise ring overlay
13613   \glshex 20DA\string;% combining anticlockwise ring overlay
13614   \glshex 20DB\string;% combining three dots above
13615   \glshex 20DC\string;% combining four dots above
13616   \glshex 20DD\string;% combining enclosing circle
13617   \glshex 20DE\string;% combining enclosing square
13618   \glshex 20DF\string;% combining enclosing diamond
13619   \glshex 20E0\string;% combining enclosing circle backslash
13620   \glshex 20E1% combining left right arrow above
13621 }

```

sxtrhyphenrules Hyphens.

```

13622 \newcommand*{\glxtrhyphenrules}{%
13623   \string'\string-\string'\string;% ASCII hyphen
13624   \glshex 00AD\string;% soft hyphen
13625   \glshex 2010\string;% hyphen
13626   \glshex 2011\string;% non-breaking hyphen
13627   \glshex 2012\string;% figure dash
13628   \glshex 2013\string;% en dash
13629   \glshex 2014\string;% em dash
13630   \glshex 2015\string;% horizontal bar
13631   \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
13632 }

```

eneralpuncrules General punctuation.

```

13633 \newcommand*{\glxtrgeneralpuncrules}{%
13634   \glxtrgeneralpuncIrules
13635   \string<\glxtrcurrencyrules
13636   \string<\glxtrgeneralpuncIIrules
13637 }

```

eneralpuncIrules First set of general punctuation.

```

13638 \newcommand*{\glxtrgeneralpuncIrules}{%
13639   \string'\glshex 005F\string'% underscore
13640   \string<\glshex 00AF% macron
13641   \string<\string'\glshex 002C\string'% comma
13642   \string<\string'\glshex 003B\string'% semi-colon
13643   \string<\string'\glshex 003A\string'% colon
13644   \string<\string'\glshex 0021\string'% exclamation mark

```

13645 \string<\glshex 00A1% inverted exclamation mark
 13646 \string<\string'\glshex 003F\string'% question mark
 13647 \string<\glshex 00BF% inverted question mark
 13648 \string<\string'\glshex 002F\string'% solidus
 13649 \string<\string'\glshex 002E\string'% full stop
 13650 \string<\glshex 00B4% acute accent
 13651 \string<\string'\glshex 0060\string'% grave accent
 13652 \string<\string'\glshex 005E\string'% circumflex accent
 13653 \string<\glshex 00A8% diaeresis
 13654 \string<\string'\glshex 007E\string'% tilde
 13655 \string<\glshex 00B7% middle dot
 13656 \string<\glshex 00B8% cedilla
 13657 \string<\string'\glshex 0027\string'% straight apostrophe
 13658 \string<\string'\glshex 0022\string'% straight double quote
 13659 \string<\glshex 00AB% left guillemet
 13660 \string<\glshex 00BB% right guillemet
 13661 \string<\string'\glshex 0028\string'% left parenthesis
 13662 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
 13663 \string<\string'\glshex 0029\string'% right parenthesis
 13664 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
 13665 \string<\string'\glshex 005B\string'% left square bracket
 13666 \string<\string'\glshex 005D\string'% right square bracket
 13667 \string<\string'\glshex 007B\string'% left curly bracket
 13668 \string<\string'\glshex 007D\string'% right curly bracket
 13669 \string<\glshex 00A7% section sign
 13670 \string<\glshex 00B6% pilcrow sign
 13671 \string<\glshex 00A9% copyright sign
 13672 \string<\glshex 00AE% registered sign
 13673 \string<\string'\glshex 0040\string'% at sign
 13674 }

trcurrencyrules General punctuation.

13675 \newcommand*{\glxtrcurrencyrules}{%
 13676 \glshex 00A4% currency sign
 13677 \string<\glshex 0E3F% Thai currency symbol baht
 13678 \string<\glshex 00A2% cent sign
 13679 \string<\glshex 20A1% colon sign
 13680 \string<\glshex 20A2% cruzeiro sign
 13681 \string<\string'\glshex 0024\string'% dollar sign
 13682 \string<\glshex 20AB% dong sign
 13683 \string<\glshex 20AC% euro sign
 13684 \string<\glshex 20A3% French franc sign
 13685 \string<\glshex 20A4% lira sign
 13686 \string<\glshex 20A5% mill sign
 13687 \string<\glshex 20A6% naira sign
 13688 \string<\glshex 20A7% peseta sign
 13689 \string<\glshex 00A3% pound sign
 13690 \string<\glshex 20A8% rupee sign
 13691 \string<\glshex 20AA% new sheqel sign

```

13692 \string<\glshex 20A9% won sign
13693 \string<\glshex 00A5% yen sign
13694 }

```

eralpuncIIrules Second set of general punctuation.

```

13695 \newcommand*{\glxtrgeneralpuncIIrules}{%
13696 \string'\glshex 002A\string'% asterisk
13697 \string<\string'\glshex 005C\string'% backslash
13698 \string<\string'\glshex 0026\string'% ampersand
13699 \string<\string'\glshex 0023\string'% hash sign
13700 \string<\string'\glshex 0025\string'% percent sign
13701 \string<\string'\glshex 002B\string'% plus sign
13702 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
13703 \string<\glshex 00B1% plus-minus sign
13704 \string<\glshex 00F7% division sign
13705 \string<\glshex 00D7% multiplication sign
13706 \string<\string'\glshex 003C\string'% less-than sign
13707 \string<\string'\glshex 003D\string'% equals sign
13708 \string<\string'\glshex 003E\string'% greater-than sign
13709 \string<\glshex 00AC% not sign
13710 \string<\string'\glshex 007C\string'% vertical bar (pipe)
13711 \string<\glshex 00A6% broken bar
13712 \string<\glshex 00B0% degree sign
13713 \string<\glshex 00B5% micron sign
13714 }

```

eralLatinIrules Basic Latin alphabet.

```

13715 \newcommand*{\glxtrGeneralLatinIrules}{%
13716 \glxtrLatinA
13717 \string<b,B%
13718 \string<c,C%
13719 \string<d,D%
13720 \string<\glxtrLatinE
13721 \string<f,F%
13722 \string<g,G%
13723 \string<\glxtrLatinH
13724 \string<\glxtrLatinI
13725 \string<j,J%
13726 \string<\glxtrLatinK
13727 \string<\glxtrLatinL
13728 \string<\glxtrLatinM
13729 \string<\glxtrLatinN
13730 \string<\glxtrLatinO
13731 \string<\glxtrLatinP
13732 \string<q,Q%
13733 \string<r,R%
13734 \string<\glxtrLatinS
13735 \string<\glxtrLatinT
13736 \string<u,U%

```

```

13737 \string<v,V%
13738 \string<w,W%
13739 \string<\glxtrLatinX
13740 \string<y,Y%
13741 \string<z,Z
13742 }

```

allLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

13743 \newcommand*{\glxtrGeneralLatinIIrules}{%
13744 \glxtrLatinA
13745 \string<b,B%
13746 \string<c,C%
13747 \string<d,D%
13748 \string<\glxtrLatinEth
13749 \string<\glxtrLatinE
13750 \string<f,F%
13751 \string<g,G%
13752 \string<\glxtrLatinH
13753 \string<\glxtrLatinI
13754 \string<j,J%
13755 \string<\glxtrLatinK
13756 \string<\glxtrLatinL
13757 \string<\glxtrLatinM
13758 \string<\glxtrLatinN
13759 \string<\glxtrLatinO
13760 \string<\glxtrLatinP
13761 \string<q,Q%
13762 \string<r,R%
13763 \string<\glxtrLatinS
13764 \string& SS \string, \glxtrLatinEszettSs
13765 \string<\glxtrLatinT
13766 \string<u,U%
13767 \string<v,V%
13768 \string<w,W%
13769 \string<\glxtrLatinX
13770 \string<y,Y%
13771 \string<z,Z%
13772 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

13773 \newcommand*{\glxtrGeneralLatinIIIrules}{%
13774 \glxtrLatinA
13775 \string<b,B%
13776 \string<c,C%
13777 \string<d,D%
13778 \string<\glxtrLatinEth
13779 \string<\glxtrLatinE
13780 \string<f,F%
13781 \string<g,G%

```

```

13782 \string<\glxtrLatinH
13783 \string<\glxtrLatinI
13784 \string<j,J%
13785 \string<\glxtrLatinK
13786 \string<\glxtrLatinL
13787 \string<\glxtrLatinM
13788 \string<\glxtrLatinN
13789 \string<\glxtrLatinO
13790 \string<\glxtrLatinP
13791 \string<q,Q%
13792 \string<r,R%
13793 \string<\glxtrLatinS
13794 \string& SZ, \glxtrLatinEszettSz
13795 \string<\glxtrLatinT
13796 \string<u,U%
13797 \string<v,V%
13798 \string<w,W%
13799 \string<\glxtrLatinX
13800 \string<y,Y%
13801 \string<z,Z%
13802 }

```

GeneralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

13803 \newcommand*{\glxtrGeneralLatinIVrules}{%
13804 \glxtrLatinA
13805 \string& AE , \glxtrLatinAELigature
13806 \string<b,B%
13807 \string<c,C%
13808 \string<d,D%
13809 \string<\glxtrLatinEth
13810 \string<\glxtrLatinE
13811 \string<f,F%
13812 \string<g,G%
13813 \string<\glxtrLatinH
13814 \string<\glxtrLatinI
13815 \string<j,J%
13816 \string<\glxtrLatinK
13817 \string<\glxtrLatinL
13818 \string<\glxtrLatinM
13819 \string<\glxtrLatinN
13820 \string<\glxtrLatinO
13821 \string& OE , \glxtrLatinOELigature
13822 \string<\glxtrLatinP
13823 \string<q,Q%
13824 \string<r,R%
13825 \string<\glxtrLatinS
13826 \string& SS , \glxtrLatinEszettSs
13827 \string<\glxtrLatinT

```

```

13828 \string& th =\glshex 00DE
13829 \string& TH =\glshex 00FE
13830 \string<u,U%
13831 \string<v,V%
13832 \string<w,W%
13833 \string<\glxtrLatinX
13834 \string<y,Y%
13835 \string<z,Z%
13836 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

13837 \newcommand*{\glxtrGeneralLatinVrules}{%
13838 \glxtrLatinA
13839 \string<b,B%
13840 \string<c,C%
13841 \string<d,D%
13842 \string<\glxtrLatinEth
13843 \string<\glxtrLatinE
13844 \string<f,F%
13845 \string<g,G%
13846 \string<\glxtrLatinH
13847 \string<\glxtrLatinI
13848 \string<j,J%
13849 \string<\glxtrLatinK
13850 \string<\glxtrLatinL
13851 \string<\glxtrLatinM
13852 \string<\glxtrLatinN
13853 \string<\glxtrLatinO
13854 \string<\glxtrLatinP
13855 \string<q,Q%
13856 \string<r,R%
13857 \string<\glxtrLatinS
13858 \string& SS , \glxtrLatinEszettSs
13859 \string<\glxtrLatinT
13860 \string& th =\glshex 00DE
13861 \string& TH =\glshex 00FE
13862 \string<u,U%
13863 \string<v,V%
13864 \string<w,W%
13865 \string<\glxtrLatinX
13866 \string<y,Y%
13867 \string<z,Z%
13868 }

```

ralLatinVrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

13869 \newcommand*{\glxtrGeneralLatinVrules}{%
13870 \glxtrLatinA
13871 \string<b,B%
13872 \string<c,C%

```

```

13873 \string<d,D%
13874 \string<\glxtrLatinEth
13875 \string<\glxtrLatinE
13876 \string<f,F%
13877 \string<g,G%
13878 \string<\glxtrLatinH
13879 \string<\glxtrLatinI
13880 \string<j,J%
13881 \string<\glxtrLatinK
13882 \string<\glxtrLatinL
13883 \string<\glxtrLatinM
13884 \string<\glxtrLatinN
13885 \string<\glxtrLatinO
13886 \string<\glxtrLatinP
13887 \string<q,Q%
13888 \string<r,R%
13889 \string<\glxtrLatinS
13890 \string& SZ , \glxtrLatinEszettSz
13891 \string<\glxtrLatinT
13892 \string& th =\glshex 00DE
13893 \string& TH =\glshex 00FE
13894 \string<u,U%
13895 \string<v,V%
13896 \string<w,W%
13897 \string<\glxtrLatinX
13898 \string<y,Y%
13899 \string<z,Z%
13900 }

```

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).

```

13901 \newcommand*{\glxtrGeneralLatinVIIrules}{%
13902 \glxtrLatinA
13903 \string<\glxtrLatinAELigature
13904 \string<b,B%
13905 \string<c,C%
13906 \string<d,D%
13907 \string<\glxtrLatinEth
13908 \string<\glxtrLatinE
13909 \string<f,F%
13910 \string<\glxtrLatinInsularG
13911 \string<\glxtrLatinH
13912 \string<\glxtrLatinI
13913 \string<j,J%
13914 \string<\glxtrLatinK
13915 \string<\glxtrLatinL
13916 \string<\glxtrLatinM
13917 \string<\glxtrLatinN
13918 \string<\glxtrLatinO

```

```

13919 \string<\glxtrLatinOELigature
13920 \string<\glxtrLatinP
13921 \string<q,Q%
13922 \string<r,R%
13923 \string<\glshex 017F=\glxtrLatinS % s and long s
13924 \string<\glxtrLatinT
13925 \string<\glxtrLatinThorn
13926 \string<u,U%
13927 \string<v,V%
13928 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
13929 \string<\glxtrLatinX
13930 \string<y,Y%
13931 \string<z,Z%
13932 }

```

1LatinVIIIrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

13933 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
13934 \glxtrLatinA
13935 \string& AE , \glxtrLatinAELigature
13936 \string<b,B%
13937 \string<c,C%
13938 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
13939 \string<\glxtrLatinE
13940 \string<f,F%
13941 \string<g,G%
13942 \string<\glxtrLatinH
13943 \string<\glxtrLatinI
13944 \string<j,J%
13945 \string<\glxtrLatinK
13946 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
13947 \string<\glxtrLatinM
13948 \string<\glxtrLatinN
13949 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
13950 \string& OE , \glxtrLatinOELigature
13951 \string<\glxtrLatinP
13952 \string<q,Q%
13953 \string<r,R%
13954 \string<\glxtrLatinS
13955 \string& SS , \glxtrLatinEszettSs
13956 \string<\glxtrLatinT
13957 \string& th =\glshex 00DE
13958 \string& TH =\glshex 00FE
13959 \string<u,U%
13960 \string<v,V%
13961 \string<w,W%
13962 \string<\glxtrLatinX
13963 \string<y,Y%
13964 \string<z,Z%

```


13965 }

\glsxtrLatinA

13966 \newcommand*{\glsxtrLatinA}{%
13967 a\string=\glshex 00AA\string=\glshex 2090,A
13968 }

\glsxtrLatinE

13969 \newcommand*{\glsxtrLatinE}{%
13970 e\string=\glshex 2091,E
13971 }

\glsxtrLatinH

13972 \newcommand*{\glsxtrLatinH}{%
13973 h\string=\glshex 2095,H
13974 }

\glsxtrLatinI

13975 \newcommand*{\glsxtrLatinI}{%
13976 i\string=\glshex 2071,I
13977 }

\glsxtrLatinK

13978 \newcommand*{\glsxtrLatinK}{%
13979 k\string=\glshex 2096,K
13980 }

\glsxtrLatinL

13981 \newcommand*{\glsxtrLatinL}{%
13982 l\string=\glshex 2097,L
13983 }

\glsxtrLatinM

13984 \newcommand*{\glsxtrLatinM}{%
13985 m\string=\glshex 2098,M
13986 }

\glsxtrLatinN

13987 \newcommand*{\glsxtrLatinN}{%
13988 n\string=\glshex 207F\string=\glshex 2099,N
13989 }

\glsxtrLatinO

13990 \newcommand*{\glsxtrLatinO}{%
13991 o\string=\glshex 00BA\string=\glshex 2092,O
13992 }

```

\glxtrLatinP
13993 \newcommand*{\glxtrLatinP}{%
13994   p\string=\glshex 209A,P
13995 }

\glxtrLatinS
13996 \newcommand*{\glxtrLatinS}{%
13997   s\string=\glshex 209B,S
13998 }

\glxtrLatinT
13999 \newcommand*{\glxtrLatinT}{%
14000   t\string=\glshex 209C,T
14001 }

\glxtrLatinX
14002 \newcommand*{\glxtrLatinX}{%
14003   x\string=\glshex 2093,X
14004 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
14005 \newcommand*{\glxtrLatinSchwa}{%
14006   \glshex 0259\string=\glshex 2094,\glshex 018F
14007 }

trLatinEszettSs
14008 \newcommand*{\glxtrLatinEszettSs}{%
14009   \glshex 00DF% eszett
14010   \string=\glshex 017Fs % long S s
14011 }

trLatinEszettSz
14012 \newcommand*{\glxtrLatinEszettSz}{%
14013   \glshex 00DF% eszett
14014   \string= \glshex 017Fz % long S z
14015 }

\glxtrLatinEth
14016 \newcommand*{\glxtrLatinEth}{%
14017   \glshex 00F0,\glshex 00D0% eth
14018 }

lsxtrLatinThorn
14019 \newcommand*{\glxtrLatinThorn}{%
14020   \glshex 00FE,\glshex 00DE% thorn
14021 }

```

LatinAELigature

```
14022 \newcommand*{\glxtrLatinAELigature}{%
14023   \glshex 00E6,\glshex 00C6% AE-ligature
14024 }
```

LatinOELigature

```
14025 \newcommand*{\glxtrLatinOELigature}{%
14026   \glshex 0153,\glshex 0152% OE-ligature
14027 }
```

\glxtrLatinAA

```
14028 \newcommand*{\glxtrLatinAA}{%
14029   \glshex 00E5=a\glshex 030A,% \aa
14030   \glshex 00C5=A\glshex 030A% \AA
14031 }
```

glxtrLatinWynn

```
14032 \newcommand*{\glxtrLatinWynn}{%
14033   \glshex 01BF,\glshex 01F7% wynn
14034 }
```

trLatinInsularG

```
14035 \newcommand*{\glxtrLatinInsularG}{%
14036   \glshex 1D79,\glshex A77D% insular G
14037   \string; g, G
14038 }
```

sxtrLatinOslash

```
14039 \newcommand*{\glxtrLatinOslash}{%
14040   \glshex 00F8,\glshex 00D8% \o, \O
14041 }
```

sxtrLatinLslash

```
14042 \newcommand*{\glxtrLatinLslash}{%
14043   \glshex 0142,\glshex 0141% \l, \L
14044 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
14045 \newcommand*{\glxtrMathUpGreekIrules}{%
14046   \glxtrUpAlpha
14047   \string<\glxtrUpBeta
14048   \string<\glxtrUpGamma
14049   \string<\glxtrUpDelta
14050   \string<\glxtrUpEpsilon
14051   \string<\glxtrUpDigamma
14052   \string<\glxtrUpZeta
14053   \string<\glxtrUpEta
14054   \string<\glxtrUpTheta
```

```

14055 \string<\glxtrUpIota
14056 \string<\glxtrUpKappa
14057 \string<\glxtrUpLambda
14058 \string<\glxtrUpMu
14059 \string<\glxtrUpNu
14060 \string<\glxtrUpXi
14061 \string<\glxtrUpOmicron
14062 \string<\glxtrUpPi
14063 \string<\glxtrUpRho
14064 \string<\glxtrUpSigma
14065 \string<\glxtrUpTau
14066 \string<\glxtrUpUpsilon
14067 \string<\glxtrUpPhi
14068 \string<\glxtrUpChi
14069 \string<\glxtrUpPsi
14070 \string<\glxtrUpOmega
14071 }

```

hUpGreekIIrules Doesn't include digamma.

```

14072 \newcommand*{\glxtrMathUpGreekIIrules}{%
14073 \glxtrUpAlpha
14074 \string<\glxtrUpBeta
14075 \string<\glxtrUpGamma
14076 \string<\glxtrUpDelta
14077 \string<\glxtrUpEpsilon
14078 \string<\glxtrUpZeta
14079 \string<\glxtrUpEta
14080 \string<\glxtrUpTheta
14081 \string<\glxtrUpIota
14082 \string<\glxtrUpKappa
14083 \string<\glxtrUpLambda
14084 \string<\glxtrUpMu
14085 \string<\glxtrUpNu
14086 \string<\glxtrUpXi
14087 \string<\glxtrUpOmicron
14088 \string<\glxtrUpPi
14089 \string<\glxtrUpRho
14090 \string<\glxtrUpSigma
14091 \string<\glxtrUpTau
14092 \string<\glxtrUpUpsilon
14093 \string<\glxtrUpPhi
14094 \string<\glxtrUpChi
14095 \string<\glxtrUpPsi
14096 \string<\glxtrUpOmega
14097 }

```

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.

```

14098 \newcommand*{\glxtrMathItalicGreekIrules}{%

```

```

14099 \glxtrMathItalicAlpha
14100 \string<\glxtrMathItalicBeta
14101 \string<\glxtrMathItalicGamma
14102 \string<\glxtrMathItalicDelta
14103 \string<\glxtrMathItalicEpsilon
14104 \string<\glxtrUpDigamma
14105 \string<\glxtrMathItalicZeta
14106 \string<\glxtrMathItalicEta
14107 \string<\glxtrMathItalicTheta
14108 \string<\glxtrMathItalicIota
14109 \string<\glxtrMathItalicKappa
14110 \string<\glxtrMathItalicLambda
14111 \string<\glxtrMathItalicMu
14112 \string<\glxtrMathItalicNu
14113 \string<\glxtrMathItalicXi
14114 \string<\glxtrMathItalicOmicron
14115 \string<\glxtrMathItalicPi
14116 \string<\glxtrMathItalicRho
14117 \string<\glxtrMathItalicSigma
14118 \string<\glxtrMathItalicTau
14119 \string<\glxtrMathItalicUpsilon
14120 \string<\glxtrMathItalicPhi
14121 \string<\glxtrMathItalicChi
14122 \string<\glxtrMathItalicPsi
14123 \string<\glxtrMathItalicOmega
14124 }

```

licGreekIIrules Doesn't include digamma.

```

14125 \newcommand*{\glxtrMathItalicGreekIIrules}{%
14126 \glxtrMathItalicAlpha
14127 \string<\glxtrMathItalicBeta
14128 \string<\glxtrMathItalicGamma
14129 \string<\glxtrMathItalicDelta
14130 \string<\glxtrMathItalicEpsilon
14131 \string<\glxtrMathItalicZeta
14132 \string<\glxtrMathItalicEta
14133 \string<\glxtrMathItalicTheta
14134 \string<\glxtrMathItalicIota
14135 \string<\glxtrMathItalicKappa
14136 \string<\glxtrMathItalicLambda
14137 \string<\glxtrMathItalicMu
14138 \string<\glxtrMathItalicNu
14139 \string<\glxtrMathItalicXi
14140 \string<\glxtrMathItalicOmicron
14141 \string<\glxtrMathItalicPi
14142 \string<\glxtrMathItalicRho
14143 \string<\glxtrMathItalicSigma
14144 \string<\glxtrMathItalicTau
14145 \string<\glxtrMathItalicUpsilon

```

```

14146 \string<\glxtrMathItalicPhi
14147 \string<\glxtrMathItalicChi
14148 \string<\glxtrMathItalicPsi
14149 \string<\glxtrMathItalicOmega
14150 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

14151 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
14152 \glshex 1D6E2% upper case alpha (maths italic)
14153 \string<\glshex 1D6E3% upper case beta (maths italic)
14154 \string<\glshex 1D6E4% upper case gamma (maths italic)
14155 \string<\glshex 1D6E5% upper case delta (maths italic)
14156 \string<\glshex 1D6E6% upper case epsilon (maths italic)
14157 \string<\glshex 03DC% upper case digamma
14158 \string<\glshex 1D6E7% upper case zeta (maths italic)
14159 \string<\glshex 1D6E8% upper case eta (maths italic)
14160 \string<\glshex 1D6E9% upper case theta (maths italic)
14161 \string<\glshex 1D6F3% upper case theta variant (maths italic)
14162 \string<\glshex 1D6EA% upper case iota (maths italic)
14163 \string<\glshex 1D6EB% upper case kappa (maths italic)
14164 \string<\glshex 1D6EC% upper case lambda (maths italic)
14165 \string<\glshex 1D6ED% upper case mu (maths italic)
14166 \string<\glshex 1D6EE% upper case nu (maths italic)
14167 \string<\glshex 1D6EF% upper case xi (maths italic)
14168 \string<\glshex 1D6F0% upper case omicron (maths italic)
14169 \string<\glshex 1D6F1% upper case pi (maths italic)
14170 \string<\glshex 1D6F2% upper case rho (maths italic)
14171 \string<\glshex 1D6F4% upper case sigma (maths italic)
14172 \string<\glshex 1D6F5% upper case tau (maths italic)
14173 \string<\glshex 1D6F6% upper case upsilon (maths italic)
14174 \string<\glshex 1D6F7% upper case phi (maths italic)
14175 \string<\glshex 1D6F8% upper case chi (maths italic)
14176 \string<\glshex 1D6F9% upper case psi (maths italic)
14177 \string<\glshex 1D6FA% upper case omega (maths italic)
14178 }

```

UpperGreekIIrules Upper case only (doesn't include upright digamma).

```

14179 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
14180 \glshex 1D6E2% upper case alpha (maths italic)
14181 \string<\glshex 1D6E3% upper case beta (maths italic)
14182 \string<\glshex 1D6E4% upper case gamma (maths italic)
14183 \string<\glshex 1D6E5% upper case delta (maths italic)
14184 \string<\glshex 1D6E6% upper case epsilon (maths italic)
14185 \string<\glshex 1D6E7% upper case zeta (maths italic)
14186 \string<\glshex 1D6E8% upper case eta (maths italic)
14187 \string<\glshex 1D6E9% upper case theta (maths italic)
14188 \string<\glshex 1D6F3% upper case theta variant (maths italic)
14189 \string<\glshex 1D6EA% upper case iota (maths italic)
14190 \string<\glshex 1D6EB% upper case kappa (maths italic)

```

```

14191 \string<\glshex 1D6EC% upper case lambda (maths italic)
14192 \string<\glshex 1D6ED% upper case mu (maths italic)
14193 \string<\glshex 1D6EE% upper case nu (maths italic)
14194 \string<\glshex 1D6EF% upper case xi (maths italic)
14195 \string<\glshex 1D6F0% upper case omicron (maths italic)
14196 \string<\glshex 1D6F1% upper case pi (maths italic)
14197 \string<\glshex 1D6F2% upper case rho (maths italic)
14198 \string<\glshex 1D6F4% upper case sigma (maths italic)
14199 \string<\glshex 1D6F5% upper case tau (maths italic)
14200 \string<\glshex 1D6F6% upper case upsilon (maths italic)
14201 \string<\glshex 1D6F7% upper case phi (maths italic)
14202 \string<\glshex 1D6F8% upper case chi (maths italic)
14203 \string<\glshex 1D6F9% upper case psi (maths italic)
14204 \string<\glshex 1D6FA% upper case omega (maths italic)
14205 }

```

LowerGreekIrules Lower case only (includes upright digamma).

```

14206 \newcommand*{\glstrMathItalicLowerGreekIrules}{%
14207 \glshex 1D6FC% lower case alpha (maths italic)
14208 \string<\glshex 1D6FD% lower case beta (maths italic)
14209 \string<\glshex 1D6FE% lower case gamma (maths italic)
14210 \string<\glshex 1D6FF% lower case delta (maths italic)
14211 \string<\glshex 1D700% lower case epsilon (maths italic)
14212 \string=<\glshex 1D716% lower case epsilon variant (maths italic)
14213 \string<\glshex 03DD% lower case digamma
14214 \string<\glshex 1D701% lower case zeta (maths italic)
14215 \string<\glshex 1D702% lower case eta (maths italic)
14216 \string<\glshex 1D703% lower case theta (maths italic)
14217 \string=<\glshex 1D717% lower case theta variant (maths italic)
14218 \string<\glshex 1D704% lower case iota (maths italic)
14219 \string<\glshex 1D705% lower case kappa (maths italic)
14220 \string=<\glshex 1D718% lower case kappa variant (maths italic)
14221 \string<\glshex 1D706% lower case lambda (maths italic)
14222 \string<\glshex 1D707% lower case mu (maths italic)
14223 \string<\glshex 1D708% lower case nu (maths italic)
14224 \string<\glshex 1D709% lower case xi (maths italic)
14225 \string<\glshex 1D70A% lower case omicron (maths italic)
14226 \string<\glshex 1D70B% lower case pi (maths italic)
14227 \string=<\glshex 1D71B% lower case pi variant (maths italic)
14228 \string<\glshex 1D70C% lower case rho (maths italic)
14229 \string=<\glshex 1D71A% lower case rho variant (maths italic)
14230 \string<\glshex 1D70D% lower case final sigma (maths italic)
14231 \string=<\glshex 1D70E% lower case sigma (maths italic)
14232 \string<\glshex 1D70F% lower case tau (maths italic)
14233 \string<\glshex 1D710% lower case upsilon (maths italic)
14234 \string<\glshex 1D711% lower case phi (maths italic)
14235 \string=<\glshex 1D719% lower case phi variant (maths italic)
14236 \string<\glshex 1D712% lower case chi (maths italic)
14237 \string<\glshex 1D713% lower case psi (maths italic)

```

```

14238 \string<\glshex 1D714% lower case omega (maths italic)
14239 }

```

Lower case only (doesn't includes upright digamma).

```

14240 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
14241 \glshex 1D6FC% lower case alpha (maths italic)
14242 \string<\glshex 1D6FD% lower case beta (maths italic)
14243 \string<\glshex 1D6FE% lower case gamma (maths italic)
14244 \string<\glshex 1D6FF% lower case delta (maths italic)
14245 \string<\glshex 1D700% lower case epsilon (maths italic)
14246 \string=\glshex 1D716% lower case epsilon variant (maths italic)
14247 \string<\glshex 1D701% lower case zeta (maths italic)
14248 \string<\glshex 1D702% lower case eta (maths italic)
14249 \string<\glshex 1D703% lower case theta (maths italic)
14250 \string=\glshex 1D717% lower case theta variant (maths italic)
14251 \string<\glshex 1D704% lower case iota (maths italic)
14252 \string<\glshex 1D705% lower case kappa (maths italic)
14253 \string=\glshex 1D718% lower case kappa variant (maths italic)
14254 \string<\glshex 1D706% lower case lambda (maths italic)
14255 \string<\glshex 1D707% lower case mu (maths italic)
14256 \string<\glshex 1D708% lower case nu (maths italic)
14257 \string<\glshex 1D709% lower case xi (maths italic)
14258 \string<\glshex 1D70A% lower case omicron (maths italic)
14259 \string<\glshex 1D70B% lower case pi (maths italic)
14260 \string=\glshex 1D71B% lower case pi variant (maths italic)
14261 \string<\glshex 1D70C% lower case rho (maths italic)
14262 \string=\glshex 1D71A% lower case rho variant (maths italic)
14263 \string<\glshex 1D70D% lower case final sigma (maths italic)
14264 \string=\glshex 1D70E% lower case sigma (maths italic)
14265 \string<\glshex 1D70F% lower case tau (maths italic)
14266 \string<\glshex 1D710% lower case upsilon (maths italic)
14267 \string<\glshex 1D711% lower case phi (maths italic)
14268 \string=\glshex 1D719% lower case phi variant (maths italic)
14269 \string<\glshex 1D712% lower case chi (maths italic)
14270 \string<\glshex 1D713% lower case psi (maths italic)
14271 \string<\glshex 1D714% lower case omega (maths italic)
14272 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

14273 \newcommand*{\glxtrMathGreekIrules}{%
14274 \glxtrMathItalicAlpha
14275 \string;\glxtrUpAlpha
14276 \string<\glxtrMathItalicBeta
14277 \string;\glxtrUpBeta
14278 \string<\glxtrMathItalicGamma
14279 \string;\glxtrUpGamma
14280 \string<\glxtrMathItalicDelta
14281 \string;\glxtrUpDelta
14282 \string<\glxtrMathItalicEpsilon

```



```

14283 \string;\glxtrUpEpsilon
14284 \string<\glxtrUpDigamma
14285 \string<\glxtrMathItalicZeta
14286 \string;\glxtrUpZeta
14287 \string<\glxtrMathItalicEta
14288 \string;\glxtrUpEta
14289 \string<\glxtrMathItalicTheta
14290 \string;\glxtrUpTheta
14291 \string<\glxtrMathItalicIota
14292 \string;\glxtrUpIota
14293 \string<\glxtrMathItalicKappa
14294 \string;\glxtrUpKappa
14295 \string<\glxtrMathItalicLambda
14296 \string;\glxtrUpLambda
14297 \string<\glxtrMathItalicMu
14298 \string;\glxtrUpMu
14299 \string<\glxtrMathItalicNu
14300 \string;\glxtrUpNu
14301 \string<\glxtrMathItalicXi
14302 \string;\glxtrUpXi
14303 \string<\glxtrMathItalicOmicron
14304 \string;\glxtrUpOmicron
14305 \string<\glxtrMathItalicPi
14306 \string;\glxtrUpPi
14307 \string<\glxtrMathItalicRho
14308 \string;\glxtrUpRho
14309 \string<\glxtrMathItalicSigma
14310 \string;\glxtrUpSigma
14311 \string<\glxtrMathItalicTau
14312 \string;\glxtrUpTau
14313 \string<\glxtrMathItalicUpsilon
14314 \string;\glxtrUpUpsilon
14315 \string<\glxtrMathItalicPhi
14316 \string;\glxtrUpPhi
14317 \string<\glxtrMathItalicChi
14318 \string;\glxtrUpChi
14319 \string<\glxtrMathItalicPsi
14320 \string;\glxtrUpPsi
14321 \string<\glxtrMathItalicOmega
14322 \string;\glxtrUpOmega
14323 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

14324 \newcommand*{\glxtrMathGreekIIrules}{%
14325 \glxtrMathItalicAlpha
14326 \string;\glxtrUpAlpha
14327 \string<\glxtrMathItalicBeta
14328 \string;\glxtrUpBeta
14329 \string<\glxtrMathItalicGamma

```

```

14330 \string;\glsxtrUpGamma
14331 \string<\glsxtrMathItalicDelta
14332 \string;\glsxtrUpDelta
14333 \string<\glsxtrMathItalicEpsilon
14334 \string;\glsxtrUpEpsilon
14335 \string<\glsxtrMathItalicZeta
14336 \string;\glsxtrUpZeta
14337 \string<\glsxtrMathItalicEta
14338 \string;\glsxtrUpEta
14339 \string<\glsxtrMathItalicTheta
14340 \string;\glsxtrUpTheta
14341 \string<\glsxtrMathItalicIota
14342 \string;\glsxtrUpIota
14343 \string<\glsxtrMathItalicKappa
14344 \string;\glsxtrUpKappa
14345 \string<\glsxtrMathItalicLambda
14346 \string;\glsxtrUpLambda
14347 \string<\glsxtrMathItalicMu
14348 \string;\glsxtrUpMu
14349 \string<\glsxtrMathItalicNu
14350 \string;\glsxtrUpNu
14351 \string<\glsxtrMathItalicXi
14352 \string;\glsxtrUpXi
14353 \string<\glsxtrMathItalicOmicron
14354 \string;\glsxtrUpOmicron
14355 \string<\glsxtrMathItalicPi
14356 \string;\glsxtrUpPi
14357 \string<\glsxtrMathItalicRho
14358 \string;\glsxtrUpRho
14359 \string<\glsxtrMathItalicSigma
14360 \string;\glsxtrUpSigma
14361 \string<\glsxtrMathItalicTau
14362 \string;\glsxtrUpTau
14363 \string<\glsxtrMathItalicUpsilon
14364 \string;\glsxtrUpUpsilon
14365 \string<\glsxtrMathItalicPhi
14366 \string;\glsxtrUpPhi
14367 \string<\glsxtrMathItalicChi
14368 \string;\glsxtrUpChi
14369 \string<\glsxtrMathItalicPsi
14370 \string;\glsxtrUpPsi
14371 \string<\glsxtrMathItalicOmega
14372 \string;\glsxtrUpOmega
14373 }

```

`\glsxtrUpAlpha`

```

14374 \newcommand*{\glsxtrUpAlpha}{%
14375   \glshex 03B1,% lower case alpha
14376   \glshex 0391% upper case alpha

```

14377 }

\glxtrUpBeta

```
14378 \newcommand*{\glxtrUpBeta}{%
14379   \glshex 03B2,% lower case beta
14380   \glshex 0392% upper case beta
14381 }
```

\glxtrUpGamma

```
14382 \newcommand*{\glxtrUpGamma}{%
14383   \glshex 03B3,% lower case gamma
14384   \glshex 0393% upper case gamma
14385 }
```

\glxtrUpDelta

```
14386 \newcommand*{\glxtrUpDelta}{%
14387   \glshex 03B4,% lower case delta
14388   \glshex 0394% upper case delta
14389 }
```

glxtrUpEpsilon

```
14390 \newcommand*{\glxtrUpEpsilon}{%
14391   \glshex 03B5% lower case epsilon
14392   \string=\glshex 03F5,% lower case epsilon variant
14393   \glshex 0395% upper case epsilon
14394 }
```

glxtrUpDigamma

```
14395 \newcommand*{\glxtrUpDigamma}{%
14396   \glshex 03DD,% lower case digamma
14397   \glshex 03DC% upper case digamma
14398 }
```

\glxtrUpZeta

```
14399 \newcommand*{\glxtrUpZeta}{%
14400   \glshex 03B6,% lower case zeta
14401   \glshex 0396% upper case zeta
14402 }
```

\glxtrUpEta

```
14403 \newcommand*{\glxtrUpEta}{%
14404   \glshex 03B7,% lower case eta
14405   \glshex 0397% upper case eta
14406 }
```

\glxtrUpTheta

```
14407 \newcommand*{\glxtrUpTheta}{%
14408   \glshex 03B8% lower case theta
```

```

14409 \string=\glshex 03D1,% lower case theta variant
14410 \glshex 0398% upper case theta
14411 }

```

$\backslash\text{glxtrUpIota}$

```

14412 \newcommand*{\glxtrUpIota}{%
14413 \glshex 03B9,% lower case iota
14414 \glshex 0399% upper case iota
14415 }

```

$\backslash\text{glxtrUpKappa}$

```

14416 \newcommand*{\glxtrUpKappa}{%
14417 \glshex 03BA,% lower case kappa
14418 \string=\glshex 03F0,% lower case kappa variant
14419 \glshex 039A% upper case kappa
14420 }

```

$\backslash\text{glxtrUpLambda}$

```

14421 \newcommand*{\glxtrUpLambda}{%
14422 \glshex 03BB,% lower lambda
14423 \glshex 039B% upper case lambda
14424 }

```

$\backslash\text{glxtrUpMu}$

```

14425 \newcommand*{\glxtrUpMu}{%
14426 \glshex 03BC,% lower case mu
14427 \glshex 039C% upper case mu
14428 }

```

$\backslash\text{glxtrUpNu}$

```

14429 \newcommand*{\glxtrUpNu}{%
14430 \glshex 03BD,% lower case nu
14431 \glshex 039D% upper case nu
14432 }

```

$\backslash\text{glxtrUpXi}$

```

14433 \newcommand*{\glxtrUpXi}{%
14434 \glshex 03BE,% lower case xi
14435 \glshex 039E% upper case xi
14436 }

```

glxtrUpOmicron

```

14437 \newcommand*{\glxtrUpOmicron}{%
14438 \glshex 03BF,% lower case omicron
14439 \glshex 039F% upper case omicron
14440 }

```

\glxtrUpPi

```
14441 \newcommand*{\glxtrUpPi}{%
14442 \glshex 03C0% lower case pi
14443 \string=\glshex 03D6,% lower case pi variant
14444 \glshex 03A0% upper case pi
14445 }
```

\glxtrUpRho

```
14446 \newcommand*{\glxtrUpRho}{%
14447 \glshex 03C1% lower case rho
14448 \string=\glshex 03F1,% lower case rho variant
14449 \glshex 03A1% upper case rho
14450 }
```

\glxtrUpSigma

```
14451 \newcommand*{\glxtrUpSigma}{%
14452 \glshex 03C2% lower case sigma
14453 \string=\glshex 03C3,% lower case sigma
14454 \glshex 03A3% upper case sigma
14455 }
```

\glxtrUpTau

```
14456 \newcommand*{\glxtrUpTau}{%
14457 \glshex 03C4,% lower case tau
14458 \glshex 03A4% upper case tau
14459 }
```

\glxtrUpUpsilon

```
14460 \newcommand*{\glxtrUpUpsilon}{%
14461 \glshex 03C5,% lower case upsilon
14462 \glshex 03A5% upper case upsilon
14463 }
```

\glxtrUpPhi

```
14464 \newcommand*{\glxtrUpPhi}{%
14465 \glshex 03C6% lower case phi
14466 \string=\glshex 03D5,% lower case phi variant
14467 \glshex 03A6% upper case phi
14468 }
```

\glxtrUpChi

```
14469 \newcommand*{\glxtrUpChi}{%
14470 \glshex 03C7,% lower case chi
14471 \glshex 03A7% upper case chi
14472 }
```

\glxtrUpPsi

```
14473 \newcommand*{\glxtrUpPsi}{%
```

```

14474 \glshex 03C8,% lower case psi
14475 \glshex 03A8% upper case psi
14476 }

```

$\backslash\mathrm{glxtrUpOmega}$

```

14477 \newcommand*{\glxtrUpOmega}{%
14478 \glshex 03C9,% lower case omega
14479 \glshex 03A9% upper case omega
14480 }

```

$\mathrm{MathItalicAlpha}$

```

14481 \newcommand*{\glxtrMathItalicAlpha}{%
14482 \glshex 1D6FC,% lower case alpha (maths italic)
14483 \glshex 1D6E2% upper case alpha (maths italic)
14484 }

```

$\mathrm{rMathItalicBeta}$

```

14485 \newcommand*{\glxtrMathItalicBeta}{%
14486 \glshex 1D6FD,% lower case beta (maths italic)
14487 \glshex 1D6E3% upper case beta (maths italic)
14488 }

```

$\mathrm{MathItalicGamma}$

```

14489 \newcommand*{\glxtrMathItalicGamma}{%
14490 \glshex 1D6FE,% lower case gamma (maths italic)
14491 \glshex 1D6E4% upper case gamma (maths italic)
14492 }

```

$\mathrm{MathItalicDelta}$

```

14493 \newcommand*{\glxtrMathItalicDelta}{%
14494 \glshex 1D6FF,% lower case delta (maths italic)
14495 \glshex 1D6E5% upper case delta (maths italic)
14496 }

```

$\mathrm{thItalicEpsilon}$

```

14497 \newcommand*{\glxtrMathItalicEpsilon}{%
14498 \glshex 1D700% lower case epsilon (maths italic)
14499 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
14500 \glshex 1D6E6% upper case epsilon (maths italic)
14501 }

```

$\mathrm{rMathItalicZeta}$

```

14502 \newcommand*{\glxtrMathItalicZeta}{%
14503 \glshex 1D701,% lower case zeta (maths italic)
14504 \glshex 1D6E7% upper case zeta (maths italic)
14505 }

```

trMathItalicEta

```
14506 \newcommand*{\glxtrMathItalicEta}{%
14507   \glshex 1D702,% lower case eta (maths italic)
14508   \glshex 1D6E8% upper case eta (maths italic)
14509 }
```

MathItalicTheta

```
14510 \newcommand*{\glxtrMathItalicTheta}{%
14511   \glshex 1D703% lower case theta (maths italic)
14512   \string=\glshex 1D717,% lower case theta variant (maths italic)
14513   \glshex 1D6E9% upper case theta (maths italic)
14514   \string=\glshex 1D6F3% upper case theta variant (maths italic)
14515 }
```

rMathItalicIota

```
14516 \newcommand*{\glxtrMathItalicIota}{%
14517   \glshex 1D704,% lower case iota (maths italic)
14518   \glshex 1D6EA% upper case iota (maths italic)
14519 }
```

MathItalicKappa

```
14520 \newcommand*{\glxtrMathItalicKappa}{%
14521   \glshex 1D705% lower case kappa (maths italic)
14522   \string=\glshex 1D718,% lower case kappa variant (maths italic)
14523   \glshex 1D6EB% upper case kappa (maths italic)
14524 }
```

athItalicLambda

```
14525 \newcommand*{\glxtrMathItalicLambda}{%
14526   \glshex 1D706,% lower case lambda (maths italic)
14527   \glshex 1D6EC% upper case lambda (maths italic)
14528 }
```

xtrMathItalicMu

```
14529 \newcommand*{\glxtrMathItalicMu}{%
14530   \glshex 1D707,% lower case mu (maths italic)
14531   \glshex 1D6ED% upper case mu (maths italic)
14532 }
```

xtrMathItalicNu

```
14533 \newcommand*{\glxtrMathItalicNu}{%
14534   \glshex 1D708,% lower case nu (maths italic)
14535   \glshex 1D6EE% upper case nu (maths italic)
14536 }
```

xtrMathItalicXi

```
14537 \newcommand*{\glxtrMathItalicXi}{%
14538   \glshex 1D709,% lower case xi (maths italic)
```

```

14539 \glshex 1D6EF% upper case xi (maths italic)
14540 }

thItalicOmicron

14541 \newcommand*{\glxtrMathItalicOmicron}{%
14542 \glshex 1D70A,% lower case omicron (maths italic)
14543 \glshex 1D6F0% upper case omicron (maths italic)
14544 }

xtrMathItalicPi

14545 \newcommand*{\glxtrMathItalicPi}{%
14546 \glshex 1D70B% lower case pi (maths italic)
14547 \string=\glshex 1D71B,% lower case pi variant (maths italic)
14548 \glshex 1D6F1% upper case pi (maths italic)
14549 }

trMathItalicRho

14550 \newcommand*{\glxtrMathItalicRho}{%
14551 \glshex 1D70C% lower case rho (maths italic)
14552 \string=\glshex 1D71A,% lower case rho variant (maths italic)
14553 \glshex 1D6F2% upper case rho (maths italic)
14554 }

MathItalicSigma

14555 \newcommand*{\glxtrMathItalicSigma}{%
14556 \glshex 1D70D% lower case final sigma (maths italic)
14557 \string=\glshex 1D70E,% lower case sigma (maths italic)
14558 \glshex 1D6F4% upper case sigma (maths italic)
14559 }

trMathItalicTau

14560 \newcommand*{\glxtrMathItalicTau}{%
14561 \glshex 1D70F,% lower case tau (maths italic)
14562 \glshex 1D6F5% upper case tau (maths italic)
14563 }

thItalicUpsilon

14564 \newcommand*{\glxtrMathItalicUpsilon}{%
14565 \glshex 1D710,% lower case upsilon (maths italic)
14566 \glshex 1D6F6% upper case upsilon (maths italic)
14567 }

trMathItalicPhi

14568 \newcommand*{\glxtrMathItalicPhi}{%
14569 \glshex 1D711% lower case phi (maths italic)
14570 \string=\glshex 1D719,% lower case phi variant (maths italic)
14571 \glshex 1D6F7% upper case phi (maths italic)
14572 }

```


trMathItalicChi

```
14573 \newcommand*{\glxtrMathItalicChi}{%
14574   \glshex 1D712,% lower case chi (maths italic)
14575   \glshex 1D6F8% upper case chi (maths italic)
14576 }
```

trMathItalicPsi

```
14577 \newcommand*{\glxtrMathItalicPsi}{%
14578   \glshex 1D713,% lower case psi (maths italic)
14579   \glshex 1D6F9% upper case psi (maths italic)
14580 }
```

MathItalicOmega

```
14581 \newcommand*{\glxtrMathItalicOmega}{%
14582   \glshex 1D714,% lower case omega (maths italic)
14583   \glshex 1D6FA% upper case omega (maths italic)
14584 }
```

thItalicPartial

```
14585 \newcommand*{\glxtrMathItalicPartial}{%
14586   \glshex 1D715% partial differential (maths italic)
14587 }
```

MathItalicNabla

```
14588 \newcommand*{\glxtrMathItalicNabla}{%
14589   \glshex 1D6FB% nabla (maths italic)
14590 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
14591 \newcommand*{\glxtrdigitrules}{%
14592   0\string=\glshex 2080\string=\glshex 2070
14593   \string<1\string=\glshex 2081\string=\glshex 00B9
14594   \string<2\string=\glshex 2082\string=\glshex 00B2
14595   \string<3\string=\glshex 2083\string=\glshex 00B3
14596   \string<4\string=\glshex 2084\string=\glshex 2074
14597   \string<5\string=\glshex 2085\string=\glshex 2075
14598   \string<6\string=\glshex 2086\string=\glshex 2076
14599   \string<7\string=\glshex 2087\string=\glshex 2077
14600   \string<8\string=\glshex 2088\string=\glshex 2078
14601   \string<9\string=\glshex 2089\string=\glshex 2079
14602 }
```

BasicDigitrules Digits from the Basic Latin set.

```
14603 \newcommand*{\glxtrBasicDigitrules}{%
14604   0\string<1\string<2\string<3\string<4%
14605   \string<5\string<6\string<7\string<8\string<9%
14606 }
```

criptDigitrules Subscript digits.

```
14607 \newcommand*{\glxtrSubScriptDigitrules}{%
14608 \glshex 2080% subscript 0
14609 \string<\glshex 2081% subscript 1
14610 \string<\glshex 2082% subscript 2
14611 \string<\glshex 2083% subscript 3
14612 \string<\glshex 2084% subscript 4
14613 \string<\glshex 2085% subscript 5
14614 \string<\glshex 2086% subscript 6
14615 \string<\glshex 2087% subscript 7
14616 \string<\glshex 2088% subscript 8
14617 \string<\glshex 2089% subscript 9
14618 }
```

criptDigitrules Superscript digits.

```
14619 \newcommand*{\glxtrSuperScriptDigitrules}{%
14620 \glshex 2070% superscript 0
14621 \string<\glshex 00B9% superscript 1
14622 \string<\glshex 00B2% superscript 2
14623 \string<\glshex 00B3% superscript 3
14624 \string<\glshex 2074% superscript 4
14625 \string<\glshex 2075% superscript 5
14626 \string<\glshex 2076% superscript 6
14627 \string<\glshex 2077% superscript 7
14628 \string<\glshex 2078% superscript 8
14629 \string<\glshex 2079% superscript 9
14630 }
```

trfractionrules Vulgar fractions.

```
14631 \newcommand*{\glxtrfractionrules}{%
14632 \glshex 215F% fraction numerator one (1/)
14633 \string<\glshex 2189% zero thirds (0/3 = 0)
14634 \string<\glshex 2152% one tenth (1/10 = 0.1)
14635 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
14636 \string<\glshex 215B% one eighth (1/8 = 0.125)
14637 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
14638 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
14639 \string<\glshex 2155% one fifth (1/5 = 0.2)
14640 \string<\glshex 00BC% one quarter (1/4 = 0.25)
14641 \string<\glshex 2153% one third (1/3 ~ 0.333)
14642 \string<\glshex 215C% three eighths (3/8 = 0.375)
14643 \string<\glshex 2156% two fifths (2/5 = 0.4)
14644 \string<\glshex 00BD% one half (1/2 = 0.5)
14645 \string<\glshex 2157% three fifths (3/5 = 0.6)
14646 \string<\glshex 215D% five eighths (5/8 = 0.625)
14647 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
14648 \string<\glshex 00BE% three quarters (3/4 = 0.75)
14649 \string<\glshex 2158% four fifths (4/5 = 0.8)
14650 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
```

```

14651 \string<\glshex 215E% seven eighths (7/8 = 0.875)
14652 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

14653 \renewcommand{\@glsxtrdialecthook}{%
14654   \ifundef\CurrentTrackedScript
14655   {%
14656     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
14657     {%
14658       \edef\CurrentTrackedScript{%
14659         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
14660     }%
14661   }%
14662 }%
14663 {}%
14664 \ifdef\CurrentTrackedScript
14665 {%
14666   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
14667   \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
14668   \let\CurrentTrackedTag\CurrentTrackedScript
14669   \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
14670   {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
14671   {}%
14672   \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
14673 }%
14674 {}%
14675 }

```

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.

```

14676 \ifdef\glsxtr@loaddialect
14677 {%
14678   \@ifpackageloaded{tracklang}
14679   {%
14680     \AnyTrackedLanguages
14681     {%
14682       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
14683     }%
14684   }%
14685 }
14686 {}
14687 }
14688 {}

```

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:

```
14689 \NeedsTeXFormat{LaTeX2e}
14690 \ProvidesPackage{glossaries-extra-stylemods}[2020/03/23 v1.44 (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`

```
14691 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
14692 \DeclareOption{all}{%
14693   \appto\@glxtr@loadstyles{%
14694     \RequirePackage{glossary-inline}%
14695     \RequirePackage{glossary-list}%
14696     \RequirePackage{glossary-tree}%
14697     \RequirePackage{glossary-mcols}%
14698     \RequirePackage{glossary-long}%
14699     \RequirePackage{glossary-longragged}%
14700     \RequirePackage{glossary-longbooktabs}%
14701     \RequirePackage{glossary-super}%
14702     \RequirePackage{glossary-superragged}%
14703     \RequirePackage{glossary-bookindex}%
14704     \RequirePackage{glossary-longextra}%
14705     \RequirePackage{glossary-topic}%
14706   }
14707 }

14708 \DeclareOption*{%
14709   \IfFileExists{glossary-\CurrentOption.sty}
14710   {\eappto\@glxtr@loadstyles{%
14711     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
14712   }%
```

```

14713   {%
14714       \PackageError{glossaries-extra-styles}%
14715       {Unknown option ‘\CurrentOption’}{}%
14716   }%
14717 }

```

Process the package options:

```
14718 \ProcessOptions
```

Load the required packages:

```
14719 \@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.

```
14720 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

ewglossarystyle

```

14721 \providecommand{\renewglossarystyle}[2]{%
14722   \ifcsundef{@glsstyle@#1}%
14723   {%
14724     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
14725   }%
14726   {%
14727     \csdef{@glsstyle@#1}{#2}%
14728   }%
14729 }

```

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.

```

14730 \ifdef{\@glsstyle@listdotted}
14731 {%
14732   \renewglossarystyle{listdotted}{%
14733     \setglossarystyle{list}%
14734     \renewcommand*{\glossentry}[2]{%
14735       \item[]\makebox[\glslistdottedwidth][l]{%
14736         \glsentryitem{##1}%
14737         \glstarget{##1}{\glossentryname{##1}}%
14738         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
14739         \glossentrydesc{##1}\glspostdescription}%
14740     \renewcommand*{\subglossentry}[3]{%
14741       \item[]\makebox[\glslistdottedwidth][l]{%
14742         \glssubentryitem{##2}%
14743         \glstarget{##2}{\glossentryname{##2}}%
14744         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%

```

```

14745 \glossentrydesc{##2}\glspostdescription}%
14746 }
14747 }
14748 {%

```

Assume the style isn't required if it hasn't already been defined.

```

14749 }

```

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.

```

14750 \ifdef{\@glsstyle@list}
14751 {%

```

listprelocation Space before number list for top-level entries.

```

14752 \newcommand{\glslistprelocation}{\glstrprelocation}

```

childprelocation Space before number list for child entries.

```

14753 \newcommand{\glslistchildprelocation}{\glslistprelocation}

```

childpostlocation Full stop after number list.

```

14754 \newcommand{\glslistchildpostlocation}{.}

```

\glslistdesc

```

14755 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}

```

lslistgroupskip

```

14756 \newcommand{\glslistgroupskip}{\nobreak\indexspace\nobreak}

```

Redefine list to use these commands.

```

14757 \renewglossarystyle{list}{%
14758 \renewenvironment{theglossary}%
14759 {\begin{description}}{\end{description}}%
14760 \renewcommand*\{glossaryheader}{}%
14761 \renewcommand*\{glsgroupheading}[1]{}%
14762 \renewcommand*\{glossentry}[2]{%
14763 \item[\glsentryitem{##1}]%
14764 \glstarget{##1}{\glossentryname{##1}}]
14765 \glslistdesc{##1}\glslistprelocation ##2}%
14766 \renewcommand*\{subglossentry}[3]{%
14767 \glssubentryitem{##2}%
14768 \glstarget{##2}{\strut}\space
14769 \glslistdesc{##2}%
14770 \glslistchildprelocation ##3\glslistchildpostlocation}%
14771 \renewcommand*\{glsgroupskip}{\ifglsnogroupskip\else\glslistgroupskip\fi}%
14772 }
14773 }
14774 {}

```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```

14775 \ifdef{\@glsstyle@altlist}
14776 {%
14777   \renewglossarystyle{altlist}{%
14778     \setglossarystyle{list}%
14779     \renewcommand*{\glossentry}[2]{%
14780       \item[\glsentryitem{##1}]%
14781       \glstarget{##1}{\glossentryname{##1}}}%
14782     \mbox{}\par\nobreak\@afterheading
14783     \glslistdesc{##1}\glslistprelocation ##2}%
14784   \renewcommand{\subglossentry}[3]{%
14785     \par
14786     \glssubentryitem{##2}%
14787     \glstarget{##2}{\strut}\glslistdesc{##2}%
14788     \glslistchildprelocation ##3}%
14789   }
14790 }
14791 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

14792 \ifdef{\@glsstyle@listgroup}
14793 {%
14794   \renewglossarystyle{listgroup}{%
14795     \setglossarystyle{list}%
14796     \renewcommand*{\glsgroupheading}[1]{%
14797       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
14798       \mbox{}\par\nobreak\@afterheading
14799     }%
14800   }
14801 }
14802 {}

```

Similarly for listhypergroup.

```

14803 \ifdef{\@glsstyle@listhypergroup}
14804 {%
14805   \renewglossarystyle{listhypergroup}{%
14806     \setglossarystyle{list}%
14807     \renewcommand*{\glossaryheader}{%
14808       \glslistnavigationitem{\glsnavigation}}%
14809     \renewcommand*{\glsgroupheading}[1]{%
14810       \item[\glslistgroupheaderfmt
14811         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
14812       \mbox{}\par\nobreak\@afterheading
14813     }%
14814   }
14815 }
14816 {}

```

Similarly for altlistgroup.

```

14817 \ifdef{\@glsstyle@altlistgroup}

```

```

14818 {%
14819   \renewglossarystyle{altlistgroup}{%
14820     \setglossarystyle{altlist}%
14821     \renewcommand*{\glsgroupheading}[1]{%
14822       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
14823       \mbox{}\par\nobreak\@afterheading
14824     }%
14825   }
14826 }
14827 {}

```

Similarly for altlisthypergroup.

```

14828 \ifdef{\@glsstyle@altlisthypergroup}
14829 {%
14830   \renewglossarystyle{altlisthypergroup}{%
14831     \setglossarystyle{altlist}%
14832     \renewcommand*{\glossaryheader}{%
14833       \glslistnavigationitem{\glsnavigation}}%
14834     \renewcommand*{\glsgroupheading}[1]{%
14835       \item[\glslistgroupheaderfmt
14836         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
14837       \mbox{}\par\nobreak\@afterheading
14838     }%
14839   }
14840 }
14841 {}

```

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`.

```

14842 \ifcsdef{\@glsstyle@long}
14843 {%
14844   \renewglossarystyle{long}{%
14845     \renewenvironment{theglossary}%
14846       {\begin{longtable}{lp{\glsdescwidth}}}%
14847       {\end{longtable}}%
14848     \renewcommand*{\glossaryheader}{}%
14849     \renewcommand*{\glsgroupheading}[1]{}%
14850     \renewcommand{\glossentry}[2]{%
14851       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14852       \glossentrydesc{##1}\glspostdescription
14853       \glstrprelocation ##2\tabularnewline
14854     }%
14855     \renewcommand{\subglossentry}[3]{%
14856       &
14857       \glssubentryitem{##2}%
14858       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription

```



```

14859     \glstrprelocation ##3\tabularnewline
14860 }%
14861 \ifglsnogroupskip
14862     \renewcommand*{\glsgroupskip}{}%
14863 \else
14864     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
14865 \fi
14866 }
14867 }
14868 {}

```

Three column style:

```

14869 \ifcsdef{@glstyle@long3col}
14870 {%
14871     \renewglossarystyle{long3col}{%
14872         \renewenvironment{theglossary}%
14873             {\begin{longtable}\lp{\glstdescwidth}p{\glspagelistwidth}}}%
14874             {\end{longtable}}}%
14875     \renewcommand*{\glossaryheader}{}%
14876     \renewcommand*{\glsgroupheading}[1]{}%
14877     \renewcommand{\glossentry}[2]{%
14878         \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14879         \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14880     }%
14881     \renewcommand{\subglossentry}[3]{%
14882         &
14883         \glssubentryitem{##2}%
14884         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14885         ##3\tabularnewline
14886     }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

14887     \ifglsnogroupskip
14888         \renewcommand*{\glsgroupskip}{}%
14889     \else
14890         \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
14891     \fi
14892 }
14893 }
14894 {}

```

Four column style:

```

14895 \ifcsdef{@glstyle@long4col}
14896 {%
14897     \renewglossarystyle{long4col}{%
14898         \renewenvironment{theglossary}%
14899             {\begin{longtable}\llll}}}%
14900             {\end{longtable}}}%
14901     \renewcommand*{\glossaryheader}{}%
14902     \renewcommand*{\glsgroupheading}[1]{}%

```

```

14903 \renewcommand{\glossentry}[2]{%
14904   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14905   \glossentrydesc{##1}\glspostdescription &
14906   \glossentrysymbol{##1} &
14907   ##2\tabularnewline
14908 }%
14909 \renewcommand{\subglossentry}[3]{%
14910   &
14911   \glssubentryitem{##2}%
14912   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14913   \glossentrysymbol{##2} & ##3\tabularnewline
14914 }%

14915 \ifglsnogroupskip
14916   \renewcommand*\{\glsgroupskip}{}%
14917 \else
14918   \renewcommand*\{\glsgroupskip}{& & \tabularnewline}%
14919 \fi
14920 }
14921 }
14922 {}

```

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`.

```

14923 \ifcsdef{@glstyle@longragged}
14924 {%
14925   \renewglossarystyle{longragged}{%
14926     \renewenvironment{theglossary}%
14927       {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%
14928       {\end{longtable}}%
14929     \renewcommand*\{\glossaryheader}{}%
14930     \renewcommand*\{\glsgroupheading}[1]{}%
14931     \renewcommand{\glossentry}[2]{%
14932       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14933       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
14934       \tabularnewline
14935     }%
14936     \renewcommand{\subglossentry}[3]{%
14937       &
14938       \glssubentryitem{##2}%
14939       \glstarget{##2}{\strut}\glossentrydesc{##2}%
14940       \glspostdescription\glxtrprelocation ##3%
14941       \tabularnewline

```

```

14942 }%
14943 \ifglsnogroupskip
14944   \renewcommand*{\glsgroupskip}{}%
14945 \else
14946   \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
14947 \fi
14948 }
14949 }
14950 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

14951 \ifcsdef{@glstyle@longragged3col}
14952 {%
14953   \renewglossarystyle{longragged3col}{%
14954     \renewenvironment{theglossary}%
14955       {\begin{longtable}{l>{\raggedright}p{\glsglwdescwidth}%
14956         >{\raggedright}p{\glsglspagelistwidth}}}%
14957       {\end{longtable}}}%
14958   \renewcommand*{\glossaryheader}{}%
14959   \renewcommand*{\glsgroupheading}[1]{}%
14960   \renewcommand{\glossentry}[2]{%
14961     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14962     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14963   }%
14964   \renewcommand{\subglossentry}[3]{%
14965     &
14966     \glssubentryitem{##2}%
14967     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14968     ##3\tabularnewline
14969   }%
14970   \ifglsnogroupskip
14971     \renewcommand*{\glsgroupskip}{}%
14972   \else
14973     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14974   \fi
14975 }
14976 }
14977 {}

```

Four column style:

```

14978 \ifcsdef{@glstyle@altlongragged4col}
14979 {%
14980   \renewglossarystyle{altlongragged4col}{%
14981     \renewenvironment{theglossary}%
14982       {\begin{longtable}{l>{\raggedright}p{\glsglwdescwidth}l%
14983         >{\raggedright}p{\glsglspagelistwidth}}}%
14984       {\end{longtable}}}%
14985   \renewcommand*{\glossaryheader}{}%

```

```

14986 \renewcommand*\glsgroupheading}[1]{}%
14987 \renewcommand{\glossentry}[2]{%
14988     \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14989     \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
14990     ##2\tabularnewline
14991 }%
14992 \renewcommand{\subglossentry}[3]{%
14993     &
14994     \glssubentryitem{##2}%
14995     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14996     \glossentrysymbol{##2} & ##3\tabularnewline
14997 }%

14998 \ifglsnogroupskip
14999     \renewcommand*\glsgroupskip{}{}%
15000 \else
15001     \renewcommand*\glsgroupskip{}& & \tabularnewline}%
15002 \fi
15003 }
15004 }
15005 {}

```

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.

```

15006 \ifcsdef{@glstyle@super}
15007 {%
15008     \renewglossarystyle{super}{%
15009         \renewenvironment{theglossary}%
15010             {\tablehead{}\tabletail{}}%
15011             \begin{supertabular}{\lp{\glsdescwidth}}}%
15012             {\end{supertabular}}%
15013     \renewcommand*\glossaryheader{}{}%
15014     \renewcommand*\glsgroupheading}[1]{}%
15015     \renewcommand{\glossentry}[2]{%
15016         \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15017         \glossentrydesc{##1}\glspostdescription
15018         \glxtrprelocation ##2\tabularnewline
15019     }%
15020     \renewcommand{\subglossentry}[3]{%
15021         &
15022         \glssubentryitem{##2}%
15023         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
15024         \glxtrprelocation ##3\tabularnewline
15025     }%
15026     \ifglsnogroupskip
15027         \renewcommand*\glsgroupskip{}{}%

```

```

15028 \else
15029 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
15030 \fi
15031 }
15032 }
15033 {}

```

Three column style:

```

15034 \ifcsdef{@glsstyle@super3col}
15035 {%
15036 \renewglossarystyle{super3col}{%
15037 \renewenvironment{theglossary}%
15038 {\tablehead{}\tabletail}%
15039 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
15040 {\end{supertabular}}%
15041 \renewcommand*{\glossaryheader}{}%
15042 \renewcommand*{\glsgroupheading}[1]{}%
15043 \renewcommand{\glossentry}[2]{%
15044 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15045 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
15046 }%
15047 \renewcommand{\subglossentry}[3]{%
15048 &
15049 \glssubentryitem{##2}%
15050 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
15051 ##3\tabularnewline
15052 }%
15053 \ifglsnogroupskip
15054 \renewcommand*{\glsgroupskip}{}%
15055 \else
15056 \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
15057 \fi
15058 }
15059 }
15060 {}

```

Four column styles:

```

15061 \ifcsdef{@glsstyle@super4col}
15062 {%
15063 \renewglossarystyle{super4col}{%
15064 \renewenvironment{theglossary}%
15065 {\tablehead{}\tabletail}%
15066 \begin{supertabular}{llll}}{%
15067 \end{supertabular}}%
15068 \renewcommand*{\glossaryheader}{}%
15069 \renewcommand*{\glsgroupheading}[1]{}%
15070 \renewcommand{\glossentry}[2]{%
15071 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15072 \glossentrydesc{##1}\glspostdescription &

```

```

15073     \glossentrysymbol{##1} & ##2\tabularnewline
15074 }%
15075 \renewcommand{\subglossentry}[3]{%
15076     &
15077     \glssubentryitem{##2}%
15078     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
15079     \glossentrysymbol{##2} & ##3\tabularnewline
15080 }%

15081 \ifglsnogroupskip
15082     \renewcommand*{\glsgroupskip}{}%
15083 \else
15084     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
15085 \fi
15086 }
15087 }
15088 {}

```

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`.

```

15089 \ifcsdef{@glstyle@superragged}
15090 {%
15091     \renewglossarystyle{superragged}{%
15092         \renewenvironment{theglossary}%
15093             {\tablehead{}\tabletail}%
15094             \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
15095             {\end{supertabular}}%
15096     \renewcommand*{\glossaryheader}{}%
15097     \renewcommand*{\glsgroupheading}[1]{}%
15098     \renewcommand{\glossentry}[2]{%
15099         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15100         \glossentrydesc{##1}\glspostdescription\glxstrprelocation ##2%
15101         \tabularnewline
15102     }%
15103     \renewcommand{\subglossentry}[3]{%
15104         &
15105         \glssubentryitem{##2}%
15106         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
15107         \glxstrprelocation ##3%
15108         \tabularnewline
15109     }%
15110     \ifglsnogroupskip
15111         \renewcommand*{\glsgroupskip}{}%
15112     \else
15113         \renewcommand*{\glsgroupskip}{& \tabularnewline}%

```

```

15114 \fi
15115 }
15116 }
15117 {}

```

Three column style:

```

15118 \ifcsdef{@glsstyle@superragged3col}
15119 {%
15120 \renewglossarystyle{superragged3col}{%
15121 \renewenvironment{theglossary}%
15122 {\tablehead{}}\tabletail{}}%
15123 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
15124 >{\raggedright}p{\glspagelistwidth}}}%
15125 {\end{supertabular}}%
15126 \renewcommand*{\glossaryheader}{}%
15127 \renewcommand*{\glsgroupheading}[1]{}%
15128 \renewcommand{\glossentry}[2]{%
15129 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15130 \glossentrydesc{##1}\glspostdescription &
15131 ##2\tabularnewline
15132 }%
15133 \renewcommand{\subglossentry}[3]{%
15134 &
15135 \glssubentryitem{##2}%
15136 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
15137 ##3\tabularnewline
15138 }%
15139 \ifglsnogroupskip
15140 \renewcommand*{\glsgroupskip}{}%
15141 \else
15142 \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
15143 \fi
15144 }
15145 }
15146 {}

```

Four columns:

```

15147 \ifcsdef{@glsstyle@altsuperragged4col}
15148 {%
15149 \renewglossarystyle{altsuperragged4col}{%
15150 \renewenvironment{theglossary}%
15151 {\tablehead{}}\tabletail{}}%
15152 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
15153 >{\raggedright}p{\glspagelistwidth}}}%
15154 {\end{supertabular}}%
15155 \renewcommand*{\glossaryheader}{}%
15156 \renewcommand{\glossentry}[2]{%
15157 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15158 \glossentrydesc{##1}\glspostdescription &

```

```

15159     \glossentrysymbol{##1} & ##2\tabularnewline
15160 }%
15161 \renewcommand{\subglossentry}[3]{%
15162     &
15163     \glssubentryitem{##2}%
15164     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
15165     \glossentrysymbol{##2} & ##3\tabularnewline
15166 }%

15167 \ifglsnogroupskip
15168     \renewcommand*{\glsgroupskip}{}%
15169 \else
15170     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
15171 \fi
15172 }
15173 }
15174 {}

```

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`.

```

15175 \ifdef{\@glsstyle@inline}
15176 {%
15177     \renewcommand*{\glspostinline}{.\spacefactor\sfcode{.}}

```

Just use `\glstrpostdescription` instead of `\glspostdescription`.

```

15178     \renewcommand*{\glsinlinedescformat}[3]{%
15179         \space#1\glstrpostdescription}
15180     \renewcommand*{\glsinlinesubdescformat}[3]{%
15181         #1\glstrpostdescription}

```

The default settings don't show the location lists, so there's no adjustment for `\glstrprelocation`.

```

15182 }
15183 {}

```

2.8 Tree Styles

Redefine both `\glstreenamfmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamfmt` to make it easier to change both at the same time or only change one without affecting the other.

```

15184 \ifdef\glstreenamfmt
15185 {

```

`\glstreedefaultnamfmt`

```

15186     \newcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}

```


`\glstreenamefmt`

```
15187 \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}
```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
15188 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreenavigationfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
15189 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreePreHeader` Takes the label as the first argument and title as the second argument so this can be modified to add a bookmark.

```
15190 \newcommand{\glstreePreHeader}[2]{}
```

```
15191 }
```

```
15192 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
15193 \ifdef{\@glsstyle@index}
```

```
15194 {
```

`\glstreeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```
15195 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

`\glstreechildprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```
15196 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Don't prohibit a page break at the start of a new group if there's no header.

`\glstreegroupskip`

```
15197 \newcommand{\glstreegroupskip}{\indexspace}
```

`\glstreegroupheaderskip` This doesn't include `\@afterheading` as it can cause interference with some styles.

```
15198 \newcommand{\glstreegroupheaderskip}{\nopagebreak\glstreegroupskip\nobreak}
```

Modify the index style.

```
15199 \renewglossarystyle{index}{%
15200   \renewenvironment{theglossary}%
15201     {\setlength{\parindent}{0pt}}%
15202     {\setlength{\parskip}{0pt plus 0.3pt}}%
15203     \let\item\glstreeitem
15204     \let\subitem\glstreesubitem
15205     \let\subsubitem\glstreesubsubitem
15206   }%
15207   {\par}%
15208   \renewcommand*{\glossaryheader}{}%
15209   \renewcommand*{\glsgroupheading}[1]{}%
15210   \renewcommand*{\glossentry}[2]{%
15211     \item\glsentryitem{#1}%
```

```

15212     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15213     \glstreesymbol{##1}%
15214     \glstreeDescLoc{##1}{##2}%
15215 }%
15216 \renewcommand{\subglossentry}[3]{%
15217     \ifcase##1\relax
15218     \item
15219     \or
15220     \subitem
15221     \glssubentryitem{##2}%
15222     \else
15223     \subsubitem
15224     \fi
15225     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
15226     \glstreechildsymbol{##2}%
15227     \glstreeChildDescLoc{##2}{##3}%
15228 }%
15229 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15230 }
15231 }
15232 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

15233 \ifdef{\@glsstyle@indexgroup}
15234 {%
15235     \renewglossarystyle{indexgroup}{%
15236         \setglossarystyle{index}%
15237         \renewcommand*{\glsgroupheading}[1]{%
15238             \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15239             \glstreePreHeader{##1}{\glsxtr@grptitle}%
15240             \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
15241             \glstreegroupheaderskip\@afterheading
15242         }%
15243     }
15244 }
15245 {}

```

Similarly for indexhypergroup.

```

15246 \ifdef{\@glsstyle@indexhypergroup}
15247 {%
15248     \renewglossarystyle{indexhypergroup}{%
15249         \setglossarystyle{index}%
15250         \renewcommand*{\glossaryheader}{%
15251             \item\glstreenavigationfmt{\glsnavigation}%
15252             \glstreegroupheaderskip\@afterheading}%
15253         \renewcommand*{\glsgroupheading}[1]{%
15254             \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15255             \glstreePreHeader{##1}{\glsxtr@grptitle}%
15256             \item\glstreegroupheaderfmt
15257                 {\glsnahypertarget{##1}{\glsxtr@grptitle}}%

```

```

15258      \glstreegroupheaderskip\@afterheading}%
15259    }%
15260  }
15261 {}

```

Adjust tree style to remove hard coded space before number list.

```

15262 \ifdef{\@glsstyle@tree}
15263 {%

```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```

15264 \newcommand{\glstreedesc}[1]{%
15265   \glstreepredesc\glossentrydesc{#1}\glspostdescription
15266 }

```

`\glstreeDescLoc`

`\glstreeDescLoc{<label>}{<location>}`

This checks for the description and symbol. If both are missing, a different separator may be required. For example, a comma and space if there's no description or symbol but just a space if either of those fields are present.

```

15267 \newcommand{\glstreeDescLoc}[2]{%
15268   \ifglshasdesc{#1}%
15269   {\glstreedesc{#1}\glstreeprelocation}%
15270   {\ifglshassymbol{#1}{\glstreeprelocation}{\glstreeNoDescSymbolPreLocation}}}%
15271   #2%
15272 }

```

`\glstreeNoDescSymbolPreLocation`

`\glstreeNoDescSymbolPreLocation`

```

15273 \newcommand{\glstreeNoDescSymbolPreLocation}{\space}

```

Similarly for the symbol.

`\glstreesymbol`

```

15274 \newcommand{\glstreesymbol}[1]{%
15275   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
15276 }%

```

And for the child entries:

lstreechilddesc

```
15277 \newcommand{\glstreechilddesc}[1]{%
15278   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
15279 }%
```

treeChildDescLoc

```
15280 \newcommand{\glstreeChildDescLoc}[2]{%
15281   \ifglshasdesc{#1}%
15282   {\glstreechilddesc{#1}\glstreechildprelocation}%
15283   {\ifglshassymbol{#1}{\glstreechildprelocation}%
15284    {\glstreeNoDescSymbolPreLocation}%
15285   }%
15286   #2%
15287 }%
```

treechildsymbol This just behaves in the same way as the top-level.

```
15288 \newcommand{\glstreechildsymbol}[1]{%
15289   \glstreesymbol{#1}%
15290 }%

15291 \renewglossarystyle{tree}{%
15292   \renewenvironment{theglossary}%
15293     {\setlength{\parindent}{0pt}%
15294      \setlength{\parskip}{0pt plus 0.3pt}}%
15295   {}%
15296   \renewcommand*{\glossaryheader}{}%
15297   \renewcommand*{\glsgroupheading}[1]{}%
15298   \renewcommand{\glossentry}[2]{%
15299     \hangindent0pt\relax
15300     \parindent0pt\relax
15301     \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15302     \glstreesymbol{##1}%
15303     \glstreeDescLoc{##1}{##2}\par
15304   }%
15305   \renewcommand{\subglossentry}[3]{%
15306     \hangindent##1\glstreeindent\relax
15307     \parindent##1\glstreeindent\relax
15308     \ifnum##1=1\relax
15309       \glssubentryitem{##2}%
15310     \fi
15311     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
15312     \glstreechildsymbol{##2}%
15313     \glstreeChildDescLoc{##2}{##3}\par
15314   }%
15315   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15316 }%
15317 }
15318 }
```

The treegroup style is redefined to discourage a page break after the heading.

```

15319 \ifdef{\@glsstyle@treegroup}
15320 {%
15321   \renewglossarystyle{treegroup}{%
15322     \setglossarystyle{tree}%
15323     \renewcommand{\glsgroupheading}[1]{%
15324       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15325       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15326       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15327       \glstreegroupheaderskip\@afterheading}%
15328   }
15329 }
15330 {}

```

Similarly for treehypergroup

```

15331 \ifdef{\@glsstyle@treehypergroup}
15332 {%
15333   \renewglossarystyle{treehypergroup}{%
15334     \setglossarystyle{tree}%
15335     \renewcommand*\glossaryheader{%
15336       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15337       \glstreegroupheaderskip\@afterheading}%
15338     \renewcommand*\glsgroupheading[1]{%
15339       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15340       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15341       \par\noindent
15342       \glstreegroupheaderfmt
15343       {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15344       \glstreegroupheaderskip\@afterheading}%
15345   }
15346 }
15347 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

15348 \ifdef{\@glsstyle@treenoname}
15349 {%

```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```

15350   \newcommand{\glstreenonamedesc}[1]{%
15351     \glstreepredesc\glossentrydesc{##1}\glspostdescription
15352   }%

```

Similarly for the symbol.

treenonamesymbol

```

15353   \newcommand{\glstreenonamesymbol}[1]{%
15354     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
15355   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

15356 \newcommand{\glstreenonamechilddesc}[1]{%
15357   \glossentrydesc{#1}\glspostdescription
15358 }%

15359 \renewglossarystyle{treenoname}{%
15360   \renewenvironment{theglossary}%
15361     {\setlength{\parindent}{0pt}%
15362      \setlength{\parskip}{0pt plus 0.3pt}}%
15363   {}%
15364   \renewcommand*\glossaryheader{}%
15365   \renewcommand*\glsgroupheading[1]{}%
15366   \renewcommand{\glossentry}[2]{%
15367     \hangindent0pt\relax
15368     \parindent0pt\relax
15369     \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15370     \glstreenonamesymbol{##1}%
15371     \glstreenonamedesc{##1}%
15372     \glstreeDescLoc{##1}{##2}\par
15373   }%
15374   \renewcommand{\subglossentry}[3]{%
15375     \hangindent##1\glstreeindent\relax
15376     \parindent##1\glstreeindent\relax
15377     \ifnum##1=1\relax
15378       \glssubentryitem{##2}%
15379     \fi
15380     \glstarget{##2}{\strut}%
15381     \glstreenonamechilddesc{##2}%
15382     \glstreechildprelocation##3\par
15383   }%
15384   \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15385 }
15386 }
15387 {}

```

The treenonamegroup style is redefined to discourage a page break after the heading.

```

15388 \ifdef{\@glssstyle@treenonamegroup}
15389 {%
15390   \renewglossarystyle{treenonamegroup}{%
15391     \setglossarystyle{treenoname}%
15392     \renewcommand{\glsgroupheading}[1]{%
15393       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15394       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15395       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15396       \glstreegroupheaderskip\@afterheading
15397     }%
15398   }
15399 }
15400 {}

```

Similarly for `treenonamehypergroup`

```

15401 \ifdef{\@glsstyle@treenonamehypergroup}
15402 {%
15403   \renewglossarystyle{treenonamehypergroup}{%
15404     \setglossarystyle{treenoname}%
15405     \renewcommand*{\glossaryheader}{%
15406       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15407       \glstreegroupheaderskip\@afterheading}%
15408     \renewcommand*{\glsgroupheading}[1]{%
15409       \glstrgetgrouptitle{##1}{\glstr@grptitle}%
15410       \glstreePreHeader{##1}{\glstr@grptitle}%
15411       \par\noindent
15412       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glstr@grptitle}}%
15413       \glstreegroupheaderskip\@afterheading}%
15414   }
15415 }
15416 {}

```

The `alttree` style is redefined to make it easier to made minor adjustments.

```

15417 \ifdef{\@glsstyle@alttree}
15418 {%

```

Only redefine this style if it's already been defined.

`bolDescLocation`

`\glstralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

15419 \newcommand{\glstralttreeSymbolDescLocation}[2]{%
15420   {%
15421     \let\par\glstrAltTreePar
15422     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
15423     \glstreeDescLoc{#1}{#2}\par
15424   }%
15425 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

15426 \newlength\glstrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

15427 \newcommand{\glstrAltTreePar}{%
15428   \@@par
15429   \glstrAltTreeSetHangIndent
15430   \setlength{\parindent}{\dimexpr\hangindent+\glstrAltTreeIndent}%
15431 }

```

bolDescLocation

`\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```
15432 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
15433   \glxtralttreeSymbolDescLocation{#2}{#3}%
15434 }
```

trreetopindent 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.

```
15435 \newlength\glxtrreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
15436 \newcommand*{\glxtralttreeInit}{%
15437   \settowidth{\glxtrreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
15438   \glxtrAltTreeIndent=\parindent
15439 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
15440 \newcommand*{\gglsetwidest}[2][0]{%
15441   \csgdef{@glswidestname\romannumeral#1}{#2}%
15442 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
15443 \newcommand*{\eglssetwidest}[2][0]{%
15444   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
15445 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
15446 \newcommand*{\xglsetwidest}[2][0]{%
15447   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
15448 }
```

glsupdatewidest Only sets if new value is wider than old value.

```
15449 \newcommand*{\glsupdatewidest}[2][0]{%
15450   \ifcsundef{@glswidestname\romannumeral#1}%
15451   {\csdef{@glswidestname\romannumeral#1}{#2}}%
15452   {%
15453     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15454     \settowidth{\dimen@ii}{#2}%
15455     \ifdim\dimen@ii>\dimen@
15456     \csdef{@glswidestname\romannumeral#1}{#2}%
15457     \fi
15458   }%
15459 }
```


glsupdatewidest As above but global definition.

```

15460 \newcommand*\glsupdatewidest[2][0]{%
15461   \ifcsundef{@glswidestname\romannumeral#1}%
15462   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
15463   {%
15464     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15465     \settowidth{\dimen@ii}{#2}%
15466     \ifdim\dimen@ii>\dimen@
15467     \csgdef{@glswidestname\romannumeral#1}{#2}%
15468     \fi
15469   }%
15470 }
```

glsupdatewidest As \glsupdatewidest but expands value.

```

15471 \newcommand*\eglsupdatewidest[2][0]{%
15472   \ifcsundef{@glswidestname\romannumeral#1}%
15473   {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
15474   {%
15475     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15476     \settowidth{\dimen@ii}{#2}%
15477     \ifdim\dimen@ii>\dimen@
15478     \protected@csedef{@glswidestname\romannumeral#1}{#2}%
15479     \fi
15480   }%
15481 }
```

glsupdatewidest As above but global.

```

15482 \newcommand*\xglsupdatewidest[2][0]{%
15483   \ifcsundef{@glswidestname\romannumeral#1}%
15484   {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
15485   {%
15486     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15487     \settowidth{\dimen@ii}{#2}%
15488     \ifdim\dimen@ii>\dimen@
15489     \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
15490     \fi
15491   }%
15492 }
```

glsgetwidestname Provide a user-level macro to obtain the widest top-level name.

```

15493 \newcommand*\glsgetwidestname{\@glswidestname}
```

glsgetwidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```

15494 \newcommand*\glsgetwidestsubname[1]{%
15495   \ifcsundef{@glswidestname\romannumeral#1}%
15496   {\@glswidestname}%
15497   {\csuse{@glswidestname\romannumeral#1}}%
15498 }
```

estTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of \glsfindwidesttoplevelname

```
15499 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

sedTopLevelName Like \glsfindwidesttoplevelname but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```
15500 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
15501   \dimen@=0pt\relax
15502   \gls@tmplen=0pt\relax
15503   \foralllglossaries[#1]{\@gls@type}%
15504   {%
15505     \forglentries[\@gls@type]{\@glo@label}%
15506     {%
15507       \ifglused{\@glo@label}%
15508       {%
15509         \ifglshasparent{\@glo@label}%
15510         {}%
15511         {%
15512           \settowidth{\dimen@}%
15513             {\glstreenamfmt{\glstryname{\@glo@label}}}%
15514           \ifdim\dimen@>\gls@tmplen
15515             \gls@tmplen=\dimen@
15516             \eglssetwidest{\glstryname{\@glo@label}}%
15517           \fi
15518         }%
15519       }%
15520     }%
15521   }%
15522 }%
15523 }
```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
15524 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
15525   \dimen@=0pt\relax
15526   \gls@tmplen=0pt\relax
15527   \foralllglossaries[#1]{\@gls@type}%
15528   {%
15529     \forglentries[\@gls@type]{\@glo@label}%
15530     {%
15531       \ifglused{\@glo@label}%
15532       {%
15533         \settowidth{\dimen@}%
15534           {\glstreenamfmt{\glstryname{\@glo@label}}}%
15535         \ifdim\dimen@>\gls@tmplen
15536           \gls@tmplen=\dimen@
15537         \eglssetwidest{\glstryname{\@glo@label}}%
15538       }%
15539     }%
15540   }%
15541 }
```

```

15538         \fi
15539     }%
15540     {}%
15541     }%
15542     }%
15543 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

15544 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
15545     \dimen@=0pt\relax
15546     \gls@tmplen=0pt\relax
15547     \forallglossaries[#1]{\@gls@type}%
15548     {%
15549         \forglsentries[\@gls@type]{\@glo@label}%
15550         {%
15551             \settowidth{\dimen@}%
15552             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15553             \ifdim\dimen@>\gls@tmplen
15554                 \gls@tmplen=\dimen@
15555                 \eglissetwidest{\glsentryname{\@glo@label}}%
15556             \fi
15557         }%
15558     }%
15559 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well.

Any entry that has a great-grandparent is ignored.

```

15560 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
15561     \dimen@=0pt\relax
15562     \dimen@i=0pt\relax
15563     \dimen@ii=0pt\relax
15564     \forallglossaries[#1]{\@gls@type}%
15565     {%
15566         \forglsentries[\@gls@type]{\@glo@label}%
15567         {%
15568             \ifglsused{\@glo@label}%
15569             {%
15570                 \ifglshasparent{\@glo@label}%
15571                 {%
15572                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
15573                     \ifglshasparent{\@glo@parent}%
15574                     {%
15575                         \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
15576                         \ifglshasparent{\@glo@parent}%
15577                     }%
15578                 }%
15579                 \settowidth{\gls@tmplen}%
15580                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15581                 \ifdim\gls@tmplen>\dimen@ii

```

```

15582         \dimen@ii=\gls@tmplen
15583         \eglssetwidest[2]{\glsentryname{\@glo@label}}}%
15584     \fi
15585 }%
15586 }%
15587 {%
15588     \settowidth{\gls@tmplen}%
15589     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15590     \ifdim\gls@tmplen>\dimen@i
15591         \dimen@i=\gls@tmplen
15592         \eglssetwidest[1]{\glsentryname{\@glo@label}}}%
15593     \fi
15594 }%
15595 }%
15596 {%
15597     \settowidth{\gls@tmplen}%
15598     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15599     \ifdim\gls@tmplen>\dimen@
15600         \dimen@=\gls@tmplen
15601         \eglssetwidest{\glsentryname{\@glo@label}}}%
15602     \fi
15603 }%
15604 }%
15605 {}%
15606 }%
15607 }%
15608 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

15609 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
15610     \dimen@=0pt\relax
15611     \dimen@i=0pt\relax
15612     \dimen@ii=0pt\relax
15613     \foralllglossaries[#1]{\@gls@type}%
15614     {%
15615         \forglsentries[\@gls@type]{\@glo@label}%
15616         {%
15617             \ifglshasparent{\@glo@label}%
15618             {%
15619                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
15620                 \ifglshasparent{\@glo@parent}%
15621                 {%
15622                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
15623                     \ifglshasparent{\@glo@parent}%
15624                     {}%
15625                     {%
15626                         \settowidth{\gls@tmplen}%
15627                         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15628                         \ifdim\gls@tmplen>\dimen@ii

```

```

15629         \dimen@ii=\gls@tmplen
15630         \eglssetwidest[2]{\glsentryname{\@glo@label}}}%
15631     \fi
15632 }%
15633 }%
15634 {%
15635     \settowidth{\gls@tmplen}%
15636         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15637     \ifdim\gls@tmplen>\dimen@i
15638         \dimen@i=\gls@tmplen
15639         \eglssetwidest[1]{\glsentryname{\@glo@label}}}%
15640     \fi
15641 }%
15642 }%
15643 {%
15644     \settowidth{\gls@tmplen}%
15645         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15646     \ifdim\gls@tmplen>\dimen@
15647         \dimen@=\gls@tmplen
15648         \eglssetwidest{\glsentryname{\@glo@label}}}%
15649     \fi
15650 }%
15651 }%
15652 }%
15653 }

```

`\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.

```

15654 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
15655     \dimen@=0pt\relax
15656     \gls@tmplen=0pt\relax
15657     #2=0pt\relax
15658     \foralllglossaries[#1]{\@gls@type}%
15659     {%
15660         \forglsentries[\@gls@type]{\@glo@label}%
15661         {%
15662             \ifglsused{\@glo@label}%
15663             {%
15664                 \settowidth{\dimen@}%
15665                     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15666                 \ifdim\dimen@>\gls@tmplen
15667                     \gls@tmplen=\dimen@
15668                     \eglssetwidest{\glsentryname{\@glo@label}}}%
15669             \fi
15670             \settowidth{\dimen@}%
15671                 {\glsentrysymbol{\@glo@label}}}%
15672             \ifdim\dimen@>#2\relax
15673                 #2=\dimen@
15674             \fi

```

```

15675     }%
15676     {}%
15677     }%
15678     }%
15679 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

15680 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
15681   \dimen@=0pt\relax
15682   \gls@tmplen=0pt\relax
15683   #2=0pt\relax
15684   \forallglossaries[#1]{\@gls@type}%
15685   {%
15686     \forglentries[\@gls@type]{\@glo@label}%
15687     {%
15688       \settowidth{\dimen@}%
15689         {\glstreenamfmt{\glentryname{\@glo@label}}}%
15690       \ifdim\dimen@>\gls@tmplen
15691         \gls@tmplen=\dimen@
15692         \eglissetwidest{\glentryname{\@glo@label}}%
15693       \fi
15694       \settowidth{\dimen@}%
15695         {\glentrysymbol{\@glo@label}}%
15696       \ifdim\dimen@>#2\relax
15697         #2=\dimen@
15698       \fi
15699     }%
15700   }%
15701 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but also measures the location list. This requires `\glentrynumberlist`. 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.

```

15702 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
15703   \dimen@=0pt\relax
15704   \gls@tmplen=0pt\relax
15705   #2=0pt\relax
15706   #3=0pt\relax
15707   \forallglossaries[#1]{\@gls@type}%
15708   {%
15709     \forglentries[\@gls@type]{\@glo@label}%
15710     {%
15711       \ifglused{\@glo@label}%
15712       {%
15713         \settowidth{\dimen@}%
15714           {\glstreenamfmt{\glentryname{\@glo@label}}}%
15715         \ifdim\dimen@>\gls@tmplen
15716           \gls@tmplen=\dimen@

```

```

15717         \eglssetwidest{\glsentryname{\@glo@label}}}%
15718     \fi
15719     \settowidth{\dimen@}%
15720     {\glsentrysymbol{\@glo@label}}}%
15721     \ifdim\dimen@>#2\relax
15722         #2=\dimen@
15723     \fi
15724     \settowidth{\dimen@}%
15725     {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15726     \ifdim\dimen@>#3\relax
15727         #3=\dimen@
15728     \fi
15729 }%
15730 {}%
15731 }%
15732 }%
15733 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

15734 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
15735     \dimen@=0pt\relax
15736     \gls@tmplen=0pt\relax
15737     #2=0pt\relax
15738     #3=0pt\relax
15739     \forallglossaries[#1]{\@gls@type}%
15740     {%
15741         \forglsentries[\@gls@type]{\@glo@label}%
15742         {%
15743             \settowidth{\dimen@}%
15744             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15745             \ifdim\dimen@>\gls@tmplen
15746                 \gls@tmplen=\dimen@
15747                 \eglssetwidest{\glsentryname{\@glo@label}}}%
15748             \fi
15749             \settowidth{\dimen@}%
15750             {\glsentrysymbol{\@glo@label}}}%
15751             \ifdim\dimen@>#2\relax
15752                 #2=\dimen@
15753             \fi
15754             \settowidth{\dimen@}%
15755             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15756             \ifdim\dimen@>#3\relax
15757                 #3=\dimen@
15758             \fi
15759         }%
15760     }%
15761 }

```

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.

```

15762 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
15763   \dimen@=0pt\relax
15764   \gls@tmplen=0pt\relax
15765   #2=0pt\relax
15766   \forallglossaries[#1]{\@gls@type}%
15767   {%
15768     \forallglsentries[\@gls@type]{\@glo@label}%
15769     {%
15770       \ifglsused{\@glo@label}%
15771       {%
15772         \settowidth{\dimen@}%
15773           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15774         \ifdim\dimen@>\gls@tmplen
15775           \gls@tmplen=\dimen@
15776           \eglssetwidest{\glsentryname{\@glo@label}}%
15777         \fi
15778         \settowidth{\dimen@}%
15779           {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15780         \ifdim\dimen@>#2\relax
15781           #2=\dimen@
15782         \fi
15783       }%
15784     }%
15785   }%
15786 }%
15787 }
```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

15788 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
15789   \dimen@=0pt\relax
15790   \gls@tmplen=0pt\relax
15791   #2=0pt\relax
15792   \forallglossaries[#1]{\@gls@type}%
15793   {%
15794     \forallglsentries[\@gls@type]{\@glo@label}%
15795     {%
15796       \settowidth{\dimen@}%
15797         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15798       \ifdim\dimen@>\gls@tmplen
15799         \gls@tmplen=\dimen@
15800         \eglssetwidest{\glsentryname{\@glo@label}}%
15801       \fi
15802       \settowidth{\dimen@}%
15803         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15804       \ifdim\dimen@>#2\relax
15805         #2=\dimen@
15806       \fi

```



```

15807     }%
15808   }%
15809 }

```

`\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`.

```

15810 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
15811   \glstreeindent=\glstxtrtreetopindent\relax
15812 }

```

`\glstxtrComputeTreeSubIndent`

`\glstxtrComputeTreeSubIndent{<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.

```

15813 \newcommand*{\glstxtrComputeTreeSubIndent}[3]{%
15814   \ifcsundef{@glswidestname\romannumeral#1}%
15815   {%
15816     \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
15817   }%
15818   {%
15819     \settowidth{#3}{\glstreenamfmt{%
15820       \csname @glswidestname\romannumeral#1\endcsname\space}}%
15821   }%
15822 }

```

`\glstxtrAltTreeSetHangIndent` Set `\hangindent` for top-level entries:

```

15823 \newcommand*{\glstxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

`\glstxtrAltTreeSetSubHangIndent` Set `\hangindent` for sub-entries:

```

15824 \newcommand*{\glstxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `\alttree`:

```

15825 \renewglossarystyle{alttree}{%
15826   \renewenvironment{theglossary}%
15827   {%
15828     \glstxtralttreeInit
15829     \def\@gls@prevlevel{-1}%
15830     \mbox{}\par}%
15831   {\par}%
15832   \renewcommand*{\glossaryheader}{}%
15833   \renewcommand*{\glsgroupheading}[1]{}%
15834   \renewcommand{\glossentry}[2]{%
15835     \ifnum\@gls@prevlevel=0\relax

```

```

15836 \else
15837 \glxstrComputeTreeIndent{##1}%
15838 \fi
15839 \parindent\glstreeindent
15840 \glxstrAltTreeSetHangIndent
15841 \makebox[Opt][r]%
15842 {%
15843 \glstreenamebox{\glstreeindent}%
15844 {%
15845 \glssentryitem{##1}%
15846 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15847 }%
15848 }%
15849 \glxstralttreeSymbolDescLocation{##1}{##2}%
15850 \def\@gls@prevlevel{0}%
15851 }
15852 \renewcommand{\subglossentry}[3]{%
15853 \ifnum##1=1\relax
15854 \glssubentryitem{##2}%
15855 \fi
15856 \ifnum\@gls@prevlevel=##1\relax
15857 \else
15858 \glxstrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
15859 \ifnum\@gls@prevlevel<##1\relax
15860 \setlength\glstreeindent\gls@tmplen
15861 \addtolength\glstreeindent\parindent
15862 \parindent\glstreeindent
15863 \else
15864 \ifnum\@gls@prevlevel=0\relax
15865 \glxstrComputeTreeIndent{##2}%
15866 \else
15867 \glxstrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
15868 \fi
15869 \addtolength\parindent{-\glstreeindent}%
15870 \setlength\glstreeindent\parindent
15871 \fi
15872 \fi
15873 \glxstrAltTreeSetSubHangIndent{##1}%
15874 \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
15875 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
15876 \glxstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
15877 \def\@gls@prevlevel{##1}%
15878 }%
15879 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15880 }
15881 }%
15882 {%
15883 }

```

Redefine `alttreegroup` so that it discourages a break after group headings.

```

15884 \ifdef{\@glsstyle@almtreegroup}
15885 {%
15886   \renewglossarystyle{almtreegroup}{%
15887     \setglossarystyle{almtree}%
15888     \renewcommand{\glsgroupheading}[1]{\par
15889       \def\@gls@prevlevel{-1}%
15890       \hangindent0pt\relax
15891       \parindent0pt\relax
15892       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15893       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15894       \glstreegroupheaderfmt{\glsxtr@grptitle}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15895   \glstreegroupheaderskip
15896   }%
15897 }%
15898 }%
15899 {%
15900 }

```

Similarly for almtreehypergroup.

```

15901 \ifdef{\@glsstyle@almtreehypergroup}
15902 {%
15903   \renewglossarystyle{almtreehypergroup}{%
15904     \setglossarystyle{almtree}%
15905     \renewcommand*\{glossaryheader}{%
15906       \par
15907       \def\@gls@prevlevel{-1}%
15908       \hangindent0pt\relax
15909       \parindent0pt\relax
15910       \glstreenavigationfmt{\glsnavigation}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15911   \glstreegroupheaderskip
15912   }%
15913   \renewcommand*\{glsgroupheading}[1]{%
15914     \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15915     \glstreePreHeader{##1}{\glsxtr@grptitle}%
15916     \par
15917     \def\@gls@prevlevel{-1}%
15918     \hangindent0pt\relax
15919     \parindent0pt\relax
15920     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15921   \glstreegroupheaderskip
15922   }%
15923 }
15924 }%
15925 {%
15926 }

```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```
15927 \ifdef{\@glsstyle@mcolindexgroup}
15928 {%
15929   \renewglossarystyle{mcolindexgroup}{%
15930     \setglossarystyle{mcolindex}%
15931     \renewcommand*{\glsgroupheading}[1]{%
15932       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15933       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15934       \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
15935       \glstreegroupheaderskip\@afterheading
15936     }%
15937   }
15938 }%
15939 {%
15940 }
```

Similarly for mcolindexhypergroup.

```
15941 \ifdef{\@glsstyle@mcolindexhypergroup}
15942 {%
15943   \renewglossarystyle{mcolindexhypergroup}{%
15944     \setglossarystyle{mcolindex}%
15945     \renewcommand*{\glossaryheader}{%
15946       \item\glstreenavigationfmt{\glsnavigation}%
15947       \glstreegroupheaderskip\@afterheading
15948     }%
15949     \renewcommand*{\glsgroupheading}[1]{%
15950       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15951       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15952       \item\glstreegroupheaderfmt
15953         {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15954       \glstreegroupheaderskip\@afterheading
15955     }%
15956   }
15957 }%
15958 {%
15959 }
```

Similarly for mcolindexspannav.

```
15960 \ifdef{\@glsstyle@mcolindexspannav}
15961 {%
15962   \renewglossarystyle{mcolindexspannav}{%
15963     \setglossarystyle{index}%
15964     \renewenvironment{theglossary}%
15965     {%
15966       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]{%
15967         \setlength{\parindent}{0pt}%
15968         \setlength{\parskip}{0pt plus 0.3pt}%

```

```

15969     \let\item\glstreeitem}%
15970 {\end{multicols}}}%
15971 \renewcommand*\glsgroupheading}[1]{%
15972     \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15973     \glstreePreHeader{##1}{\glxtr@grptitle}%
15974     \item\glstreegroupheaderfmt
15975         {\glsnahypertarget{##1}{\glxtr@grptitle}}}%
15976     \glstreegroupheaderskip\@afterheading
15977 }%
15978 }
15979 }%
15980 {%
15981 }

```

Similarly for mcoltreegroup.

```

15982 \ifdef{\@glsstyle@mcoltreegroup}
15983 {%
15984     \renewglossarystyle{mcoltreegroup}{%
15985         \setglossarystyle{mcoltree}%
15986         \renewcommand*\glsgroupheading}[1]{%
15987             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15988             \glstreePreHeader{##1}{\glxtr@grptitle}%
15989             \par\noindent\glstreegroupheaderfmt{\glxtr@grptitle}%
15990             \glstreegroupheaderskip\@afterheading
15991         }%
15992     }
15993 }%
15994 {%
15995 }

```

Similarly for mcoltreehypergroup.

```

15996 \ifdef{\@glsstyle@mcoltreehypergroup}
15997 {%
15998     \renewglossarystyle{mcoltreehypergroup}{%
15999         \setglossarystyle{mcoltree}%
16000         \renewcommand*\glossaryheader{%
16001             \par\noindent\glstreenavigationfmt{\glsnavigation}%
16002             \glstreegroupheaderskip
16003         }%
16004         \renewcommand*\glsgroupheading}[1]{%
16005             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
16006             \glstreePreHeader{##1}{\glxtr@grptitle}%
16007             \par\noindent
16008             \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glxtr@grptitle}}}%
16009             \glstreegroupheaderskip\@afterheading
16010         }%
16011     }
16012 }%
16013 {%
16014 }

```

Similarly for mcoltreesspannav.

```

16015 \ifdef{\@glsstyle@mcoltreesspannav}
16016 {%
16017   \renewglossarystyle{mcoltreesspannav}{%
16018     \setglossarystyle{tree}%
16019     \renewenvironment{theglossary}%
16020     {%
16021       \begin{multicols}{\glsmcols}%
16022       [\noindent\glstreenavigationfmt{\glsnavigation}}%
16023       \setlength{\parindent}{0pt}%
16024       \setlength{\parskip}{0pt plus 0.3pt}%
16025     }%
16026     {\end{multicols}}}%
16027   \renewcommand*\{glsgroupheading}[1]{%
16028     \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16029     \glstreePreHeader{##1}{\glstr@grptitle}%
16030     \par\noindent
16031     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glstr@grptitle}}%
16032     \glstreegroupheaderskip\@afterheading
16033   }%
16034 }
16035 }%
16036 {%
16037 }

```

Similarly for mcoltreenonamegroup.

```

16038 \ifdef{\@glsstyle@mcoltreenonamegroup}
16039 {%
16040   \renewglossarystyle{mcoltreenonamegroup}{%
16041     \setglossarystyle{mcoltreenoname}%
16042     \renewcommand{\glsgroupheading}[1]{%
16043       \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16044       \glstreePreHeader{##1}{\glstr@grptitle}%
16045       \par\noindent\glstreegroupheaderfmt{\glstr@grptitle}%
16046       \glstreegroupheaderskip\@afterheading
16047     }%
16048   }
16049 }%
16050 {%
16051 }

```

Similarly for mcoltreenonamehypergroup.

```

16052 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
16053 {%
16054   \renewglossarystyle{mcoltreenonamehypergroup}{%
16055     \setglossarystyle{mcoltreenoname}%
16056     \renewcommand*\{glossaryheader}{%
16057       \par\noindent\glstreenavigationfmt{\glsnavigation}%
16058       \glstreegroupheaderskip
16059     }%

```

```

16060 \renewcommand*{\glsgroupheading}[1]{%
16061 \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
16062 \glstreePreHeader{##1}{\glxtr@grptitle}%
16063 \par\noindent
16064 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glxtr@grptitle}}%
16065 \glstreegroupheaderskip\@afterheading}%
16066 }
16067 }%
16068 {%
16069 }

```

Similarly for mcoltreenonamespannav.

```

16070 \ifdef{\@glsstyle@mcoltreenonamespannav}
16071 {%
16072 \renewglossarystyle{mcoltreenonamespannav}{%
16073 \setglossarystyle{treenoname}%
16074 \renewenvironment{theglossary}%
16075 {%
16076 \begin{multicols}{\glsmcols}%
16077 [\noindent\glstreenavigationfmt{\glsnavigation}}%
16078 \setlength{\parindent}{0pt}%
16079 \setlength{\parskip}{0pt plus 0.3pt}%
16080 }%
16081 {\end{multicols}}}%
16082 \renewcommand*{\glsgroupheading}[1]{%
16083 \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
16084 \glstreePreHeader{##1}{\glxtr@grptitle}%
16085 \par\noindent
16086 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glxtr@grptitle}}%
16087 \glstreegroupheaderskip\@afterheading}%
16088 }
16089 }%
16090 {%
16091 }

```

mcolaltnav needs adjusting so that it uses \glxtraltnavInit This doesn't use \mbox{\par which would unbalance the top of the columns.

```

16092 \ifdef{\@glsstyle@mcolaltnav}
16093 {%
16094 \renewglossarystyle{mcolaltnav}{%
16095 \setglossarystyle{altnav}%
16096 \renewenvironment{theglossary}%
16097 {%
16098 \glxtraltnavInit
16099 \def\@gls@prevlevel{-1}%
16100 \begin{multicols}{\glsmcols}%
16101 }%
16102 {\par\end{multicols}}}%
16103 }
16104 }%

```

```
16105 {%
16106 }
```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```
16107 \ifdef{\@glsstyle@mcolalmtreegroup}
16108 {%
16109   \renewglossarystyle{mcolalmtreegroup}{%
16110     \setglossarystyle{mcolalmtree}%
16111     \renewcommand{\glsgroupheading}[1]{%
16112       \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16113       \glstreePreHeader{##1}{\glstr@grptitle}%
16114       \par
16115       \def\@gls@prevlevel{-1}%
16116       \hangindent0pt\relax
16117       \parindent0pt\relax
16118       \glstreegroupheaderfmt{\glstr@grptitle}%
16119       \glstreegroupheaderskip
16120     }%
16121   }
16122 }%
16123 {%
16124 }
```

Similarly for mcolalmtreehypergroup.

```
16125 \ifdef{\@glsstyle@mcolalmtreehypergroup}
16126 {%
16127   \renewglossarystyle{mcolalmtreehypergroup}{%
16128     \setglossarystyle{mcolalmtree}%
16129     \renewcommand*\{glossaryheader}{%
16130       \par
16131       \def\@gls@prevlevel{-1}%
16132       \hangindent0pt\relax
16133       \parindent0pt\relax
16134       \glstreenavigationfmt{\glsnavigation}%
16135       \glstreegroupheaderskip
16136     }%
16137     \renewcommand*\{glsgroupheading}[1]{%
16138       \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16139       \glstreePreHeader{##1}{\glstr@grptitle}%
16140       \par
16141       \def\@gls@prevlevel{-1}%
16142       \hangindent0pt\relax
16143       \parindent0pt\relax
16144       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glstr@grptitle}}%
16145       \glstreegroupheaderskip
16146     }%
16147   }
16148 }%
16149 {%
16150 }
```


Similarly for mcolalttreespannav.

```
16151 \ifdef{\@glsstyle@mcolalttreespannav}
16152 {%
16153   \renewglossarystyle{mcolalttreespannav}{%
16154     \setglossarystyle{alttree}%
16155     \renewenvironment{theglossary}%
16156     {%
16157       \glstralttreeInit
16158       \def\@gls@prevlevel{-1}%
16159       \begin{multicols}{\glsmcols}%
16160         [\noindent\glstreenavigationfmt{\glsnavigation}}%
16161     }%
16162     {\par\end{multicols}}}%
16163   \renewcommand*\@gls@groupheading[1]{%
16164     \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16165     \glstreePreHeader{##1}{\glstr@grptitle}%
16166     \par
16167     \def\@gls@prevlevel{-1}%
16168     \hangindentOpt\relax
16169     \parindentOpt\relax
16170     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glstr@grptitle}}%
16171     \glstreegroupheaderskip
16172   }%
16173 }
16174}%
16175 {%
16176 }
```

Reset the default style

```
16177 \ifx\@glossary@default@style\relax
16178 \else
16179   \setglossarystyle{\@glstr@current@style}
16180 \fi
```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
16181 \NeedsTeXFormat{LaTeX2e}
16182 \ProvidesPackage{glossary-bookindex}[2020/03/23 v1.44 (NLCT)]
```

Load required packages.

```
16183 \RequirePackage{multicol}
16184 \RequirePackage{glossary-tree}
```

trbookindexcols Number of columns.

```
16185 \newcommand{\glstrbookindexcols}{2}
```

trbookindexname Format used for top-level entries. (Argument is the label.)

```
16186 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

ookindexsubname Format used for sub entries.

```
16187 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

xsxtrprelocation Provide in case glossaries-stylemods isn't loaded.

```
16188 \providecommand*{\glsxtrprelocation}{\space}
```

ndexprelocation 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.

```
16189 \newcommand*{\glsxtrbookindexprelocation}[1]{%
16190    \glxtrifhasfield{location}{#1}%
16191    {,\glsxtrprelocation}%
16192    {\glsxtrprelocation}%
16193 }
```

xsubprelocation Separator used before location list for sub-entries.

```
16194 \newcommand*{\glsxtrbookindexsubprelocation}[1]{%
16195    \glsxtrbookindexprelocation{#1}%
16196 }
```

okindexlocation

```
\glsxtrbookindexlocation{\langle label \rangle}{\langle location \rangle}
```

Displays the location.

```
16197 \newcommand*{\glsxtrbookindexlocation}[2]{#2}
```

ndexsublocation

`\glxtrbookindexlocation{<label>}{<location>}`

Displays the location for sub-entries.

```
16198 \newcommand*{\glxtrbookindexsublocation}{\glxtrbookindexlocation}
```

xparentchildsep Separator used between top-level parent and child entry.

```
16199 \newcommand{\glxtrbookindexparentchildsep}{\nopagebreak}
```

rentsubchildsep Separator used between sub-level parent and child entry.

```
16200 \newcommand{\glxtrbookindexparentsubchildsep}{\glxtrbookindexparentchildsep}
```

ookindexbetween Between two top-level entries identified by the labels in the arguments.

```
16201 \newcommand{\glxtrbookindexbetween}[2]{}%
```

indexsubbetween Between two level 1 entries identified by the labels in the arguments.

```
16202 \newcommand{\glxtrbookindexsubbetween}[2]{}%
```

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.

```
16203 \newcommand{\glxtrbookindexsubsubbetween}[2]{}%
```

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.

```
16204 \newcommand{\glxtrbookindexatendgroup}[1]{}%
```

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.

```
16205 \newcommand{\glxtrbookindexsubatendgroup}[1]{}%
```

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.

```
16206 \newcommand{\glxtrbookindexsubsubatendgroup}[1]{}%
```

kindexgroupskip Group separator.

```
16207 \newcommand{\glxtrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

dexformatheader Group separator.

```
16208 \newcommand*{\glxtrbookindexformatheader}[1]{%
16209 \par{\centering\glstreegroupheaderfmt{#1}\par}%
16210 }
```

okindexbookmark Book mark group heading if supported.

```
16211 \ifdef\pdfbookmark
16212 {%
16213 \newcommand*{\glxtrbookindexbookmark}[2]{%
16214 \ifdefstring{@@glossarysec}{chapter}%
16215 {\pdfbookmark[1]{#1}{#2}}%
16216 }
```

```

16216     {\pdfbookmark[2]{#1}{#2}}%
16217   }
16218 }
16219 {%
16220   \newcommand*{\glxtrbookindexbookmark}[2]{}
16221 }

```

bookmarkprefix Make the bookmark label prefix used for letter groups depend on the glossary label (instead of original hardcoded “index.”).

```

16222 \newcommand*{\glxtrbookindexbookmarkprefix}{\currentglossary.}

```

indexcolspread

```

16223 \newcommand*{\glxtrbookindexcolspread}{}

```

indexmulticolseenv

```

16224 \newcommand*{\glxtrbookindexmulticolseenv}{multicols}

```

Define the style.

```

16225 \newglossarystyle{bookindex}{%
16226   \setglossarystyle{index}%
16227   \renewenvironment{theglossary}%
16228   {%
16229     \ifnum\glxtrbookindexcols>1\relax
16230     \ifdefempty\glxtrbookindexcolspread
16231     {%
16232       \edef\glxtr@beginbookindex{%
16233         \noexpand\begin{\glxtrbookindexmulticolseenv}
16234         {\glxtrbookindexcols}%
16235       }%
16236     }%
16237     {%
16238       \edef\glxtr@beginbookindex{%
16239         \noexpand\begin{\glxtrbookindexmulticolseenv}%
16240         {\glxtrbookindexcols}{\glxtrbookindexcolspread}%
16241       }%
16242     }%
16243   \else
16244     \def\glxtr@beginbookindex{}%
16245   \fi
16246   \glxtr@beginbookindex
16247   \setlength{\parindent}{0pt}%
16248   \setlength{\parskip}{0pt plus 0.3pt}%
16249   \let\@glxtr@bookindex@sep\glxtrbookindexparentchildsep
16250   \let\@glxtr@bookindex@subsep\glxtrbookindexparentschildsep
16251   \let\@glxtr@bookindex@between\@gobble
16252   \let\@glxtr@bookindex@subbetween\@gobble
16253   \let\@glxtr@bookindex@subsubbetween\@gobble
16254   \let\@glxtr@bookindex@atendgroup\relax
16255   \let\@glxtr@bookindex@subatendgroup\relax

```

```

16256 \let\@glsxtr@bookindex@subsubatendgroup\relax
16257 \let\@glsxtr@bookindexgroupskip\relax
16258 }%
16259 {%

```

Do end group hooks.

```

16260 \@glsxtr@bookindex@subsubatendgroup
16261 \@glsxtr@bookindex@subatendgroup
16262 \@glsxtr@bookindex@atendgroup

```

End multicol environment.

```

16263 \ifnum\glsxtrbookindexcols>1\relax
16264 \edef\glsxtr@endbookindex{%
16265 \noexpand\end{\glsxtrbookindexmulticolscenv}%
16266 }%
16267 \else
16268 \def\glsxtr@endbookindex{%
16269 \fi
16270 \glsxtr@endbookindex
16271 }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

16272 \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

16273 \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

16274 \@glsxtr@bookindex@between{##1}%

```

Update separators.

```

16275 \let\@glsxtr@bookindex@sep\glsxtrbookindexparentchildsep
16276 \let\@glsxtr@bookindex@subsep\glsxtrbookindexparentsubchildsep
16277 \let\@glsxtr@bookindex@subbetween\@gobble
16278 \let\@glsxtr@bookindex@subsubbetween\@gobble
16279 \edef\@glsxtr@bookindex@between{%
16280 \noexpand\glsxtrbookindexbetween{##1}%
16281 }%
16282 \edef\@glsxtr@bookindex@atendgroup{%
16283 \noexpand\glsxtrbookindexatendgroup{##1}%
16284 }%
16285 \let\@glsxtr@bookindex@subatendgroup\relax
16286 \let\@glsxtr@bookindex@subsubatendgroup\relax

```

Format entry.

```

16287 \glstreeitem
16288 \glsentryitem{##1}%
16289 \glstarget{##1}{\glsxtrbookindexname{##1}}%
16290 \glsxtrbookindexprelocation{##1}%
16291 \glsxtrbookindexlocation{##1}{##2}%
16292 }%
16293 \renewcommand{\subglossentry}[3]{%
16294 \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```
16295     \glstreeitem
16296     \or
```

Level 1.

```
16297     \@glstr@bookindex@sep
16298     \@glstr@bookindex@subbetween{##2}%
16299     \let\@glstr@bookindex@sep\relax
```

Update separators.

```
16300     \let\@glstr@bookindex@subsubbetween\@gobble
16301     \let\@glstr@bookindex@subsep\glstrbookindexparentsubchildsep
16302     \edef\@glstr@bookindex@subbetween{%
16303         \noexpand\glstrbookindexsubbetween{##2}%
16304     }%
16305     \edef\@glstr@bookindex@atsubendgroup{%
16306         \noexpand\glstrbookindexatsubendgroup{##1}%
16307     }%
```

Start sub-item.

```
16308     \glstreesubitem
16309     \glssubentryitem{##2}%
16310     \else
```

All other levels.

```
16311     \@glstr@bookindex@subsep
16312     \@glstr@bookindex@subsubbetween{##2}%
```

Update separators.

```
16313     \let\@glstr@bookindex@subsep\relax
16314     \edef\@glstr@bookindex@subsubbetween{%
16315         \noexpand\glstrbookindexsubsubbetween{##2}%
16316     }%
16317     \edef\@glstr@bookindex@atsubsubendgroup{%
16318         \noexpand\glstrbookindexatsubsubendgroup{##1}%
16319     }%
```

Start sub-sub-item.

```
16320     \glstreesubsubitem
16321     \fi
```

Format entry.

```
16322     \glstarget{##2}{\glstrbookindexsubname{##2}}%
16323     \glstrbookindexsubprelocation{##2}%
16324     \glstrbookindexsublocation{##2}{##3}%
16325     }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
16326     \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
16327     \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
16328 \glxtr@bookindex@subsubatendgroup
16329 \glxtr@bookindex@subatendgroup
16330 \glxtr@bookindex@atendgroup
16331 \glxtr@bookindexgroupskip
```

Update separators.

```
16332 \let\glxtr@bookindexgroupskip\glxtrbookindexgroupskip
16333 \let\glxtr@bookindex@between\@gobble
16334 \let\glxtr@bookindex@atendgroup\relax
16335 \let\glxtr@bookindex@subatendgroup\relax
16336 \let\glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
16337 \glxtrgetgrouptitle{##1}{\glxtrcurrentgrptitle}%
```

Do the PDF bookmark if supported.

```
16338 \glxtrbookindexbookmark{\glxtrcurrentgrptitle}{\glxtrbookindexbookmarkprefix##1}%
```

Format the group title.

```
16339 \glxtrbookindexformatheader{\glxtrcurrentgrptitle}%
16340 \nopagebreak\indexspace\nopagebreak\@afterheading
16341 }%
16342 }
```

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.

`bookindexthepage` The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```
16343 \newcommand{\glxtrbookindexthepage}{%
16344 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
16345 }
```

`bookindexmarkentry` Writes entry information to the `.aux` file. The argument is the entry label.

```
16346 \newcommand*{\glxtrbookindexmarkentry}[1]{%
16347 \protected@write\@auxout
16348 {\let\glxtrbookindexthepage\relax}%
16349 {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
16350 }
```

`setbookindexmark`

```
16351 \newcommand*{\glxtr@setbookindexmark}[2]{%
16352 \ifcsundef\glxtr@idxfirstmark@#1}%
16353 {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
16354 {}%
16355 \csgdef\glxtr@idxlastmark@#1}{#2}%
16356 }
```

dexfirstmarkfmt

```
16357 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
16358   \glstryname{#1}%
16359 }
```

kindexfirstmark

```
16360 \newcommand*{\glxtrbookindexfirstmark}{%
16361   \letcs{\glxtr@label}{\glxtr@idxfirstmark@\glxtrbookindexthepage}%
16362   \ifdef\glxtr@label
16363     {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%
16364     {}%
16365 }
```

ndexlastmarkfmt

```
16366 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
16367   \glstryname{#1}%
16368 }
```

okindexlastmark

```
16369 \newcommand*{\glxtrbookindexlastmark}{%
16370   \letcs{\glxtr@label}{\glxtr@idxlastmark@\glxtrbookindexthepage}%
16371   \ifdef\glxtr@label
16372     {\glxtrbookindexlastmarkfmt{\glxtr@label}}%
16373     {}%
16374 }
```


4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
16375 \NeedsTeXFormat{LaTeX2e}  
16376 \ProvidesPackage{glossary-longextra}[2020/03/23 v1.44 (NLCT)]
```

Load required packages.

```
16377 \RequirePackage{glossary-longbooktabs}
```

ongextraNameFmt

```
\glslongextraNameFmt{<label>}
```

Governs the way the name is displayed.

```
16378 \newcommand{\glslongextraNameFmt}[1]{%  
16379 \glstentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%  
16380 }
```

ongextraDescFmt

```
\glslongextraDescFmt{<label>}
```

Governs the way the description is displayed.

```
16381 \newcommand{\glslongextraDescFmt}[1]{%  
16382 \glossentrydesc{#1}\glspostdescription  
16383 }
```

gextraSymbolFmt

```
\glslongextraSymbolFmt{<label>}
```

Governs the way the symbol is displayed.

```
16384 \newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

extraLocationFmt

```
\glslongextraLocationFmt{<label>}{<location list>}
```

Governs the way the location is displayed.

```
16385 \newcommand{\glslongextraLocationFmt}[2]{#2}
```

extraSubNameFmt

```
\glslongextraSubNameFmt{<level>}{<label>}
```

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
16386 \newcommand{\glslongextraSubNameFmt}[2]{%
16387 \glssubentryitem{#2}\glstarget{#2}{\strut}%
16388 }
```

extraSubDescFmt

```
\glslongextraSubDescFmt{<level>}{<label>}
```

Governs the way the child description is displayed.

```
16389 \newcommand{\glslongextraSubDescFmt}[2]{%
16390 \glslongextraDescFmt{#2}%
16391 }
```

extraSubSymbolFmt

```
\glslongextraSubSymbolFmt{<level>}{<label>}
```

Governs the way the child symbol is displayed.

```
16392 \newcommand{\glslongextraSubSymbolFmt}[2]{%
16393 \glslongextraSymbolFmt{#2}%
16394 }
```

extraSubLocationFmt

```
\glslongextraSubLocationFmt{<level>}{<label>}{<location list>}
```

Governs the way the child location list is displayed.

```
16395 \newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

`gextraNameAlign` Alignment for the name column.

```
16396 \newcommand{\glslongextraNameAlign}{l}
```

`gextraDescAlign` Alignment for the description column.

```
16397 \newcommand{\glslongextraDescAlign}{>{\raggedright}p{\glstdescwidth}}
```

`extraSymbolAlign` Alignment for the symbol column.

```
16398 \newcommand{\glslongextraSymbolAlign}{c}
```

`raLocationAlign` Alignment for the location column.

```
16399 \newcommand{\glslongextraLocationAlign}{>{\raggedright}p{\glspagelistwidth}}
```

`extraGroupHeading` 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).

```
16400 \newcommand{\glslongextraGroupHeading}[2]{}
```

`extraHeaderFormat` Format for the column headers.

```
16401 \newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}
```

`aNameDescHeader`

```
16402 \newcommand{\glslongextraNameDescHeader}{%
16403 \glslongextraNameDescTabularHeader\endhead
16404 \glslongextraNameDescTabularFooter\endfoot
16405 }
```

`scTabularHeader`

```
16406 \newcommand{\glslongextraNameDescTabularHeader}{%
16407 \toprule
16408 \glslongextraHeaderFmt\entryname &
16409 \glslongextraHeaderFmt\descriptionname\tabularnewline
16410 \midrule
16411 }
```

`scTabularFooter`

```
16412 \newcommand{\glslongextraNameDescTabularFooter}{%
16413 \bottomrule
16414 }
```

Unlike the `alttree` style, there aren't different widths for the hierarchical levels.

`gextraSetWidest` Provide in case the tree styles haven't been loaded.

```
16415 \newcommand*{\glslongextraSetWidest}[1]{%
16416 \def\@glslongextrawidestname{#1}%
16417 }
```

`extrawidestname` Pick up the widest name from the `alttree` style if it has been set. (Will expand to nothing otherwise.)

```
16418 \newcommand*{\@glslongextrawidestname}{\csuse{@glswidestname}}
```

traUpdateWidest

```

16419 \newcommand*{\glslongextraUpdateWidest}[1]{%
16420   \ifundef\@glslongextrawidestname
16421   {\def\@glslongextrawidestname{#1}}%
16422   {%
16423     \settowidth{\dimen@}{\@glslongextrawidestname}%
16424     \settowidth{\dimen@ii}{#1}%
16425     \ifdim\dimen@ii>\dimen@
16426     \def\@glslongextrawidestname{#1}%
16427     \fi
16428   }%
16429 }
```

dateWidestChild

`\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`.

```

16430 \newcommand*{\glslongextraUpdateWidestChild}[2]{}
```

traSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```

16431 \newcommand{\glslongextraSetDescWidth}{%
16432   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%
16433   \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
16434   \ifdim\dimen@>\gls@tmplen
16435     \gls@tmplen=\dimen@
16436   \fi
```

Has the widest name been set.

Description width is `\linewidth` less `4\tabcolsep` less the width of the name column.

```

16437   \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
16438 }
```

SymSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```

16439 \newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```

16440   \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```

16441   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```

Subtract 2\tabcolsep and the symbol header width.

```

16442 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
16443 }

```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that only have name, location and description columns.

```

16444 \newcommand{\glslongextraLocSetDescWidth}{%
  Work out the size for just the name and description style.
16445 \glslongextraSetDescWidth
  Subtract 2\tabcolsep and the location list column width.
16446 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
16447 }

```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that have name, symbol, location and description columns.

```

16448 \newcommand{\glslongextraSymLocSetDescWidth}{%
  Work out the size for just the name, symbol and description style.
16449 \glslongextraSymSetDescWidth
  Subtract 2\tabcolsep and the location list column width.
16450 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
16451 }

```

ExtraUseTabular If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```

16452 \newif\ifGlsLongExtraUseTabular
16453 \GlsLongExtraUseTabularfalse

```

raTabularVAlign Only used with the tabular setting.

```

16454 \newcommand*{\glslongextraTabularVAlign}{c}

```

long-name-desc Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```

16455 \newglossarystyle{long-name-desc}%
16456 {%
16457 \ifGlsLongExtraUseTabular
16458 \renewenvironment{theglossary}%
16459 {%
16460 \glslongextraSetDescWidth
16461 \edef\@glslongextra@begintab{%
16462 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16463 \expandonce\glslongextraNameAlign
16464 \expandonce\glslongextraDescAlign}}%
16465 \@glslongextra@begintab
16466 }%
16467 {%

```

```

16468     \glslongextraNameDescTabularFooter
16469     \end{tabular}%
16470 }%
16471 \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
16472 \else
16473 \renewenvironment{theglossary}%
16474 {%
16475     \glspatchLToutput
16476     \glslongextraSetDescWidth
16477     \edef\@glslongextra@begintab{%
16478         \noexpand\begin{longtable}{%
16479             \expandonce\glslongextraNameAlign
16480             \expandonce\glslongextraDescAlign}}%
16481     \@glslongextra@begintab
16482 }%
16483 {\end{longtable}}%
16484 \renewcommand*{\glossaryheader}{\glslongextraNameDescHeader}%
16485 \fi
16486 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
16487 \renewcommand{\glossentry}[2]{%
16488     \glslongextraNameFmt{##1} &
16489     \glslongextraDescFmt{##1}\tabularnewline
16490 }%
16491 \renewcommand{\subglossentry}[3]{%
16492     \glslongextraSubNameFmt{##1}{##2}
16493     &
16494     \glslongextraSubDescFmt{##1}{##2}%
16495     \tabularnewline
16496 }%
16497 \ifglsgroupskip
16498     \renewcommand*{\glsgroupskip}{}%
16499 \else
16500     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16501 \fi
16502 }

```

cLocationHeader

```

16503 \newcommand{\glslongextraNameDescLocationHeader}{%
16504     \glslongextraNameDescLocationTabularHeader\endhead
16505     \glslongextraNameDescLocationTabularFooter\endfoot
16506 }

```

onTabularHeader

```

16507 \newcommand{\glslongextraNameDescLocationTabularHeader}{%
16508     \toprule
16509     \glslongextraHeaderFmt\entryname &
16510     \glslongextraHeaderFmt\descriptionname &
16511     \glslongextraHeaderFmt\pagelistname\tabularnewline
16512     \midrule

```

16513 }

onTabularFooter

```
16514 \newcommand{\glslongextraNameDescLocationTabularFooter}{%
16515   \bottomrule
16516 }
```

g-name-desc-loc Three columns: name, description and location list.

```
16517 \newglossarystyle{long-name-desc-loc}%
16518 {%
16519   \ifGlsLongExtraUseTabular
16520     \renewenvironment{theglossary}%
16521       {%
16522         \glslongextraLocSetDescWidth
16523         \edef\@glslongextra@begintab{%
16524           \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16525             \expandonce\glslongextraNameAlign
16526             \expandonce\glslongextraDescAlign
16527             \expandonce\glslongextraLocationAlign
16528           }}%
16529         \@glslongextra@begintab
16530       }%
16531       {%
16532         \glslongextraNameDescLocationTabularFooter
16533         \end{tabular}%
16534       }%
16535     \renewcommand*\glossaryheader{\glslongextraNameDescLocationTabularHeader}%
16536   \else
16537     \renewenvironment{theglossary}%
16538       {%
16539         \glspatchLToutput
16540         \glslongextraLocSetDescWidth
16541         \edef\@glslongextra@begintab{%
16542           \noexpand\begin{longtable}%
16543             \expandonce\glslongextraNameAlign
16544             \expandonce\glslongextraDescAlign
16545             \expandonce\glslongextraLocationAlign
16546           }}%
16547         \@glslongextra@begintab
16548       }%
16549     {\end{longtable}}%
16550   \renewcommand*\glossaryheader{\glslongextraNameDescLocationHeader}%
16551   \fi
16552   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16553   \renewcommand{\glossentry}[2]{%
16554     \glslongextraNameFmt{##1} &
16555     \glslongextraDescFmt{##1} &
16556     \glslongextraLocationFmt{##1}{##2}\tabularnewline
16557   }%
```

```

16558 \renewcommand{\subglossentry}[3]{%
16559     \glslongextraSubNameFmt{##1}{##2}&
16560     \glslongextraSubDescFmt{##1}{##2}&
16561     \glslongextraSubLocationFmt{##1}{##2}{##3}%
16562     \tabularnewline
16563 }%
16564 \ifglsgnogroupskip
16565     \renewcommand*{\glsgroupskip}{}%
16566 \else
16567     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16568 \fi
16569 }

```

aDescNameHeader

```

16570 \newcommand{\glslongextraDescNameHeader}{%
16571     \glslongextraDescNameTabularHeader\endhead
16572     \glslongextraDescNameTabularFooter\endfoot
16573 }

```

meTabularHeader

```

16574 \newcommand{\glslongextraDescNameTabularHeader}{%
16575     \toprule
16576     \glslongextraHeaderFmt\descriptionname&
16577     \glslongextraHeaderFmt\entryname \tabularnewline
16578     \midrule
16579 }

```

meTabularFooter

```

16580 \newcommand{\glslongextraDescNameTabularFooter}{%
16581     \bottomrule
16582 }

```

long-desc-name Like name-desc but swaps the columns.

```

16583 \newglossarystyle{long-desc-name}%
16584 {%
16585     \ifGlsLongExtraUseTabular
16586     \renewenvironment{theglossary}%
16587     {%
16588         \glslongextraSetDescWidth
16589         \edef\@glslongextra@begintab{%
16590             \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16591                 \expandonce\glslongextraDescAlign
16592                 \expandonce\glslongextraNameAlign}}%
16593         \@glslongextra@begintab
16594     }%
16595     {%
16596         \glslongextraDescNameTabularFooter
16597         \end{tabular}%
16598     }%

```



```

16599 \renewcommand*{\glossaryheader}{\glslongextraDescNameTabularHeader}%
16600 \else
16601 \renewenvironment{theglossary}%
16602 {%
16603 \glspatchLToutput
16604 \glslongextraSetDescWidth
16605 \edef\@glslongextra@begintab{%
16606 \noexpand\begin{longtable}{%
16607 \expandonce\glslongextraDescAlign
16608 \expandonce\glslongextraNameAlign}}%
16609 \@glslongextra@begintab
16610 }%
16611 {\end{longtable}}%
16612 \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
16613 \fi
16614 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
16615 \renewcommand{\glossentry}[2]{%
16616 \glslongextraDescFmt{##1} &
16617 \glslongextraNameFmt{##1}\tabularnewline
16618 }%
16619 \renewcommand{\subglossentry}[3]{%
16620 \glslongextraSubDescFmt{##1}{##2} &
16621 \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16622 }%
16623 \ifglsgroupskip
16624 \renewcommand*{\glsgroupskip}{}%
16625 \else
16626 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16627 \fi
16628 }

```

nDescNameHeader

```

16629 \newcommand{\glslongextraLocationDescNameHeader}{%
16630 \glslongextraLocationDescNameTabularHeader\endhead
16631 \glslongextraLocationDescNameTabularFooter\endfoot
16632 }

```

meTabularHeader

```

16633 \newcommand{\glslongextraLocationDescNameTabularHeader}{%
16634 \toprule
16635 \glslongextraHeaderFmt\pagelistname&
16636 \glslongextraHeaderFmt\descriptionname&
16637 \glslongextraHeaderFmt\entryname \tabularnewline
16638 \midrule
16639 }

```

meTabularFooter

```

16640 \newcommand{\glslongextraLocationDescNameTabularFooter}{%
16641 \bottomrule

```

16642 }

g-loc-desc-name Three columns: location, description and name.

```
16643 \newglossarystyle{long-loc-desc-name}%
16644 {%
16645   \ifGlsLongExtraUseTabular
16646     {%
16647       \glslongextraLocSetDescWidth
16648       \edef\@glslongextra@begintab{%
16649         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16650           \expandonce\glslongextraLocationAlign
16651           \expandonce\glslongextraDescAlign
16652           \expandonce\glslongextraNameAlign}}%
16653       \@glslongextra@begintab
16654     }%
16655     {%
16656       \glslongextraLocationDescNameTabularFooter
16657       \end{tabular}%
16658     }%
16659     \renewcommand*\glossaryheader{\glslongextraLocationDescNameTabularHeader}%
16660   \else
16661     \renewenvironment{theglossary}%
16662     {%
16663       \glspatchLToutput
16664       \glslongextraLocSetDescWidth
16665       \edef\@glslongextra@begintab{%
16666         \noexpand\begin{longtable}{%
16667           \expandonce\glslongextraLocationAlign
16668           \expandonce\glslongextraDescAlign
16669           \expandonce\glslongextraNameAlign}}%
16670       \@glslongextra@begintab
16671     }%
16672     {\end{longtable}}%
16673     \renewcommand*\glossaryheader{\glslongextraLocationDescNameHeader}%
16674   \fi
16675   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16676   \renewcommand{\glossentry}[2]{%
16677     \glslongextraLocationFmt{##1}{##2} &
16678     \glslongextraDescFmt{##1} &
16679     \glslongextraNameFmt{##1}\tabularnewline
16680   }%
16681   \renewcommand{\subglossentry}[3]{%
16682     \glslongextraSubLocationFmt{##1}{##2}{##3} &
16683     \glslongextraSubDescFmt{##1}{##2} &
16684     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16685   }%
16686   \ifglsgroupskip
16687     \renewcommand*\glsgroupskip{}%
16688   \else
```

```

16689 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16690 \fi
16691 }

```

meDescSymHeader

```

16692 \newcommand{\glslongextraNameDescSymHeader}{%
16693 \glslongextraNameDescSymTabularHeader\endhead
16694 \glslongextraNameDescSymTabularFooter\endfoot
16695 }

```

ymTabularHeader

```

16696 \newcommand{\glslongextraNameDescSymTabularHeader}{%
16697 \toprule
16698 \glslongextraHeaderFmt\entryname &
16699 \glslongextraHeaderFmt\descriptionname &
16700 \glslongextraHeaderFmt\symbolname\tabularnewline
16701 \midrule
16702 }

```

ymTabularFooter

```

16703 \newcommand{\glslongextraNameDescSymTabularFooter}{%
16704 \bottomrule
16705 }

```

g-name-desc-sym Three column style with symbol in the third column.

```

16706 \newglossarystyle{long-name-desc-sym}%
16707 {%
16708 \ifGlsLongExtraUseTabular
16709 \renewenvironment{theglossary}%
16710 {%
16711 \glslongextraSymSetDescWidth
16712 \edef\@glslongextra@begintab{%
16713 \noexpand\begin{tabular}{\glslongextraTabularVAlign}{%
16714 \expandonce\glslongextraNameAlign
16715 \expandonce\glslongextraDescAlign
16716 \expandonce\glslongextraSymbolAlign
16717 }}%
16718 \@glslongextra@begintab
16719 }%
16720 {%
16721 \glslongextraNameDescSymTabularFooter
16722 \end{tabular}%
16723 }%
16724 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymTabularHeader}%
16725 \else
16726 \renewenvironment{theglossary}%
16727 {%
16728 \glspatchLToutput
16729 \glslongextraSymSetDescWidth

```

```

16730 \edef\@glslongextra@begintab{%
16731 \noexpand\begin{longtable}{%
16732 \expandonce\glslongextraNameAlign
16733 \expandonce\glslongextraDescAlign
16734 \expandonce\glslongextraSymbolAlign
16735 }}%
16736 \@glslongextra@begintab
16737 }%
16738 {\end{longtable}}}%
16739 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymHeader}%
16740 \fi
16741 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16742 \renewcommand{\glossentry}[2]{%
16743 \glslongextraNameFmt{##1} &
16744 \glslongextraDescFmt{##1} &
16745 \glslongextraSymbolFmt{##1}\tabularnewline
16746 }%
16747 \renewcommand{\subglossentry}[3]{%
16748 \glslongextraSubNameFmt{##1}{##2} &
16749 \glslongextraSubDescFmt{##1}{##2} &
16750 \glslongextraSubSymbolFmt{##1}{##2}%
16751 \tabularnewline
16752 }%
16753 \ifglsnogroupskip
16754 \renewcommand*{\glsgroupskip}{}%
16755 \else
16756 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16757 \fi
16758 }

```

mLocationHeader

```

16759 \newcommand{\glslongextraNameDescSymLocationHeader}{%
16760 \glslongextraNameDescSymLocationTabularHeader\endhead
16761 \glslongextraNameDescSymLocationTabularFooter\endfoot
16762 }

```

onTabularHeader

```

16763 \newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
16764 \toprule
16765 \glslongextraHeaderFmt\entryname &
16766 \glslongextraHeaderFmt\descriptionname &
16767 \glslongextraHeaderFmt\symbolname &
16768 \glslongextraHeaderFmt\pagelistname\tabularnewline
16769 \midrule
16770 }

```

onTabularFooter

```

16771 \newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
16772 \bottomrule

```

16773 }

me-desc-sym-loc Four columns: name, description and location

```
16774 \newglossarystyle{long-name-desc-sym-loc}%
16775 {%
16776   \ifGlsLongExtraUseTabular
16777     \renewenvironment{theglossary}%
16778       {%
16779         \glslongextraSymLocSetDescWidth
16780         \edef\@glslongextra@begintab{%
16781           \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16782             \expandonce\glslongextraNameAlign
16783             \expandonce\glslongextraDescAlign
16784             \expandonce\glslongextraSymbolAlign
16785             \expandonce\glslongextraLocationAlign
16786           }}%
16787         \@glslongextra@begintab
16788       }%
16789       {%
16790         \glslongextraNameDescSymLocationTabularFooter
16791         \end{tabular}%
16792       }%
16793   \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationTabularHeader}%
16794 \else
16795   \renewenvironment{theglossary}%
16796     {%
16797       \glspatchLToutput
16798       \glslongextraSymLocSetDescWidth
16799       \edef\@glslongextra@begintab{%
16800         \noexpand\begin{longtable}%
16801           \expandonce\glslongextraNameAlign
16802           \expandonce\glslongextraDescAlign
16803           \expandonce\glslongextraSymbolAlign
16804           \expandonce\glslongextraLocationAlign
16805         }}%
16806       \@glslongextra@begintab
16807     }%
16808     {\end{longtable}}%
16809   \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationHeader}%
16810 \fi
16811 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16812 \renewcommand{\glossentry}[2]{%
16813   \glslongextraNameFmt{##1} &
16814   \glslongextraDescFmt{##1} &
16815   \glslongextraSymbolFmt{##1}&
16816   \glslongextraLocationFmt{##1}{##2}\tabularnewline
16817 }%
16818 \renewcommand{\subglossentry}[3]{%
16819   \glslongextraSubNameFmt{##1}{##2} &
```

```

16820 \glslongextraSubDescFmt{##1}{##2} &
16821 \glslongextraSubSymbolFmt{##1}{##2}&
16822 \glslongextraSubLocationFmt{##1}{##2}{##3}%
16823 \tabularnewline
16824 }%
16825 \ifglsgnogroupskip
16826 \renewcommand*{\glsgroupskip}{}%
16827 \else
16828 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16829 \fi
16830 }

```

meSymDescHeader

```

16831 \newcommand{\glslongextraNameSymDescHeader}{%
16832 \glslongextraNameSymDescTabularHeader\endhead
16833 \glslongextraNameSymDescTabularFooter\endfoot
16834 }

```

scTabularHeader

```

16835 \newcommand{\glslongextraNameSymDescTabularHeader}{%
16836 \toprule
16837 \glslongextraHeaderFmt\entryname &
16838 \glslongextraHeaderFmt\symbolname &
16839 \glslongextraHeaderFmt\descriptionname\tabularnewline
16840 \midrule
16841 }

```

scTabularFooter

```

16842 \newcommand{\glslongextraNameSymDescTabularFooter}{%
16843 \bottomrule
16844 }

```

g-name-sym-desc Three column style with symbol in the second column.

```

16845 \newglossarystyle{long-name-sym-desc}%
16846 {%
16847 \ifGlsLongExtraUseTabular
16848 \renewenvironment{theglossary}%
16849 {%
16850 \glslongextraSymSetDescWidth
16851 \edef\@glslongextra@begintab{%
16852 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16853 \expandonce\glslongextraNameAlign
16854 \expandonce\glslongextraSymbolAlign
16855 \expandonce\glslongextraDescAlign
16856 }}%
16857 \@glslongextra@begintab
16858 }%
16859 {%
16860 \glslongextraNameSymDescTabularFooter

```

```

16861 \end{tabular}%
16862 }%
16863 \renewcommand*{\glossaryheader}{\glslongextraNameSymDescTabularHeader}%
16864 \else
16865 \renewenvironment{theglossary}%
16866 {%
16867 \glspatchLToutput
16868 \glslongextraSymSetDescWidth
16869 \edef\@glslongextra@begintab{%
16870 \noexpand\begin{longtable}{%
16871 \expandonce\glslongextraNameAlign
16872 \expandonce\glslongextraSymbolAlign
16873 \expandonce\glslongextraDescAlign
16874 }}%
16875 \@glslongextra@begintab
16876 }%
16877 {\end{longtable}}%
16878 \renewcommand*{\glossaryheader}{\glslongextraNameSymDescHeader}%
16879 \fi
16880 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16881 \renewcommand{\glossentry}[2]{%
16882 \glslongextraNameFmt{##1} &
16883 \glslongextraSymbolFmt{##1} &
16884 \glslongextraDescFmt{##1}\tabularnewline
16885 }%
16886 \renewcommand{\subglossentry}[3]{%
16887 \glslongextraSubNameFmt{##1}{##2} &
16888 \glslongextraSubSymbolFmt{##1}{##2} &
16889 \glslongextraSubDescFmt{##1}{##2}\tabularnewline
16890 }%
16891 \ifglsgroupskip
16892 \renewcommand*{\glsgroupskip}{}%
16893 \else
16894 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16895 \fi
16896 }

```

cLocationHeader

```

16897 \newcommand{\glslongextraNameSymDescLocationHeader}{%
16898 \glslongextraNameSymDescLocationTabularHeader\endhead
16899 \glslongextraNameSymDescLocationTabularFooter\endfoot
16900 }

```

onTabularHeader

```

16901 \newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
16902 \toprule
16903 \glslongextraHeaderFmt\entryname &
16904 \glslongextraHeaderFmt\symbolname &
16905 \glslongextraHeaderFmt\descriptionname &

```

```

16906 \glslongextraHeaderFmt\pagelistname\tabularnewline
16907 \midrule
16908 }

```

onTabularFooter

```

16909 \newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
16910 \bottomrule
16911 }

```

me-sym-desc-loc Four column style with symbol in the second column.

```

16912 \newglossarystyle{long-name-sym-desc-loc}%
16913 {%
16914 \ifGlsLongExtraUseTabular
16915 \renewenvironment{theglossary}%
16916 {%
16917 \glslongextraSymLocSetDescWidth
16918 \edef\@glslongextra@begintab{%
16919 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16920 \expandonce\glslongextraNameAlign
16921 \expandonce\glslongextraSymbolAlign
16922 \expandonce\glslongextraDescAlign
16923 \expandonce\glslongextraLocationAlign
16924 }}%
16925 \@glslongextra@begintab
16926 }%
16927 {%
16928 \glslongextraNameSymDescLocationTabularFooter
16929 \end{tabular}}%
16930 }%
16931 \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationTabularHeader}%
16932 \else
16933 \renewenvironment{theglossary}%
16934 {%
16935 \glspatchLToutput
16936 \glslongextraSymLocSetDescWidth
16937 \edef\@glslongextra@begintab{%
16938 \noexpand\begin{longtable}{%
16939 \expandonce\glslongextraNameAlign
16940 \expandonce\glslongextraSymbolAlign
16941 \expandonce\glslongextraDescAlign
16942 \expandonce\glslongextraLocationAlign
16943 }}%
16944 \@glslongextra@begintab
16945 }%
16946 {\end{longtable}}%
16947 \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationHeader}%
16948 \fi
16949 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16950 \renewcommand{\glossentry}[2]{%

```



```

16951 \glslongextraNameFmt{##1} &
16952 \glslongextraSymbolFmt{##1} &
16953 \glslongextraDescFmt{##1} &
16954 \glslongextraLocationFmt{##1}{##2}\tabularnewline
16955 }%
16956 \renewcommand{\subglossentry}[3]{%
16957 \glslongextraSubNameFmt{##1}{##2} &
16958 \glslongextraSubSymbolFmt{##1}{##2} &
16959 \glslongextraSubDescFmt{##1}{##2} &
16960 \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
16961 }%
16962 \ifglsgnognogroupskip
16963 \renewcommand*\{\glsgroupskip}{}%
16964 \else
16965 \renewcommand*\{\glsgroupskip}{\glspenaltygroupskip}%
16966 \fi
16967 }

```

mDescNameHeader

```

16968 \newcommand{\glslongextraSymDescNameHeader}{%
16969 \glslongextraSymDescNameTabularHeader\endhead
16970 \glslongextraSymDescNameTabularFooter\endfoot
16971 }

```

meTabularHeader

```

16972 \newcommand{\glslongextraSymDescNameTabularHeader}{%
16973 \toprule
16974 \glslongextraHeaderFmt\symbolname &
16975 \glslongextraHeaderFmt\descriptionname &
16976 \glslongextraHeaderFmt\entryname\tabularnewline
16977 \midrule
16978 }

```

meTabularFooter

```

16979 \newcommand{\glslongextraSymDescNameTabularFooter}{%
16980 \bottomrule
16981 }

```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```

16982 \newglossarystyle{long-sym-desc-name}%
16983 {%
16984 \ifGlsLongExtraUseTabular
16985 \renewenvironment{theglossary}%
16986 {%
16987 \glslongextraSymSetDescWidth
16988 \edef\@glslongextra@begintab{%
16989 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16990 \expandonce\glslongextraSymbolAlign

```

```

16991         \expandonce\glslongextraDescAlign
16992         \expandonce\glslongextraNameAlign
16993     } }%
16994     \@glslongextra@begintab
16995 }%
16996 {%
16997     \glslongextraSymDescNameTabularFooter
16998     \end{tabular}%
16999 }%
17000 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
17001 \else
17002 \renewenvironment{theglossary}%
17003 {%
17004     \glspatchLToutput
17005     \glslongextraSymSetDescWidth
17006     \edef\@glslongextra@begintab{%
17007         \noexpand\begin{longtable}{%
17008             \expandonce\glslongextraSymbolAlign
17009             \expandonce\glslongextraDescAlign
17010             \expandonce\glslongextraNameAlign
17011         } }%
17012     \@glslongextra@begintab
17013 }%
17014     {\end{longtable}}%
17015 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameHeader}%
17016 \fi
17017 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
17018 \renewcommand{\glossentry}[2]{%
17019     \glslongextraSymbolFmt{##1} &
17020     \glslongextraDescFmt{##1} &
17021     \glslongextraNameFmt{##1}\tabularnewline
17022 }%
17023 \renewcommand{\subglossentry}[3]{%
17024     \glslongextraSubSymbolFmt{##1}{##2} &
17025     \glslongextraSubDescFmt{##1}{##2} &
17026     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17027 }%
17028 \ifglsgroupskip
17029     \renewcommand*{\glsgroupskip}{}%
17030 \else
17031     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
17032 \fi
17033 }

```

mDescNameHeader

```

17034 \newcommand{\glslongextraLocationSymDescNameHeader}{%
17035     \glslongextraLocationSymDescNameTabularHeader\endhead
17036     \glslongextraLocationSymDescNameTabularFooter\endfoot
17037 }

```

meTabularHeader

```

17038 \newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
17039   \toprule
17040   \glslongextraHeaderFmt\pagelistname &
17041   \glslongextraHeaderFmt\symbolname &
17042   \glslongextraHeaderFmt\descriptionname &
17043   \glslongextraHeaderFmt\entryname\tabularnewline
17044   \midrule
17045 }

```

meTabularFooter

```

17046 \newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
17047   \bottomrule
17048 }

```

c-sym-desc-name Four column style with location list, symbol, description and name.

```

17049 \newglossarystyle{long-loc-sym-desc-name}%
17050 {%
17051   \ifGlsLongExtraUseTabular
17052     \renewenvironment{theglossary}%
17053     {%
17054       \glslongextraSymLocSetDescWidth
17055       \edef\@glslongextra@begintab{%
17056         \noexpand\begin{tabular}\@glslongextraTabularVAlign}{%
17057         \expandonce\glslongextraLocationAlign
17058         \expandonce\glslongextraSymbolAlign
17059         \expandonce\glslongextraDescAlign
17060         \expandonce\glslongextraNameAlign
17061       }}%
17062       \@glslongextra@begintab
17063     }%
17064     {%
17065       \glslongextraLocationSymDescNameTabularFooter
17066       \end{tabular}%
17067     }%
17068     \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameTabularHeader}%
17069   \else
17070     \renewenvironment{theglossary}%
17071     {%
17072       \glspatchLToutput
17073       \glslongextraSymLocSetDescWidth
17074       \edef\@glslongextra@begintab{%
17075         \noexpand\begin{longtable}{%
17076         \expandonce\glslongextraLocationAlign
17077         \expandonce\glslongextraSymbolAlign
17078         \expandonce\glslongextraDescAlign
17079         \expandonce\glslongextraNameAlign
17080       }}%
17081       \@glslongextra@begintab

```

```

17082 }%
17083 {\end{longtable}}}%
17084 \renewcommand*{\glossaryheader}{\glslongextraLocationSymDescNameHeader}%
17085 \fi
17086 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
17087 \renewcommand{\glossentry}[2]{%
17088   \glslongextraLocationFmt{##1}{##2} &
17089   \glslongextraSymbolFmt{##1} &
17090   \glslongextraDescFmt{##1} &
17091   \glslongextraNameFmt{##1}\tabularnewline
17092 }%
17093 \renewcommand{\subglossentry}[3]{%
17094   \glslongextraSubLocationFmt{##1}{##2}{##3} &
17095   \glslongextraSubSymbolFmt{##1}{##2} &
17096   \glslongextraSubDescFmt{##1}{##2} &
17097   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17098 }%
17099 \ifglsgnogroupskip
17100   \renewcommand*{\glsgroupskip}{}%
17101 \else
17102   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
17103 \fi
17104 }

```

scSymNameHeader

```

17105 \newcommand{\glslongextraDescSymNameHeader}{%
17106   \glslongextraDescSymNameTabularHeader\endhead
17107   \glslongextraDescSymNameTabularFooter\endfoot
17108 }

```

meTabularHeader

```

17109 \newcommand{\glslongextraDescSymNameTabularHeader}{%
17110   \toprule
17111   \glslongextraHeaderFmt\descriptionname &
17112   \glslongextraHeaderFmt\symbolname &
17113   \glslongextraHeaderFmt\entryname\tabularnewline
17114   \midrule
17115 }

```

meTabularFooter

```

17116 \newcommand{\glslongextraDescSymNameTabularFooter}{%
17117   \bottomrule
17118 }

```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```

17119 \newglossarystyle{long-desc-sym-name}%
17120 {%
17121   \ifGlsLongExtraUseTabular

```

```

17122 \renewenvironment{theglossary}%
17123 {%
17124     \glslongextraSymSetDescWidth
17125     \edef\@glslongextra@begintab{%
17126         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
17127             \expandonce\glslongextraDescAlign
17128             \expandonce\glslongextraSymbolAlign
17129             \expandonce\glslongextraNameAlign
17130         }}%
17131     \@glslongextra@begintab
17132 }%
17133 {%
17134     \glslongextraDescSymNameTabularFooter
17135     \end{tabular}%
17136 }%
17137 \renewcommand*\glossaryheader{\glslongextraDescSymNameTabularHeader}%
17138 \else
17139 \renewenvironment{theglossary}%
17140 {%
17141     \glspatchLToutput
17142     \glslongextraSymSetDescWidth
17143     \edef\@glslongextra@begintab{%
17144         \noexpand\begin{longtable}{%
17145             \expandonce\glslongextraDescAlign
17146             \expandonce\glslongextraSymbolAlign
17147             \expandonce\glslongextraNameAlign
17148         }}%
17149     \@glslongextra@begintab
17150 }%
17151 {\end{longtable}}%
17152 \renewcommand*\glossaryheader{\glslongextraDescSymNameHeader}%
17153 \fi
17154 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
17155 \renewcommand{\glossentry}[2]{%
17156     \glslongextraDescFmt{##1} &
17157     \glslongextraSymbolFmt{##1} &
17158     \glslongextraNameFmt{##1}\tabularnewline
17159 }%
17160 \renewcommand{\subglossentry}[3]{%
17161     \glslongextraSubDescFmt{##1}{##2} &
17162     \glslongextraSubSymbolFmt{##1}{##2} &
17163     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17164 }%
17165 \ifglsnogroupskip
17166 \renewcommand*\glsgroupskip{}%
17167 \else
17168 \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
17169 \fi
17170 }

```

scSymNameHeader

```
17171 \newcommand{\glslongextraLocationDescSymNameHeader}{%
17172   \glslongextraLocationDescSymNameTabularHeader\endhead
17173   \glslongextraLocationDescSymNameTabularFooter\endfoot
17174 }
```

meTabularHeader

```
17175 \newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
17176   \toprule
17177   \glslongextraHeaderFmt\pagelistname &
17178   \glslongextraHeaderFmt\descriptionname &
17179   \glslongextraHeaderFmt\symbolname &
17180   \glslongextraHeaderFmt\entryname\tabularnewline
17181   \midrule
17182 }
```

meTabularFooter

```
17183 \newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
17184   \bottomrule
17185 }
```

c-desc-sym-name Four column style with location list, description, symbol and name.

```
17186 \newglossarystyle{long-loc-desc-sym-name}%
17187 {%
17188   \ifGlsLongExtraUseTabular
17189     \renewenvironment{theglossary}%
17190     {%
17191       \glslongextraSymLocSetDescWidth
17192       \edef\@glslongextra@begintab{%
17193         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
17194           \expandonce\glslongextraLocationAlign
17195           \expandonce\glslongextraDescAlign
17196           \expandonce\glslongextraSymbolAlign
17197           \expandonce\glslongextraNameAlign
17198         }}%
17199       \@glslongextra@begintab
17200     }%
17201     {%
17202       \glslongextraLocationDescSymNameTabularFooter
17203       \end{tabular}%
17204     }%
17205     \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameTabularHeader}%
17206   \else
17207     \renewenvironment{theglossary}%
17208     {%
17209       \glspatchLToutput
17210       \glslongextraSymLocSetDescWidth
17211       \edef\@glslongextra@begintab{%
17212         \noexpand\begin{longtable}{%

```

```

17213         \expandonce\glslongextraLocationAlign
17214         \expandonce\glslongextraDescAlign
17215         \expandonce\glslongextraSymbolAlign
17216         \expandonce\glslongextraNameAlign
17217     } }%
17218     \@glslongextra@begintab
17219 }%
17220 {\end{longtable}}%
17221 \renewcommand*{\glossaryheader}{\glslongextraLocationDescSymNameHeader}%
17222 \fi
17223 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
17224 \renewcommand{\glossentry}[2]{%
17225     \glslongextraLocationFmt{##1}{##2} &
17226     \glslongextraDescFmt{##1} &
17227     \glslongextraSymbolFmt{##1} &
17228     \glslongextraNameFmt{##1}\tabularnewline
17229 }%
17230 \renewcommand{\subglossentry}[3]{%
17231     \glslongextraSubLocationFmt{##1}{##2}{##3} &
17232     \glslongextraSubDescFmt{##1}{##2} &
17233     \glslongextraSubSymbolFmt{##1}{##2} &
17234     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17235 }%
17236 \ifglsnogroupskip
17237     \renewcommand*{\glsgroupskip}{}%
17238 \else
17239     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
17240 \fi
17241 }

```

5 topic styles (glossary-topic.sty)

5.1 Package Initialisation and Options

Provides “topic” styles where top-level entries are considered a topic.

```
17242 \NeedsTeXFormat{LaTeX2e}
```

```
17243 \ProvidesPackage{glossary-topic}[2020/03/23 v1.44 (NLCT)]
```

Load required package.

```
17244 \RequirePackage{multicol}
```

The top-level entries act like headers. If the top-level entry has a description it's placed below the name.

topic

```
17245 \newglossarystyle{topic}{%
```

```
17246   \renewenvironment{theglossary}{%
```

```
17247     {%
```

```
17248       \glstopicInit
```

```
17249       \def\glstopic@prechildren{}}%
```

```
17250       \def\glstopic@prevlevel{-1}}%
```

```
17251     }%
```

```
17252   {\par}}%
```

```
17253   \renewcommand*{\glossaryheader}{}}%
```

```
17254   \renewcommand*{\glsgroupheading}[1]{%
```

```
17255     \def\glstopic@prevlevel{-1}}%
```

```
17256     \glstopicGroupHeading{##1}}%
```

```
17257   }%
```

```
17258   \renewcommand{\glossentry}[2]{%
```

```
17259     \hangindent0pt\relax
```

```
17260     \parindent\glstopicParIndent\relax
```

```
17261     \glstopicItem{##1}{##2}}%
```

If there isn't a description, penalise a page break.

```
17262     \ifglshasdesc{##1}}%
```

```
17263     {%
```

```
17264       \def\glstopic@prechildren{}}%
```

```
17265     }%
```

```
17266     {%
```

```
17267       \def\glstopic@prechildren{\nopagebreak}}%
```

```
17268     }%
```

```
17269   }%
```

```
17270   \renewcommand{\subglossentry}[3]{%
```

```
17271     \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
```



```

17272 \def\glstopic@prevlevel{##1}%
17273 \glstopicAssignSubIndent{##1}%
17274 \glstopicSubItem{##1}{##2}{##3}%
17275 }%
17276 \renewcommand*{\glsgroupskip}{}%
17277 }

```

`\picGroupHeading`

```
\glstopicGroupHeading{<group label>}
```

May be redefined if letter group headings are required. For example:

```

\renewcommand*{\glstopicGroupHeading}[1]{%
  \glstrgetgrouptitle{#1}{\thisgrptitle}%
  \section*{\thisgrptitle}%
}

17278 \newcommand*{\glstopicGroupHeading}[1]{%

```

`\glstopicItem`

```
\glstopicItem{<label>}{<location list>}
```

```

17279 \newcommand*{\glstopicItem}[2]{%
17280   \glsparskip\glstopicPreSkip\glsparskip\noindent
17281   \glstopicMarker{#1}%
17282   \glstopicTitleFont
17283   {%
17284     \glstentryitem{#1}\glstarget{#1}{\glstopicTitle{#1}}%
17285   }%
17286   \ifglshasdesc{#1}%
17287   {\glsparskip\nobreak\glstopicMidSkip\glsparskip\nobreak
17288     \@afterheading\glstopicDesc{#1}\glsparskip\glstopicPostSkip}%
17289   {\glsparskip\nobreak\glstopicPostSkip}%
17290   \glstopicLoc{#1}{#2}%
17291 }

```

`\glstopicMarker` May be used to insert a bookmark etc if required.

```
17292 \newcommand*{\glstopicMarker}[1]{%

```

`\glstopicName`

```

17293 \newcommand*{\glstopicTitle}[1]{\Glossentryname{#1}%
17294   \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
17295 }

```

`\glstopicTitleFont`

```
17296 \newcommand*{\glstopicTitleFont}[1]{\textbf{\large #1}}
```

```
\glstopicDesc
17297 \newcommand*{\glstopicDesc}[1]{\Glossentrydesc{#1}\glspostdescription}
```

```
\glstopicLoc
17298 \newcommand*{\glstopicLoc}[2]{}
```

```
stopicParIndent
17299 \newlength\glstopicParIndent
17300 \setlength\glstopicParIndent{20pt}
```

```
stopicSubIndent
17301 \newlength\glstopicSubIndent
17302 \setlength\glstopicSubIndent{20pt}
```

```
\glstopicInit
17303 \newcommand{\glstopicInit}{}%
```

AssignSubIndent

```
\glstopicAssignSubIndent{<level>}
```

Used to set the indentation for sub-levels.

```
17304 \newcommand*{\glstopicAssignSubIndent}[1]{%
17305   \par
17306   \parindent\dimexpr#1\glstopicSubIndent-\glstopicSubIndent\relax
17307   \glstopicAssignWidest{#1}%
17308   \hangindent\dimexpr\parindent+\glstopicwidest\relax
17309 }
```

```
\glstopicwidest
17310 \newlength\glstopicwidest
```

picAssignWidest

```
\glstopicAssignWidest{<level>}
```

Used in the definition of `\glstopicAssignSubIndent` to set the indentation from the widest name for the given level. This will require glossary-tree to set the values.

```
17311 \newcommand*{\glstopicAssignWidest}[1]{%
17312   \ifcsundef{@glswidestlength\romannumeral#1}%
17313   {%
17314     \ifcsdef{@glswidestname\romannumeral#1}%
17315     {%
17316       \settowidth{\glstopicwidest}{%
```

```

17317      \glstopicSubNameFont{\csuse{@glswidestname\romannumeral#1}}}%
17318      \glstopicSubItemSep
17319      }%
17320      }%
17321      {\setlength{\glstopicwidest}{0pt}}}%

    Save the value so that it doesn't have to keep being recalculated.

17322      \csedef{@glswidestlength\romannumeral#1}{\the\glstopicwidest}%
17323      }%
17324      {\setlength{\glstopicwidest}{\csuse{@glswidestlength\romannumeral#1}}}%
17325      }

```

glstopicPreSkip

```
17326 \newcommand*{\glstopicPreSkip}{\medskip}
```

glstopicMidSkip

```
17327 \newcommand*{\glstopicMidSkip}{\smallskip}
```

lstopicPostSkip

```
17328 \newcommand*{\glstopicPostSkip}{\smallskip}
```

glstopicSubItem

```
\glstopicSubItem{<level>}{<label>}{<location list>}
```

```

17329 \newcommand*{\glstopicSubItem}[3]{%
17330   \glstopicSubItemBox{#1}{\glstopicSubNameFont{\glstentryitem{#2}}%
17331     \glstarget{#2}{\glossentryname{#2}}}%
17332   \glstopicSubItemSep
17333   }%
17334   \ifglshassymbol{#2}{(\glossentrysymbol{#2})\space}{}%

17335   \ifglshasdesc{#2}%
17336     {\glossentrydesc{#2}\glspostdescription\glstopicSubPreLocSep}{}%
17337   \glstopicSubLoc{#2}{#3}%
17338 }

```

topicSubItemSep

```
17339 \newcommand*{\glstopicSubItemSep}{\quad}
```

topicSubItemBox

```
\glstopicSubItemBox{<level>}{<text>}
```

```

17340 \newcommand*{\glstopicSubItemBox}[2]{%
17341   \ifdim\glstopicwidest>0pt\relax\makebox[\glstopicwidest][l]{#2}\else#2\fi
17342 }

```

opicSubNameFont

```
17343 \newcommand*{\glstopicSubNameFont}[1]{\textbf{#1}}
```

picSubPreLocSep

```
17344 \newcommand*{\glstopicSubPreLocSep}{\space}
```

\glstopicSubLoc

```
17345 \newcommand*{\glstopicSubLoc}[2]{#2}
```

\glstopicCols

```
17346 \newcommand*{\glstopicCols}{2}
```

glstopicColsEnv

```
17347 \newcommand*{\glstopicColsEnv}{multicols}
```

topicmcols

```
17348 \newglossarystyle{topicmcols}{%
17349   \renewenvironment{theglossary}%
17350   {%
17351     \glstopicInit
17352     \def\glstopic@prechildren{}%
17353     \def\glstopic@postchildren{}%
17354     \def\glstopic@prevlevel{-1}%
17355   }%
17356   {%
17357     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17358     \par
17359   }%
17360   \renewcommand*{\glossaryheader}{}%
17361   \renewcommand*{\glsgroupeheading}[1]{%
17362     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17363     \def\glstopic@prevlevel{-1}%
17364     \glstopicGroupHeading{##1}%
17365   }%
17366   \renewcommand{\glossentry}[2]{%
17367     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17368     \def\glstopic@prevlevel{0}%
17369     \hangindent0pt\relax
17370     \parindent\glstopicParIndent\relax
17371     \glstopicItem{##1}{##2}%
17372     \ifnum\glstopicCols>1\relax
```

If there isn't a description, penalise a page break.

```
17373     \ifglshasdesc{##1}%
17374     {%
17375       \edef\glstopic@prechildren{%
17376         \noexpand\begin{\glstopicColsEnv}{\glstopicCols}%
17377       }%
17378     }%
```

```

17379      {%
17380      \edef\glstopic@prechildren{%
17381      \noexpand\nopagebreak
17382      \noexpand\begin{\glstopicColsEnv}{\glstopicCols}%
17383      }%
17384      }%
17385      \edef\glstopic@postchildren{\noexpand\end{\glstopicColsEnv}}%
17386      \fi
17387      }%
17388      \renewcommand{\subglossentry}[3]{%
17389      \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
17390      \def\glstopic@prevlevel{##1}%
17391      \glstopicAssignSubIndent{##1}%
17392      \glstopicSubItem{##1}{##2}{##3}%
17393      }%
17394      \renewcommand*{\glsgroupskip}{}%
17395      }

```

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)			
General: Initial experimental release	5	
0.2 (2015-11-30)			
\Glsfmtshort: new	365	
\glsfmtshort: new	364	
\Glsfmtshortpl: new	365	
\glsfmtshortpl: new	364	
short: switched inline full form to short			
(long)	256	
0.3 (2015-12-02)			
\@ACRlong: added redefinition	91	
\@ACRlongpl: added redefinition	92	
\@ACRshort: added redefinition	89	
\@ACRshortpl: added redefinition	90	
\@Acrlong: added redefinition	91	
\@Acrlongpl: added redefinition	92	
\@Acrshort: added redefinition	89	
\@Acrshortpl: added redefinition	90	
\@GLSdesc: added redefinition	85	
\@GLSdescplural@: added redefinition	.	85	
\@GLSfirst@: added redefinition	82	
\@GLSfirstplural@: added redefinition		84	
\@GLSname@: added redefinition	84	
\@GLSplural@: added redefinition	83	
\@GLSsymbol@: added redefinition	86	
\@GLSsymbolplural@: added			
redefinition	86	
\@GLStext@: added redefinition	81	
\@GLSuseri@: added redefinition	86	
\@GLSuserii@: added redefinition	87	
\@GLSuseriii@: added redefinition	87	
\@GLSuseriv@: added redefinition	87	
\@GLSuserv@: added redefinition	88	
\@GLSuservi@: added redefinition	88	
\@Glsdesc@: added redefinition	84	
\@Glsdescplural@: added redefinition	.	85	
\@Glsfirst@: added redefinition	82	
\@Glsfirstplural@: added redefinition		83	
\@Glsname@: added redefinition	84	
\@Glsplural@: added redefinition	83	
\@Glsymbol@: added redefinition	85	
\@Glsymbolplural@: added			
redefinition	86	
\@Glsxtr@defaultnoglossarywarning:			
new	150	
\@Glsxtr@field@linkdefs: new	80	
\@Glsxtr@insertdots: new	221	
\@print@glossary: added redefinition		146	
\glsabbrvdefaultfont: renamed from			
\abbrvdefaultfont	227	
\glsaccessdesc: new	180	
\glsaccessdescplural: new	181	
\glsaccessfirst: new	178	
\glsaccessfirstplural: new	178	
\Glsaccesslong: new	182	
\glsaccesslong: new	182	
\glsaccessname: new	176	
\glsaccessplural: new	177	
\Glsaccessshort: new	181	
\glsaccessshort: new	181	
\Glsaccessshortpl: new	182	

\glsaccesssshortpl: new	182	\cGLSpl: new	120
\glsaccessssymbol: new	179	\cGLSpl@: new	120
\glsaccessssymbolplural: new	179	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	177	new	115
\glstentryfmt: added check for short ..	72	\cGLS: new	119
\glslongpltok: new	221	\cGLSformat: new	120
\glsshortpltok: new	221	\cGLSpl: new	120
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	120
name in \setkeys	222	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	18
for plural	218	\glsenableentrycount: new	115
\GLSxtrlongpl: new	237	\glsfirstabbrvdefaultfont: new ..	227
\Glsxtrlongpl: new	237	\glsfirstlongdefaultfont: new ...	227
\glsxtrlongpl: new	236	\Glsfmtfirst: new	368
\glsxtrNoGlossaryWarning: new	24	\glsfmtfirst: new	368
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	369
new	217	\glsfmtfirstpl: new	369
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	367
new	217	\glsfmtplural: new	367
\glsxtrpostlinkendsentence: new ..	217	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	236	\Glsxtrtitleshort	365
\Glsxtrshortpl: new	235	renamed from \glstentryfmtshort ..	365
\glsxtrshortpl: new	234	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\Glsxtrtitleshort	364
\glslabeltok	248	renamed from \glstentryfmtshort ..	364
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	246	\Glsxtrtitleshortpl	365
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	365
redefinition of \acronymtype	20	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\Glsxtrtitleshortpl	364
\Glsxtrshort	365	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	364
\glsxtrshort	364	\Glsfmttext: new	366
\Glsfmtshortpl: changed to use		\glsfmttext: new	366
\glsxtrshortpl	365	\glshasattribute: new	194
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	194
\glsxtrshortpl	364	\glsxtremsuffix: new	297
\glsxtrifemptyglossary: new	33	\GlsXtrEnableEntryCounting: new ..	114
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	117
argument	198	\glsxtrscfont: new	264
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	264
argument	198	\glsxtrsmfont: new	280
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	281
default type to \acronymtype	129	short-em: new	305
\newterm: fixed name argument	197	short-em-desc: new	307
0.5 (2015-12-07)		short-em-footnote: new	316
\cGLS: new	119	short-em-long: new	301
\cGLS@: new	119	short-em-long-desc: new	303

short-em-postfootnote: new	319	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	275	attribute	354
short-sc-postfootnote: new	278	\Glsxtrheadtext: now uses headuc	
short-sm: new	285	attribute	357
short-sm-desc: new	286	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	292	attribute	357
short-sm-long: new	283	short-em-footnote: switch off regular	
short-sm-long-desc: new	284	attribute if set	317
short-sm-postfootnote: new	294	short-em-footnote-desc: switch off	
long-noshort-em: new	309	regular attribute if set	319
long-noshort-em-desc: new	313	short-long: switch off regular attribute	
long-noshort-sm: new	288	if set	247
long-noshort-sm-desc: new	290	short-long-desc: switch off regular	
long-short-em: new	297	attribute if set	249
long-short-em-desc: new	299	short-postfootnote-desc: switch off	
long-short-sm: new	281	regular attribute if set	255
long-short-sm-desc: new	282	short-sc-footnote: switch off regular	
0.5.1 (2015-12-02)		attribute if set	275
\Glsaccessstext: new	177	short-sc-footnote-desc: switch off	
0.5.1 (2015-12-07)		regular attribute if set	277
\@glxtr@doaccsupp: new	23	short-sm-footnote: switch off regular	
General: removed \ifglxtruseuchead	354	attribute if set	292
\Glsaccessdesc: new	180	short-sm-footnote-desc: switch off	
\Glsaccessdescplural: new	181	regular attribute if set	294
\Glsaccessfirst: new	178	long-short: switch off regular attribute	
\Glsaccessfirstplural: new	178	if set	245
\Glsaccessname: new	176	long-short-desc: switch off regular	
\Glsaccessplural: new	177	attribute if set	246
\Glsaccesssymbol: new	179	long-short-sc-desc: switch off regular	
\Glsaccesssymbolplural: new	180	attribute if set	266
\Glsxtrheadfirst: now uses headuc		footnote: switch off regular attribute if	
attribute	359	set	250
\glxtrheadfirst: now uses headuc		postfootnote: switch off regular	
attribute	358	attribute if set	253
\Glsxtrheadfirstplural: now uses		0.5.2 (2015-12-08)	
headuc attribute	360	\@GLSdesc@: added accessibility support	85
\glxtrheadfirstplural: now uses		\@GLSdescplural@: added accessibility	
headuc attribute	359	support	85
\Glsxtrheadplural: now uses headuc		\@GLSfirst@: added accessibility	
attribute	358	support	82
\glxtrheadplural: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	357	support	84
\Glsxtrheadshort: now uses headuc		\@GLSname@: added accessibility support	84
attribute	355	\@GLSplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	83
attribute	354	\@GLSsymbol@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	86
attribute	355	\@GLSsymbolplural@: added	
		accessibility support	86

\@GLStext@: added accessibility support	81	\GLSaccesslongpl: new	183, 193
\@Glsdesc@: added accessibility support	84	\Glsaccesslongpl: new	183
\@Glsdescplural@: added accessibility support	85	\glsaccesslongpl: new	183
\@Glsfirst@: added accessibility support	82	\GLSaccessname: new	176, 190
\@Glsfirstplural@: added accessibility support	83	\GLSaccessplural: new	177, 190
\@Glsname@: add accessibility support	84	\GLSaccessshort: new	182, 192
\@Glsplural@: added accessibility support	83	\GLSaccessshortpl: new	182, 192
\@Glsymbol@: added accessibility support	85	\GLSaccesssymbol: new	179, 191
\@Glsymbolplural@: added accessibility support	86	\GLSaccesssymbolplural: new	180, 191
\@Glstext@: added accessibility support	81	\GLSaccessstext: new	177, 190
\@Glsdesc@: added accessibility support	84	\glsentryfmt: moved	
\@Glsdescplural@: added accessibility support	85	\glssetabbrvfmt from	
\@Glsfirst@: added accessibility support	82	\glsxtrabbrvfmt to here	72
\@Glsfirstplural@: added accessibility support	83	\GlsXtrEnableInitialTagging: new	213
\@Glsname@: added accessibility support	84	\glsxtrfieldtitlecase: new	199
\@Glsplural@: added accessibility support	82	\GlsXtrFormatLocationList: new	70
\@Glsymbol@: added accessibility support	85	\glsxtrnewabbrevpresetkeyhook:	
\@Glsymbolplural@: added accessibility support	86	new	225
\@Glstext@: added accessibility support	81	\glsxtrtagfont: new	215
\@Glsdesc@: added accessibility support	84	\KV@printgloss@nonumberlist: added	72
\@Glsdescplural@: added accessibility support	85	\mfu@checkword@do: added	214
\@Glsfirst@: added accessibility support	82	\setabbreviationstyle: added check	
\@Glsfirstplural@: added accessibility support	83	for post-definition style switch	241
\@Glsname@: added accessibility support	84	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	82	\@glsxtr@autoindex@at: new	210
\@Glsymbol@: added accessibility support	85	\@glsxtr@autoindex@encap: new	210
\@Glsymbolplural@: added accessibility support	86	\@glsxtr@autoindex@esc: new	211
\@Glstext@: added accessibility support	81	\@glsxtr@autoindex@level: new	210
\@glsxtr@activate@initialtagging:		\@glsxtr@autoindex@setname: new	208
new	215	\@glsxtr@doabbreviationsdef: new	20
\@glsxtr@do@titlecaps@warn: new	214	General: removed	
\@glsxtr@tag: new	215	\GlsXtrNoGlsWarningNoAutoMakeMain	148
General: fixed typo in glossaries-accsupp and tidied up code to use just one		\glsdescwidth: added	69
\@ifpackageloaded	176	\glspagelistwidth: added	69
removed \glsxtrabbrvfmt	238	\glsxtrdoautoindexname: new	208
\glossaryentrynumbers: added	69	\glsxtrpostnamehook: new	205
\Glossentrydesc: added	212	\if@glsxtr@format@override: new	207
\Glossentryname: added	203	\ProvidesGlossariesExtraLang: new	373
\Glossentrysymbol: added	213	\RequireGlossariesExtraLang: new	373
\glossentrysymbol: added	212	0.5.4 (2015-12-15)	
\GLSaccessdesc: new	180, 191	\@newglossaryentry@defunitcounters:	
\GLSaccessdescplural: new	181, 192	new	121
\GLSaccessfirst: new	178, 190	\@GLSxtr@p@acrlong@: new	106
\GLSaccessfirstplural: new	179, 191	\@GLSxtr@p@acrlongpl@: new	106
\GLSaccesslong: new	183, 192	\@GLSxtr@p@acrshort@: new	105
		\@GLSxtr@p@acrshortpl@: new	106
		\@GLSxtr@p@long@: new	105
		\@GLSxtr@p@longpl@: new	105

\@GLSxtr@p@plural@: new	104	\@sGlsXtrEnableOnTheFly: new	65
\@GLSxtr@p@short@: new	104	\cGlsformat: added	120
\@GLSxtr@p@shortpl@: new	105	\cGlsformat: added	120
\@GLSxtr@p@text@: new	104	\cGlsplformat: added	121
\@GlsXtrEnableOnTheFly: new	65	\cGlsplformat: added	120
\@Glsxtr: new	66	\gl:disablehyper: added	102
\@Glsxtr@p@acrlong@: new	106	\gl:donohyperlink: added	102
\@Glsxtr@p@acrlongpl@: new	106	\gl:enableentryunitcount: new ...	123
\@Glsxtr@p@acrshort@: new	105	\gl:shasattribute: added check for	
\@Glsxtr@p@acrshortpl@: new	106	entry's existence	194
\@Glsxtr@p@long@: new	105	\gl:sifattribute: added check for	
\@Glsxtr@p@longpl@: new	105	entry's existence	195
\@Glsxtr@p@plural@: new	104	\gl:postlinkhook: added existence	
\@Glsxtr@p@short@: new	104	check	216
\@Glsxtr@p@shortpl@: new	105	\Glsxtr: new	66
\@Glsxtr@p@text@: new	104	\gl:xtr: new	66
\@Glsxtrpl: new	67	\gl:xtrcat: new	65
\@alt@gl:shyp@opt: new	99	\gl:xtrdohyperlink: added	101
\@gl:alt@hyp@opt: new	99	\gl:xtrdowrglossaryhook: new	99
\@gl:alt@hyp@opt@char: new	99	\GlsXtrEnableEntryUnitCounting:	
\@gl:alt@hyp@opt@keys: new	100	new	126
\@gl:increment@currunitcount:		\GlsXtrEnableOnTheFly: new	65
new	122	\Glsxtrpl: new	67
\@gl:local@increment@currunitcount:		\gl:xtrpl: new	66
new	122	\gl:xtrpostlocalreset: new	114
\@gl:setdefault@gl:link@opts:		\gl:xtrpostlocalunset: new	113
new	97	\gl:xtrpostreset: new	114
\@gl:xtr: new	66	\gl:xtrpostunset: new	112
\@gl:xtr@addunitcounter: new	121	\gl:xtrprotectlinks: new	103
\@gl:xtr@currunitcount: new	123	\GlsXtrSetAltModifier: new	100
\@gl:xtr@ifunitcounter: new	121	\GlsXtrSetDefaultGlsOpts: new	98
\@gl:xtr@p@acrlong@: new	106	\gl:xtrstarflywarn: new	65
\@gl:xtr@p@acrlongpl@: new	106	\GlsXtrWarning: new	67
\@gl:xtr@p@acrshort@: new	105	\MakeAcronymsAbbreviations: now	
\@gl:xtr@p@acrshortpl@: new	105	disables \setacronymstyle	129
\@gl:xtr@p@long@: new	105	1.0 (2016-01-24)	
\@gl:xtr@p@longpl@: new	105	\@gl:xtr@autoindexcrossrefs: new .	18
\@gl:xtr@p@plural@: new	104	\@gl:xtr@idx@displaynumberlist:	
\@gl:xtr@p@short@: new	104	new	140
\@gl:xtr@p@shortpl@: new	104	\@gl:xtr@idx@entrynumberlist: new	141
\@gl:xtr@p@text@: new	104	\@gl:xtr@noidx@displaynumberlist:	
\@gl:xtr@prevunitcount: new	123	new	140
\@gl:xtr@setentryunitcountunsetattr:		\@gl:xtr@noidx@entrynumberlist:	
new	126	new	141
\@gl:xtr@unitcountlist: new	121	\@gl:xtr@noidx@numberlistloop:	
\@gl:xtrpl: new	66	new	141
\@newglossaryentryposthook: added		\@gl:xtr@reg@glosslist: new	131
empty see value if not set and added		\makeglossaries: new	131
'see' to field key map	54		

1.01 (2016-02-02)	
\glxtrdiscardperiod: added check for first use	218
short-desc: fixed typo in \glxtrinlinefullformat and added missing second argument ...	258
1.02 (2016-04-25)	
\@glxtr@current@style: new	68
\Glsfmtfull: new	372
\Glsfmtfull: new	371
\Glsfmtfullpl: new	373
\Glsfmtfullpl: new	372
\Glsfmtlong: new	370
\Glsfmtlong: new	370
\Glsfmtlongpl: new	371
\Glsfmtlongpl: new	370
\Glsxtrheadfull: new	363
\Glsxtrheadfullpl: new	362
\Glsxtrheadfullpl: new	363
\Glsxtrheadfullpl: new	362
\Glsxtrheadlong: new	361
\Glsxtrheadlong: new	360
\Glsxtrheadlongpl: new	362
\Glsxtrheadlongpl: new	361
\Glsxtrtitlefull: new	363
\Glsxtrtitlefull: new	362
\Glsxtrtitlefullpl: new	364
\Glsxtrtitlefullpl: new	363
\Glsxtrtitlelong: new	361
\Glsxtrtitlelong: new	360
\Glsxtrtitlelongpl: new	362
\Glsxtrtitlelongpl: new	361
short-postfootnote-desc: added redef of \glxtrsetupfulldefs ..	255
\ifglxtrinsertinside: new	244
postfootnote: added redef of \glxtrsetupfulldefs	253
stylemods: new	24
1.03 (2016-04-27)	
\@GLSfirstplural@: bug fix: misspelt cs name	84
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	83
\@GLSfirstplural@: bug fix: misspelt cs name	83
\@GLsplural@: fixed bug \@GLsplural@ should be redefined not \@GLsplural	83
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	82
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	361
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	354
1.04 (2015-04-30)	
short-em-footnote: renamed from “footnote-em”	316
1.04 (2016-05-02)	
\@@glxtrpostloctag: new	71
\@GLSdesc@: set abbreviation and regular format	85
\@GLSdescplural@: set abbreviation and regular format	85
\@GLSfirst@: set abbreviation format ..	82
\@GLSfirstplural@: set abbreviation and regular format	84
\@GLSname@: set abbreviation and regular format	84
\@GLSplural@: set abbreviation and regular format	83
\@GLSsymbol@: set regular format	86
\@GLSsymbolplural@: set regular format	86
\@GLStext@: set abbreviation and regular format	81
\@GLSuseri@: set regular format	86
\@GLSuserii@: set regular format	87
\@GLSuseriii@: set regular format	87
\@GLSuseriv@: set regular format	87
\@GLSuseriv@: set regular format	88
\@GLSuservi@: set regular format	88
\@GLSdesc@: set abbreviation and regular format	84
\@GLSdescplural@: set abbreviation and regular format	85
\@GLSfirst@: set abbreviation and regular format	82
\@GLSfirstplural@: set abbreviation and regular format	83
\@GLSname@: set abbreviation and regular format	84
\@GLsplural@: set abbreviation and regular format	83
\@GLssymbol@: set regular format	85
\@GLssymbolplural@: set regular format	86
\@GLStext@: set abbreviation and regular format	81
\@GLSuseri@: set regular format	86
\@GLSuserii@: set regular format	87
\@GLSuseriii@: set regular format	87

\@Glsuseriv@: set regular format	87	\glslongfont: new	227
\@Glsuserv@: set regular format	88	\glslonguserfont: new	323
\@Glsuservi@: set regular format	88	\glstrassignfieldfont: new	81
\@gls@preglossaryhook: added check for entry's existence	215	\GlsXtrEnablePreLocationTag: new	70
\@glsdesc@: set abbreviation and regular format	84	\glstrfirstscfont: new	264
\@glsdescplural@: set abbreviation and regular format	85	\glstrfirstsmfont: new	281
\@glsfirst@: set abbreviation and regular format	82	\glstrlongshortdescsort: new	246
\@glsfirstplural@: set abbreviation and regular format	83	\glstrpostnamehook: added category check	205
\@glsname@: set abbreviation and regular format	84	\glstrregularfont: new	72
\@glsplural@: set abbreviation and regular format	82	\glstruserfield: new	322
\@glsymbol@: set regular format	85	\glstruserparen: new	322
\@glsymbolplural@: set regular format	86	\glstrusersuffix: new	323
\@glstext@: set abbreviation and regular format	81	\GlsXtrWarnDeprecatedAbbrStyle: new	243
\@glstr@deprecated@abbrstyle: new	243	short-em-long-em: new	303
\@glstr@do@style: new	25	short-em-long-em-desc: new	305
\@glstr@dolocktag: new	72	short-em-nolong: new	307
\@glstr@idx@entrynumberlist: switched from \let to \newcommand	141	short-em-nolong-desc: new	308
\@glstr@pagetag: new	71	short-em-postfootnote: renamed from "postfootnote-em"	319
\@glstr@pagetag: new	71	short-footnote: new	251
\@glstr@prelocktag: new	71	short-long-user: new	330
\@glstr@postlocktag: new	71	short-long-user-desc: new	331
\@glstr@prelocktag: new	71	short-nolong: new	257
\glossentrydesc: added glossdescfont attribute check	199	short-nolong-desc: new	259
\Glossentryname: added glossnamefont attribute check	203	short-postfootnote: new	254
\glossentryname: added glossnamefont attribute check	201	short-sc-footnote: renamed from "footnote-sc"	275
moved post name hook inside condition	203	short-sc-nolong: new	270
\glsabbrvmfont: new	297	short-sc-nolong-desc: new	271
\glsabbrvuserfont: new	322	short-sc-postfootnote: renamed from "postfootnote-sc"	278
\glsfirstabbrvmfont: new	297	short-sm-footnote: renamed from "footnote-sm"	292
\glsfirstabbrvuserfont: new	322	short-sm-nolong: new	286
\glsfirstlongemfont: new	297	short-sm-nolong-desc: new	288
\glsfirstlonguserfont: new	323	short-sm-postfootnote: renamed from "postfootnote-sm"	294
\glsifnotregularcategory: new	196	\letabbreviationstyle: new	243
\glslongdefaultfont: new	227	\newabbreviationstyle: bug fix: corrected test for existence	242
\glslongemfont: new	297	long-em-noshort-em: new	311
		long-em-noshort-em-desc: new	314
		long-em-short-em: new	299
		long-em-short-em-desc: new	301
		long-noshort: new	263
		long-noshort-desc: new	262

long-noshort-em: renamed from “long-em”	309	\glxtrComputeTreeIndent: new ...	449
long-noshort-em-desc: renamed from “long-desc-em”	313	\glxtrComputeTreeSubIndent: new	449
long-noshort-sc: renamed from “long-sc”	272	\glxtrtreetopindent: new	440
long-noshort-sc-desc: renamed from “long-desc-sc”	273	short-em-long: fixed incorrect font used by long form	302
long-noshort-sm: renamed from “long-sm”	288	\xglsssetwidest: new	440
long-noshort-sm-desc: renamed from \long-desc-sm	290	1.06 (2016-06-18)	
long-short-user: new	323	\@glstdoifexistsorwarn: new	17
long-short-user-desc: new	329	\@glxtr@docdefval: new	17
\renewabbreviationstyle: new	243	\@glxtr@usesee: new	54
style: new	25	General: disabled docdef key at the start of the document	31
1.05 (2016-06-10)		docdef option changed to choice	16
\eglssetwidest: new	440	\glxtr@usesee: new	54
\glsFindWidestAnyName: new	443	\glxtrusesee: new	54
\glsFindWidestAnyNameLocation: new	448	\glxtruseseeformat: new	54
\glsFindWidestAnyNameSymbol: new	446	\if@glxtrdocdefrestricted: new ..	17
\glsFindWidestAnyNameSymbolLocation: new	447	1.07 (2016-08-15)	
\glsFindWidestLevelTwo: new	444	\@@glxtrp: new	106
\glsFindWidestUsedAnyName: new ..	442	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsFindWidestUsedAnyNameLocation: new	447	\@GLSfirstplural@: added check for nohyperfirst attribute	84
\glsFindWidestUsedAnyNameSymbol: new	445	\@GLSxtrp: new	107
\glsFindWidestUsedAnyNameSymbolLocation: new	446	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsFindWidestUsedLevelTwo: new .	443	\@GLSfirstplural@: added check for nohyperfirst attribute	83
\glsFindWidestUsedTopLevelName: new	442	\@GLSxtrp: new	107
\glsfirstlongfootnotefont: new ..	249	\@gls@preglossaryhook: added \glossxtrsetpopts	215
\glsgetwidestname: new	441	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsgetwidestsubname: new	441	\@GLSfirstplural@: added check for nohyperfirst attribute	83
\glslongfootnotefont: new	249	\@glxtrinmark: new	351
\glxtrAltTreeIndent: new	439	\@glxtrnotinmark: new	352
\glxtralttreeInit: new	440	\@glxtrp: new	107
\glxtrAltTreePar: new	439	\@glxtrp@opt: new	106
\glxtrAltTreeSetHangIndent: new	449	\glossxtrsetpopts: new	106
\glxtrAltTreeSetSubHangIndent: new	449	\glsp: new	109
\glxtralttreeSubSymbolDescLocation: new	440	\glsp: new	109
\glxtralttreeSymbolDescLocation: new	439	\glxtr@entry@p: new	108
		\glxtrabbrvfootnote: new	249
		\glxtrchecknohyperfirst: new	82
		\glxtrfieldtitlecasecs: new	199
		\glxtrifinmark: new	351
		\GLSxtrp: new	110
		\Glsxtrp: new	109

\glxtrp: new	108	\glxtrassignfieldfont: added check for existence	81
\glxtrsetpopts: new	106	\glxtrresourcefile: new	151
short-long-desc: added text key	249	\printunsrtglossaries: new	158
fixed misspelling of \glsabbrvfont in plural key	249	\printunsrtglossary: new	157
long-short-desc: added missing text key	246	1.09 (2016-12-16)	
fixed misspelling of \glsabbrvfont .	246	\@glxtr@gettype: new	139
footnote: changed first forms to use \glsfirstlongfootnotefont ...	250	\@glxtr@mixed@assign@sortkey: new	140
postfootnote: removed \footnote from first keys	252	\@printglossary: redefined to save options	138
switched from \glsfirstlongfont to \glsfirstlongfootnotefont ...	254	\glxtr@makeglossaries: new	139
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glxtrifwasfirstuse	131	1.10 (2016-12-17)	
1.08 (2016-12-13)		\@GLSpl@: fixed bug caused by typo in command name	74
\@@glxtr@record: new	8	1.11 (2017-01-19)	
\@GLS@: added \@glxtr@record	74	\@glxtr@do@redef@forglsentries: new	6
\@GLSpl@: added \@glxtr@record ...	74	\@glxtr@noidx@do: new	165
\@Gls@: added \@glxtr@record	73	\@glxtr@redef@forglsentries: new .	6
\@GLspl@: added \@glxtr@record ...	74	\@glxtr@shortcutsval: new	22
\@gls@: added \@glxtr@record	73	\@glxtr@unsrt@getgrouptitle: new	164
\@gls@link@: added \@glxtr@record	74	\@print@noidx@glossary: added redefinition	143
\@gls@field@link: added \@glxtr@record	73	\glxtr@addloclistfield: added group key	14
\@gls@saveentrycounter: new	30	added location key	14
\@glsdisp: added \@glxtr@record ..	74	\glxtr@fields: new	153
\@glspl@: added \@glxtr@record ...	73	\glxtr@linkprefix: new	153
\@glxtr@dorecord: new	10	\glxtr@org@newignoredglossary: new	49
\@glxtr@err@undefaction: new	6	\glxtr@s@newignoredglossary: new	50
\@glxtr@record: new	7	\glxtr@shortcutsval: new	153
\@glxtr@warn@onexistsordo: new ...	6	\glxtr@texencoding: new	153
\@glxtr@warn@undefaction: new	6	\glxtr@writefields: new	153
\@print@unsrt@glossary: new	158	\GlsXtrLoadResources: new	152
record: added record package option ...	15	\glxtrpageref: new	46
\glsadd: added \@glxtr@record	80	\glxtrresourcefile: changed extension to .glstex	151
\glsdoifexists: now defines \glslabel	52	\newignoredglossary: added starred version	49
\glxtr@@do@wrglossary: new	30	1.12 (2017-02-03)	
\glxtr@addloclistfield: new	13	\@@glxtr@recordcounter: new	13
\glxtr@indexonly@saveentrycounter: new	13	\@gls@preglossaryhook: check for definition	215
\glxtr@record: new	155	\@glxtr@counterrecordhook: new .	155
\glxtr@resource: new	152	\@glxtr@display@loc: new	144
\glxtr@saveentrycounter: new	30	\@glxtr@docounterrecord: new ...	155
\glxtr@setup@record: new	13		

\@glxstr@longnewglossaryentry:		\glxstrfieldifinlist: new	37
new	48	\glxstrfieldlistadd: new	36
\@glxstr@noop@recordcounter: new	13	\glxstrfieldlistadd: new	36
\@glxstr@op@recordcounter: new	13	\glxstrfieldlistgadd: new	36
\@glxstr@provide@storagekey: new	33	\glxstrfieldlistxadd: new	36
\@glxstr@s@longnewglossaryentry:		\glxstrfieldxifinlist: new	37
new	48	\glxstrfmt: new	34
\@glxstrentryfmt: new	35	\GlsXtrFmtDefaultOptions: new	34
\@glxstrindexaliased: new	97	\GlsXtrFmtField: new	34
\@glxstrsetaliasnoindex: new	97	\glxstrifkeydefined: new	33
\@newglossaryentryposthook: added		\glxstrindexaliased: new	98
check for alias key	60	\GlsXtrLetField: new	41
\@no@glxstrindexaliased: new	98	\GlsXtrLetFieldToField: new	42
\@printunsrtglossary: new	158	\GlsXtrLoadResources: removed	
General: added target key to printgloss		restriction on only one per document	152
family	138	\glxstrlocrangefmt: new	145
\apptoglossarypreamble: new	47	\glxstrpostlongdescription: new	49
\csGlsXtrLetField: new	42	\glxstrprovidestoragekey: new	33
\eGlsXtrSetField: new	42	\GlsXtrRecordCounter: new	155
\gGlsXtrSetField: new	42	\glxstrresourcecount: new	152
\glxnoidxdisplayloc: added		\glxstrresourcefile: added catcode	
redefinition	144	change for @	152
\glssettocitle: added patch	50	\glxstrsetaliasnoindex: new	97
\glxstr@counterrecord: new	155	\GlsXtrSetField: new	41
\glxstr@langtag: new	153	\glxstrsetfieldifexists: new	41
\glxstr@newabbreviation: new	222	\glxstrunsrtdo: new	164
\glxstr@org@newignoredglossary:		\GlsXtrusefield: new	40
Added check for existence	49	\glxtrusefield: new	40
\glxstr@pluralsuffixes: new	153	short-postlong-user: new	327
\glxstr@provideignoredglossary:		short-postlong-user-desc: new	329
new	51	\longnewglossaryentry: added starred	
\glxstr@s@newignoredglossary:		version	48
Added check for existence	50	long-postshort-user: new	324
\glxstr@s@provideignoredglossary:		long-postshort-user-desc: new	326
new	51	postdot: new	19
\glxstrabbrvpluralsuffix: new	227	\pretoglossarypreamble: new	47
\glxstralias: new	60	\print@noop@unsrtglossaryunit:	
\glxstrcopytoglossary: new	52	new	164
\glxstrdeffield: new	41	\print@op@unsrtglossaryunit: new	163
\glxstrdisplayendloc: new	144	\printunsrtglossary: added starred	
\glxstrdisplayendlochook: new	145	form	157
\glxstrdisplaysingleloc: new	144	\printunsrtglossaryhandler: new	163
\glxstrdisplaystartloc: new	144	\printunsrtglossaryunit: new	13
\glxstrdohyperlink: added check for		\printunsrtglossaryunitsetup: new	163
alias field	101	\provideignoredglossary: new	51
\glxstredeffield: new	41	\s@glxstr@provide@storagekey: new	34
\glxstrentryfmt: new	35	\s@printunsrtglossary: new	158
\glxstrfieldddolistloop: new	36	\xGlsXtrSetField: new	42
\glxstrfieldforlistloop: new	36		

1.13 (2017-02-07)	
\@glsdisp: removed	
\@glsxtr@org@glsdisp	74
\glsxtrsetaliasnoindex: switched to	
\providecommand	97
1.14 (2017-04-18)	
\@gls@link: added redefinition	77
\@gls@noidx@getgrouptitle: new ..	141
\@gls@removespaces: new	145
\@glsxtr@do@automake@err: new ...	155
\@glsxtr@org@gloautosee: new	29
\@glsxtr@record: added third arg	7
\@glsxtr@recordsee: new	13
General: added \glsadd option	
theHvalue	79
added \glsadd option thevalue	79
\glsdisablehyper: added redefinition	102
\glsenableentrycount: fixed	
assignment of \@cGls@	116
\glsenableentryunitcount: fixed	
assignment of \@cGls@	124
\glsnavigation: new	143
\glsxtr@org@getgrouptitle: new ..	142
\glsxtr@recordsee: new	7
\glsxtr@writefields: added check for	
automake	154
\glsxtrdisplayendloc: added check	
for empty format	144
\glsxtrgetgrouptitle: new	142
\glsxtrinitwrgloss: new	75
\glsxtrlocationhyperlink: new ...	146
\glsxtrsetgrouptitle: new	142
\glsxtrsupphypernumber: new	146
\ifglsxtrwrglossbefore: new	75
1.15 (2017-05-10)	
\@glsxtr@dorecord: corrected	
premature expansion of \@glslocref	10
short-em-long-em: fixed spelling of	
\glsabbrvfont	304
short-long: fixed spelling of	
\glsabbrvfont	247
short-long-user: fixed spelling of	
\glsabbrvfont	330
short-postfootnote-desc: fixed	
spelling of \glsabbrvfont	255
short-postlong-user: fixed spelling of	
\glsabbrvfont	327
short-postlong-user-desc: fixed	
spelling of \glsabbrvfont	329
long-em-short-em: fixed spelling of	
\glsabbrvfont	300
long-postshort-user: fixed spelling of	
\glsabbrvfont	324
long-postshort-user-desc: fixed	
spelling of \glsabbrvfont	326
long-short: fixed spelling of	
\glsabbrvfont	245
long-short-user: fixed spelling of	
\glsabbrvfont	323
footnote: fixed spelling of	
\glsabbrvfont	250
postfootnote: fixed spelling of	
\glsabbrvfont	253
1.16 (2017-06-15)	
\@glo@autosee: added redefinition	29
\@gls@noidx@getgrouptitle: fixed	
bug	141
\@glsxtr@addunusedxrefs: added	
check for seealso field	61
\@glsxtr@checkgroup: use \csuse	
instead of \csname	164
\@glsxtr@dorecordnodefer: new	11
\@glsxtr@record@only@setup: added	
check for \@gls@setupsort@none ..	15
\@print@unsrt@glossary: corrected	
misspelt command	158
\@printunsrt@glossary@handler:	
new	162
\gls@checkseeallowed: added	
redefinition	29
\glsxtr@writefields: added	
\providecommand lines	153
\glsxtrautoindex: new	208
\glsxtrautoindexassignsort: new ..	209
\glsxtrautoindexentry: new	208
\glsxtrindexseealso: new	58
\glsxtrseealsolabels: new	60
\glsxtrseelist: new	57
\glsxtruseseealso: new	57
\glsxtruseseealsoformat: new	57
\seealso: new	58
autoseeindex: new	18
1.17 (2017-08-09)	
\@glsxtr@mark@wordseps: new	222
\@glsxtr@markwordseps: new	222
\@glsxtr@noidx@displaynumberlist:	
replace hard-coded ?? with	
\glsxtrundeftag	140

\@glxstr@noidx@entrynumberlist: replace hard-coded ?? with \glxstrundeftag 141	\Glsxtrsubsequentfmt: new 241
\@glxstr@noidx@numberlistloop: replace hard-coded ?? with \glxstrundeftag 141	\glxtrsubsequentfmt: new 240
\@glxstrifhyphenstart: new 332	\Glsxtrsubsequentplfmt: new 241
General: removed some inconsistencies in the abbreviation styles 244	\glxtrsubsequentplfmt: new 241
\glabbrvhyphenfont: new 333	\glxtrword: new 222
\glabbrvonlyfont: new 347	\glxtrwordsep: new 221
\glabbrvscfont: new 264	short-hyphen-long-hyphen: new ... 342
\glabbrvsmfont: new 280	short-hyphen-long-hyphen-desc: new 343
\glabbrvuserfont: initialised to default font 322	short-hyphen-postlong-hyphen: new 344
\glfirstabbrvhyphenfont: new ... 333	short-hyphen-postlong-hyphen-desc: new 346
\glfirstabbrvonlyfont: new 347	short-long-user-desc: corrected first forms 332
\glfirstabbrvscfont: new 264	short-nolong-desc-noreg: new ... 259
\glfirstabbrvsmfont: new 281	short-nolong-noreg: new 257
\glfirstlonghyphenfont: new ... 333	long-em-noshort-em-desc-noreg: new 316
\glfirstlongonlyfont: new 347	long-em-noshort-em-noreg: new ... 312
\glslonghyphenfont: new 333	long-hyphen-noshort-desc-noreg: new 335
\glslongonlyfont: new 347	long-hyphen-noshort-noreg: new .. 337
\glslonguserfont: initialised to default font 323	long-hyphen-postshort-hyphen: new 339
\glxstr@newabbreviation: added \glxstrorgshort and \glxstrorglong 222	long-hyphen-postshort-hyphen-desc: new 341
\GlsXtrDefineAcShortcuts: new 21	long-hyphen-short-hyphen: new ... 333
\glxstrgenabbrvfmt: added check for \ifglxstrinsertinside 238	long-hyphen-short-hyphen-desc: new 334
\glxstrhyphensuffix: new 333	long-noshort-desc-noreg: new 262
\glxstrifhyphenstart: new 332	long-noshort-noreg: new 263
\glxstrlonghyphen: new 338	long-only-short-only: new 348
\glxstrlonghyphennoshort: new ... 335	long-only-short-only-desc: new .. 349
\glxstrlonghyphenshort: new 332	long-short-user-desc: corrected first forms 330
\glxstrlongshortdescname: new ... 246	1.18 (2017-08-10) stylemods: changed default value to "default" 24
\glxstronlydescname: new 349	1.19 (2017-09-09) \@glxstr@defaultnumberformat: new . 7
\glxstronlydescsort: new 349	\@glxstr@dorecord: Use \@glxstr@recordloc instead of \@glxstr@recordloc 10
\glxstronlysuffix: new 347	\@glxstr@dorecordnodefer: Use \theglentrycounter for the location rather than \@glxstr@recordloc .. 11
\glxstrtparen: new 225	\@glxstr@record@setting: new 14
\glxstrposthyphenlong: new 344	\@glxstr@record@setting@alsoindex: new 14
\glxstrposthyphenshort: new 338	\@glxstrifhasfield: new 38
\glxstrposthyphensequent: new 339	
\glxstrshortdescname: new 257	
\glxstrshorthyphen: new 344	
\glxstrshorthyphenlong: new 341	
\glxstrshortlongdescname: new ... 248	
\glxstrshortlongdescsort: new ... 248	

General: added \glslink option			
theHvalue	75		
added \glslink option thevalue ...	75		
\glstr@writefields: removed			
double-quotes around \jobname ..	154		
\glstrdoautoindexname: changed			
format test	208		
\glstrhyperlink: new	102		
\glstrifhasfield: new	38		
\GlsXtrSetDefaultNumberFormat:			
new	7		
\s@glstrifhasfield: new	38		
1.20 (2017-09-11)			
\@glstrhypernameprefix: new	138		
\glsdohypertarget: added redefinition	139		
\printunrtglossaryunitsetup:			
switched from redefining			
\glolinkprefix to			
\@glstrhypernameprefix	163		
1.21 (2017-11-03)			
\@glstr@record: added check for			
default options	9		
\@glstrwrglossmark: new	27		
\glslink: changed \let to \def	103		
\glstr@checkgroup: new	164		
\glstr@defpostpunc: new	19		
\glstr@do@record@wrglossary:			
new	8		
\glstr@dosee@alsoindex@glossary:			
new	29		
\glstr@doseeglossary: new	28		
\glstr@noidx@do: removed code			
dealing with the group	166		
\glstr@record@setting@off: new ..	15		
\glstr@record@setting@only: new ..	14		
\glstr@rglstrigger@record: new ..	170		
\glstrglossentry: new	156		
\glstrnewgls: new	167		
\glstrsetaliasnoindex: changed to			
use \glstrifhasfield instead of			
\ifglshasfield	97		
\glstrwrglossmark: new	26		
\@rGLS: new	172		
\@rGLS@: new	173		
\@rGLSpl: new	173		
\@rGLSpl@: new	173		
\@rGls: new	172		
\@rGls@: new	172		
\@rGlspl: new	172		
\@rGlspl@: new	172		
\@rGls@: new	171		
\@rGlspl: new	171		
\@rGlspl@: new	171		
General: adjusted mcolalttree	455		
modified index to remove hard coded			
\space	433		
modified list to remove hard coded			
\space	422		
moved conditional outside of			
\glsgroupskip	425–432		
new	458		
redefined altlistgroup to discourage			
breaks after group headings	423		
redefined altlisthypergroup to			
discourage breaks after group			
headings	424		
redefined alttreegroup to discourage			
breaks after group headings	450		
redefined alttreehypergroup to			
discourage breaks after group			
headings	451		
redefined indexgroup to discourage			
breaks after group headings	434		
redefined indexhypergroup to			
discourage breaks after group			
headings	434		
redefined listgroup to discourage			
breaks after group headings	423		
redefined listhypergroup to			
discourage breaks after group			
headings	423		
redefined mcolalttreegroup to			
discourage breaks after group			
headings	456		
redefined mcolalttreehypergroup to			
discourage breaks after group			
headings	456		
redefined mcolalttreespannav to			
discourage breaks after group			
headings	457		
redefined mcolindexgroup to			
discourage breaks after group			
headings	452		
redefined mcolindexhypergroup to			
discourage breaks after group			
headings	452		

redefined mcolindexspannav to		\glstreechildprelocation: new ...	433
discourage breaks after group		\glstreeprelocation: new	433
headings	452	\glstriggerrecordformat: new	171
redefined mcoltreegroup to		\glseuseabbrvfont: new	238
discourage breaks after group		\glseuselongfont: new	238
headings	453	\glstr@do@alsoindex@wrglossary:	
redefined mcoltreehypergroup to		new	8
discourage breaks after group		\glstr@org@@do@wrglossary: new ..	30
headings	453	\glstr@org@dohyperlink: new	100
redefined mcoltreenonamegroup to		\glstr@setbookindexmark: new ...	463
discourage breaks after group		\glstrbookindexatendgroup: new ..	459
headings	454	\glstrbookindexbetween: new	459
redefined		\glstrbookindexbookmark: new ...	459
mcoltreenonamehypergroup to		\glstrbookindexcols: new	458
discourage breaks after group		\glstrbookindexcolspread: new ..	460
headings	454	\glstrbookindexfirstmark: new ..	464
redefined mcoltreenonamespannav to		\glstrbookindexfirstmarkfmt: new	464
discourage breaks after group		\glstrbookindexformatheader: new	459
headings	455	\glstrbookindexgroupskip: new ..	459
redefined mcoltreespannav to		\glstrbookindexlastmark: new ...	464
discourage breaks after group		\glstrbookindexlastmarkfmt: new	464
headings	454	\glstrbookindexmarkentry: new ..	463
redefined treegroup to discourage		\glstrbookindexname: new	458
breaks after group headings	437	\glstrbookindexparentchildsep:	
redefined treehypergroup to		new	459
discourage breaks after group		\glstrbookindexparentschildsep:	
headings	437	new	459
redefined treenonamegroup to		\glstrbookindexprelocation: new	458
discourage breaks after group		\glstrbookindexsubatendgroup:	
headings	438	new	459
redefined treenonamehypergroup to		\glstrbookindexsubbetween: new ..	459
discourage breaks after group		\glstrbookindexsubname: new	458
headings	439	\glstrbookindexsubprelocation:	
debug: new	27	new	458
\glssetwidest: new	440	\glstrbookindexsubsubatendgroup:	
\glsdisablehyper: added check for		new	459
existence	102	\glstrbookindexsubsubbetween:	
changed to use \def rather than \let	102	new	459
\glsenablehyper: changed to use \def		\glstrbookindexthepage: new	463
rather than \let	102	\glstrdetoklocation: new	169
\Glsfmtname: new	366	\glxtrenablerecordcount: new ...	169
\Glsfmtname: new	365	\glstrglossentry: new	155
\glshex: new	375	\glstrgroupfield: new	164
\glslstchildpostlocation: new ..	422	\Glsxrheadname: new	356
\glslstchildprelocation: new ...	422	\glxrheadname: new	356
\glslstprelocation: new	422	\GlsXtrIfFieldEqStr: new	42
\glsnahyperlink: patched	100	\glstriflabelinlist: new	163
\glseeitemformat: new	55	\glstrifrecordtrigger: new	170
\glsshowtarget: new	28		

\glxtrindexseealso: added check that the entry exists	58	\rGLSpl: new	173
\glxtrinithyperoutside: new	76	\rGlspl: new	172
\GlsXtrLocationRecordCount: new ..	169	\rglspl: new	171
\glxtrnewgls: new	166, 168	\rGLSplformat: new	174
\glxtrnewGLSlike: new	168	\rGlsplformat: new	174
\glxtrnewglslike: new	168	\rglsplformat: new	173
\glxtrnewrgls: new	168	\s@glxtrifhasfield: switched from \ifdef to \ifundef	38
\glxtrnewrglslike: new	168	1.22 (2017-11-08)	
\glxtrprelocation: new	421, 458	\@glxtr@nopostpunc: new	137
\GlsXtrRecordCount: new	169	\@glxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glxtractivatenopost	136
\glxtrrecordtriggervalue: new ..	169	\@glxtrglossentryother: new	157
\glxtrresourcefile: now disables record key	151	\glossentrynameother: new	206
\glxtrresourceinit: new	152	\glseeitemformat: switched check from regular to short	55
\GlsXtrSetRecordCountAttribute: new	169	\glxtr@setaccessdisplay: new ...	205
\Glsxtrtitlename: new	356	\glxtr@writefields: provide \glxtr@record in aux file	153
\glxtrtitlename: new	356	\glxtractivatenopost: new	137
\glxtrtitleorpdforheading: new ..	352	\glxtrbookindexprelocation: removed check for no post dot	458
\GlsXtrTotalRecordCount: new	168	\glxtrglossentryother: new	156
\glxtrwrglossmark: new	27	\glxtrnopostpunc: new	137
short-em: new	306	1.23 (2017-11-12)	
short-sc: corrected first letter uppercasing	269	\@@glxtrfmt: added check for indexing added grouping	35
short-sm: corrected first letter uppercasing	285	new	34
shortcuts: ac	23	\@glxtr@nopostpunc@postdesc: new	137
\ifglxtr@hyperoutside: new	75	\@glxtr@restore@postpunc: new ..	137
all: new	420	\@glxtrentryfmt: fixed missing label argument	35
nolong-short: new	259	\@glxtrfmt: new	34
nolong-short-em: new	308	\eglsupdatewidest: new	441
nolong-short-noreg: new	260	\gglupdatewidest: new	441
nolong-short-sc: new	271	\glsupdatewidest: new	440
nolong-short-sm: new	288	\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand	21
nopostdot: new	19	\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand	22
postpunc: new	19	\glxtrfmtdisplay: new	35
\printunsrtglossaryentryprocesshook: new	162	\glxtrifcustomdiscardperiod: new	216
\printunsrtglossarypredoglossary: new	162	\GlsXtrIfFieldUndef: new	40
\printunsrtglossaryskipentry: new	162	\glxtrrestorepostpunc: new	138
\rGLS: new	172	\s@glxtrfmt: new	34
\rGls: new	172	\s@glxtrfmt: new	34
\rgls: new	171		
\rGLSformat: new	174		
\rGlsformat: new	173		
\rglsformat: new	173		

\xglupdatewidest:new	441	\glxtrcombingdiacriticIrules:	
1.24 (2017-11-14)		new	391
\gl sadd: added \@gls@setsort	80	\glxtrcombingdiacriticIVrules:	
\glxtrforcsvfield:new	37	new	392
\glxtrlocalsetgrouptitle:new ..	142	\glxtrcombingdiacriticrules:	
1.25 (2017-11-14)		new	391
\glxtrbookindexmulticolseenv:new	460	\glxtrcontrolrules:new	390
1.25 (2017-11-24)		\glxtrcurrencyrules:new	394
\glsextrapostnamehook:new	205	\glxtrdigitrules:new	417
\glxtrfootnotename:new	249	\glxtrfractionrules:new	418
\glxtrlongnoshortdescname:new ..	260	\glxtrGeneralLatinIIIrules:new	396
\glxtrlongnoshortname:new	263	\glxtrGeneralLatinIIrules:new ..	396
\glxtrlongshortname:new	244	\glxtrGeneralLatinIrules:new ..	395
\glxtrlongshortuserdescname:new	326	\glxtrGeneralLatinIVrules:new ..	397
\glxtronlyname:new	347	\glxtrGeneralLatinVIIrules:new	400
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVIrules:new ..	398
changed to use \glxtrparen	217	\glxtrGeneralLatinVrules:new ..	398
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules:new ..	395
changed to use \glxtrparen	217	\glxtrgeneralpuncIrules:new ...	393
\glxtrshortlongname:new	247	\glxtrgeneralpuncrules:new	393
\glxtrshortlonguserdescname:new	328	\glxtrhyphenrules:new	393
\glxtrshortnolongname:new	255	\glxtrLatinA:new	401
1.26 (2018-01-05)		\glxtrLatinAA:new	403
\@glxtr@do@inc@linkcount:new ..	174	\glxtrLatinAELigature:new	403
\glslinkpresetkeys:new	76	\glxtrLatinE:new	401
\glxtr@inc@linkcount:new	76	\glxtrLatinEszettSs:new	402
\GlsXtrEnableLinkCounting:new ..	175	\glxtrLatinEszettSz:new	402
\GlsXtrIfLinkCounterDef:new	175	\glxtrLatinEth:new	402
\glxtrinclinkcounter:new	175	\glxtrLatinH:new	401
\GlsXtrLinkCounterName:new	175	\glxtrLatinI:new	401
\GlsXtrLinkCounterValue:new	175	\glxtrLatinInsularG:new	403
\GlsXtrTheLinkCounter:new	175	\glxtrLatinK:new	401
1.27 (2018-02-26)		\glxtrLatinL:new	401
\@glxtrdialecthook:new	31	\glxtrLatinLslash:new	403
General: added		\glxtrLatinM:new	401
glossaries-extra-bib2gls.sty	374	\glxtrLatinN:new	401
\Alpha:new	387	\glxtrLatinO:new	401
\Beta:new	387	\glxtrLatinOELigature:new	403
\Chi:new	388	\glxtrLatinOslash:new	403
\Digamma:new	388	\glxtrLatinP:new	402
\Epsilon:new	387	\glxtrLatinS:new	402
\Eta:new	387	\glxtrLatinSchwa:new	402
\glxtr@loaddialect:new	374	\glxtrLatinT:new	402
\glxtrBasicDigitrules:new	417	\glxtrLatinThorn:new	402
\glxtrcombingdiacriticIIrules:		\glxtrLatinWynn:new	403
new	392	\glxtrLatinX:new	402
\glxtrcombingdiacriticIrules:		\glxtrMathGreekIIrules:new	409
new	391	\glxtrMathGreekIrules:new	408

\glstredfield: changed \csedef to \protected\csedef	41	\glsaddpresetkeys: new	79
\glsxtrlocalsetgrouptitle: changed \csedef \protected\csedef	142	\glsuserdescription: new	323
\glsxtrsetgrouptitle: changed \csxdef \protected\csxdef	142	\glsxtrabbreviationfont: new	72
1.29 (2018-04-09)		\GlsXtrDualBackLink: new	377
\@gls@removespaces: added expansion	145	\GlsXtrDualField: new	377
\@glsxtr@dorecord: don't suppress expansion of \@glsrecordlocref if counter isn't page	11	\GlsXtrExpandedFmt: new	76
\@glsxtr@wrglossary@locationhyperlink: new	26	\GLSxtrlong: added \@glsxtr@record	234
\glsxtr@inc@wrglossaryctr: new ...	25	\Glsxtrlong: added \@glsxtr@record	233
\glsxtr@wrglossarylocation: new .	376	\glsxtrlong: added \@glsxtr@record	233
\GlsXtrBibTeXEntryAliases: new ..	377	\GLSxtrlongpl: added \@glsxtr@record	238
\glsxtrfieldforlistloop: corrected argument order in \forlistcsloop	36	\Glsxtrlongpl: added \@glsxtr@record	237
\GlsXtrIndexCounterLink: new	376	\glsxtrlongpl: added \@glsxtr@record	237
\GlsXtrInternalLocationHyperlink: new	25	\GLSxtrshort: added \@glsxtr@record	232
\GlsXtrProvideBibTeXFields: new .	377	\Glsxtrshort: added \@glsxtr@record	232
indexcounter: new	26	\glsxtrshort: added \@glsxtr@record	231
\setentrycounter: new	145	\GLSxtrshortpl: added \@glsxtr@record	236
1.30 (2018-04-25)		\Glsxtrshortpl: added \@glsxtr@record	235
\@glsxtr@record: added check for post-key hook	9	\glsxtrshortpl: added \@glsxtr@record	235
added check for pre-key hook	9	\GlsXtrStartUnsetBuffering: new .	112
\@GLSxtr@fullpl: added \@glsxtr@record	230	\GlsXtrStopUnsetBuffering: new ..	113
\@GlsXtrStopUnsetBuffering: new .	113	indexcounter: added check for wrglossary counter	26
\@Glsxtr@fullpl: added \@glsxtr@record	230	\s@GlsXtrStopUnsetBuffering: new	113
\@glsxtr@dorecord: don't suppress expansion of \@glsrecordlocref ..	11	1.31 (2018-05-09)	
\@glsxtr@full: added \@glsxtr@record	228	\@GlsXtrStartUnsetBuffering: new	112
\@glsxtr@fullpl: added \@glsxtr@record	229	\@gls@ifaccessattribute@set: new	184
\@glsxtr@glossadd@postkeys: new ..	10	\@gls@initaccesskeys: new ...	184, 193
\@glsxtr@glossadd@prekeys: new ...	10	\@gls@setup@default@short@access: new	186
\@glsxtr@glslink@postkeys: new ...	10	\@glsxtr@record@noglossarywarning: new	151
\@glsxtr@glslink@prekeys: new	10	\@glsxtrbuffer@nodup@unset: new .	112
\@glsxtr@local@textformat: new ...	76	General: added prefix key for glslink .	76
\@glsxtr@unset: new	112	added prefix key for printgloss ..	138
\@glsxtrbuffer@unset: new	112	changed \let to \def	138
\glsadd: added \glsaddpostsetkeys .	80	\glsaddeach: new	80
added \glsaddpresetkeys	80	\glsapturedgroup: new	375
\glsaddpostsetkeys: new	79	\glsdefpostdesc: new	216
		\glsdefpostlink: new	217
		\glsdefpostname: new	205

<code>\glsdohypertarget</code> : bug fix: ensure that new version is picked up	139	<code>\if@glxtrdocdefrestricted</code> : changed to allow for atom as well ...	17
<code>\glslistdesc</code> : new	422	<code>docdef</code> : atom	17
<code>\glslocalreseteach</code> : new	114	1.35 (2018-08-13)	
<code>\glslocalunseteach</code> : new	114	<code>\@gl@link@</code> : initialise post-link hook commands	75
<code>\glstreechilddesc</code> : new	436	1.36 (2018-08-18)	
<code>\glstreechildsymbol</code> : new	436	<code>\glxtrautoindexesc</code> : new	208
<code>\glstreedefaultnamefmt</code> : new	432	<code>\glxtrdisplaysupplc</code> : new	378
<code>\glstreedesc</code> : new	435	<code>\glxtrmultisupplcation</code> : new ...	378
<code>\glstreegroupheaderfmt</code> : added redefinition	433	1.37 (2018-11-30)	
<code>\glstreenamefmt</code> : added redefinition .	433	<code>\@glxtr@record</code> : added check for auto-add	9
<code>\glstreenavigationfmt</code> : added redefinition	433	<code>\@dGLS</code> : new	386
<code>\glstreenonamechilddesc</code> : new	438	<code>\@dGLSpl</code> : new	387
<code>\glstreenonamedesc</code> : new	437	<code>\@dGls</code> : new	386
<code>\glstreenonamesymbol</code> : new	437	<code>\@dGlspl</code> : new	386
<code>\glstreesymbol</code> : new	435	<code>\@dgls</code> : new	386
<code>\glxtr@newabbreviation</code> : added <code>\ExtraCustomAbbreviationFields</code>	222	<code>\@dglspl</code> : new	386
<code>\GlsXtrForUnsetBufferedList</code> : new	113	<code>\@gl@getcounterprefix</code> : new	30
<code>\GlsXtrIfFieldCmpNum</code> : new	39	<code>\@glslongextrawidestname</code> : new ...	467
<code>\GlsXtrIfFieldEqNum</code> : new	39	<code>\@glxtr@bibgls@removespaces</code> : new	380
<code>\GlsXtrIfFieldEqXpStr</code> : new	43	<code>\@glxtr@check@bibgls@nameref</code> : new	152
<code>\GlsXtrIfFieldNonZero</code> : new	39	<code>\@glxtr@do@nameref@record</code> : new ..	12
<code>\GlsXtrIfHasNonZeroChildCount</code> : new	376	<code>\@glxtr@get@prefixedlabel</code> : new .	385
<code>\GlsXtrIfXpFieldEqXpStr</code> : new	43	<code>\@glxtr@if@record@only</code> : new	15
<code>\glxtrpostlinkAddSymbolDescOnFirstUse</code> : new	217	<code>\@glxtr@ifnum@mmode</code> : new	12
<code>\GlsXtrRecordWarning</code> : new	149	<code>\@glxtr@labelprefixes</code> : new	384
<code>\glxtrRevertToCmarks</code> : new	351	<code>\@glxtr@prefixlabellist</code> : new ...	385
<code>\GlsXtrStandaloneGlossaryType</code> : new	156	<code>\@glxtr@providenewgls</code> : new	166
<code>\GlsXtrStandaloneSubEntryItem</code> : new	156	<code>\@glxtr@record@only@setup</code> : new ..	15
<code>\s@GlsXtrStartUnsetBuffering</code> : new	112	<code>\@glxtr@record@setting@nameref</code> : new	14
1.32 (2018-05-24)		<code>\@glxtr@use@equation@counter@or</code> : new	76
<code>\GlsXtrForeignText</code> : new	44	General: new	465
<code>\GlsXtrForeignTextField</code> : new	46	page: nameref	11
<code>\GlsXtrUnknownDialectWarning</code> : new	46	<code>\dGLS</code> : new	386
1.33 (2018-07-26)		<code>\dgls</code> : new	385
<code>\ifglsused</code> : added redefinition	48	<code>\dglsdisp</code> : new	387
1.34 (2018-07-29)		<code>\dglslink</code> : new	387
<code>\gls@begindocdefs</code> : atom	62	<code>\dGLSpl</code> : new	386
<code>\GlsXtrIfUnusedOrUndefined</code> : new ..	32	<code>\glsadd</code> : added grouping	80
<code>\glxtrNoGlossaryWarning</code> : added package warning	24	ensure that <code>\glsadd</code> performs indexing	80
		<code>\glslongextraDescAlign</code> : new	467
		<code>\glslongextraDescFmt</code> : new	465
		<code>\glslongextraDescNameHeader</code> : new	472

\glslongextraDescNameTabularFooter:	\glslongextraNameDescSymTabularFooter:
new 472	new 475
\glslongextraDescNameTabularHeader:	\glslongextraNameDescSymTabularHeader:
new 472	new 475
\glslongextraDescSymNameHeader:	\glslongextraNameDescTabularFooter:
new 484	new 467
\glslongextraDescSymNameTabularFooter:	\glslongextraNameDescTabularHeader:
new 484	new 467
\glslongextraDescSymNameTabularHeader:	\glslongextraNameFmt: new 465
new 484	\glslongextraNameSymDescHeader:
\glslongextraGroupHeading: new .. 467	new 478
\glslongextraHeaderFormat: new .. 467	\glslongextraNameSymDescLocationHeader:
\glslongextraLocationAlign: new . 467	new 479
\glslongextraLocationDescNameHeader:	\glslongextraNameSymDescLocationTabularFooter:
new 473	new 480
\glslongextraLocationDescNameTabularFooter:	\glslongextraNameSymDescLocationTabularHeader:
new 473	new 479
\glslongextraLocationDescNameTabularHeader:	\glslongextraNameSymDescTabularFooter:
new 473	new 478
\glslongextraLocationDescSymNameHeader:	\glslongextraNameSymDescTabularHeader:
new 485	new 478
\glslongextraLocationDescSymNameTabularFooter:	\glslongextraSetDescWidth: new .. 468
new 486	\glslongextraSetWidest: new 467
\glslongextraLocationDescSymNameTabularHeader:	\glslongextraSubDescFmt: new 466
new 486	\glslongextraSubLocationFmt: new 466
\glslongextraLocationFmt: new ... 466	\glslongextraSubNameFmt: new 466
\glslongextraLocationSymDescNameHeader:	\glslongextraSubSymbolFmt: new .. 466
new 482	\glslongextraSymbolAlign: new ... 467
\glslongextraLocationSymDescNameTabularFooter:	\glslongextraSymbolFmt: new 465
new 483	\glslongextraSymDescNameHeader:
\glslongextraLocationSymDescNameTabularHeader:	new 481
new 482	\glslongextraSymDescNameTabularFooter:
\glslongextraLocSetDescWidth: new 469	new 481
\glslongextraNameAlign: new 467	\glslongextraSymDescNameTabularHeader:
\glslongextraNameDescHeader: new 467	new 481
\glslongextraNameDescLocationHeader:	\glslongextraSymLocSetDescWidth:
new 470	new 469
\glslongextraNameDescLocationTabularFooter:	\glslongextraSymSetDescWidth: new 468
new 471	\glslongextraTabularVAlign: new . 469
\glslongextraNameDescLocationTabularHeader:	\glslongextraUpdateWidest: new .. 468
new 470	\glslongextraUpdateWidestChild:
\glslongextraNameDescSymHeader:	new 468
new 475	\glsrenewcommand: new 376
\glslongextraNameDescSymLocationHeader:	\glsseeitemformat: removed reference
new 476	to \glslabel 55
\glslongextraNameDescSymLocationTabularFooter:	\glsxtr@dblfloat: new 18
new 476	\glsxtr@do@autoadd: new 76
\glslongextraNameDescSymLocationTabularHeader:	\glsxtr@float: new 18
new 476	\glsxtr@record@nameref: new 155

\glstr@renewcommand: new	376	1.38 (2018-12-01)
\glstr@writefields: provide		\glslongextraNameFmt: bug fix:
\glstr@record@nameref in aux		removed double param
file	153	all: added glossary-longextra
\glstraddlabelprefix: new	384	1.39 (2019-03-22)
\GlsXtrAutoAddOnFormat: new	77	\@GlsXtrIfFieldCmpNum: new
\glstrclearlabelprefixes: new	384	\@GlsXtrIfFieldEqNum: new
\glstrdisplaylocnameref: new	378	\@GlsXtrIfFieldEqStr: new
\glstrfmtexternalnameref: new	381	\@GlsXtrIfFieldEqXpStr: new
\glstrfmtinternalnameref: new	381	\@GlsXtrIfFieldNonZero: new
\GLSXRhiername: new	56	\@GlsXtrIfXpFieldEqXpStr: new
\GLSxtrhiername: new	56	\@glstr@removespaces: changed \x to
\GlsXtrhiername: new	56	\@glo@tmp
\Glsxtrhiername: new	55	\@glstr@dorecord: added protection
\glstrhiername: new	55	for fragile commands
\glstrhiernamesep: new	57	General: added label key for
\glstridentifyglslike: new	166	printgloss
\glstrifinlabelprefixlist: new	385	\glstrbookindexlocation: new
\GlsXtrLocationField: new	165	\glstrbookindexsublocation: new
\glstrnameloclink: new	380	\glstrentryparentname: new
\glstrnamerefink: new	379	\GlsXtrIfFieldCmpNum: added starred
\glstrprependlabelprefix: new	384	version
\GlsXtrSetAltModifier: write modifier		\GlsXtrIfFieldEqNum: added starred
to aux	100	version
\glstrSetWidest: new	381	\GlsXtrIfFieldEqStr: added starred
\glstrSetWidestFallback: new	383	form
\GlsXtrStandaloneEntryName: new	156	\GlsXtrIfFieldEqXpStr: added starred
\GlsXtrStandaloneEntryOther: new	157	form
\GLSxtrusefield: new	41	\GlsXtrIfFieldNonZero: added starred
\Glsxtrusefield: fixed internal		version
command and added check for		\GlsXtrIfXpFieldEqXpStr: added
\texorpdfstring	40	starred form
\ifGlsLongExtraUseTabular: new	469	\glstrsetglossarylabel: new
floats: new	18	\glstrshortdescname: corrected to
long-desc-name: new	472	show long form as advertised in the
long-desc-sym-name: new	484	manual
long-loc-desc-name: new	474	short-desc: corrected to omit
long-loc-desc-sym-name: new	486	description key as advertised in the
long-loc-sym-desc-name: new	483	manual
long-name-desc: new	469	short-em-desc: bug fix: omit description
long-name-desc-loc: new	471	key as advertised in the manual
long-name-desc-sym: new	475	short-sc-desc: bug fix: omit description
long-name-desc-sym-loc: new	477	key as advertised in the manual
long-name-sym-desc: new	478	short-sm-desc: corrected to omit
long-name-sym-desc-loc: new	480	description key as advertised in the
long-sym-desc-name: new	481	manual
equations: new	18	\s@GlsXtrIfFieldCmpNum: new
		\s@GlsXtrIfFieldEqNum: new
		\s@GlsXtrIfFieldEqStr: new

\s@GlsXtrIfFieldEqXpStr: new	43	1.41 (2019-04-09)	General: changed \thisgrptitle to	
\s@GlsXtrIfXpFieldEqXpStr: new . . .	44		\glxtrcurrentgrptitle	463
1.4.2 (??)			\glslistgroupskip: new	422
\@glossentrysymbol: new	212		\glstopicAssignSubIndent: moved	
\glstentrypdfsymbol: new	212		\par from \glstopicSubItem	490
1.40 (2019-03-22)			\glstopicSubItem: added check for	
General: new	488		description	491
\glstopicAssignSubIndent: new . . .	490		moved \par to	
\glstopicAssignWidest: new	490		\glstopicAssignSubIndent	491
\glstopicCols: new	492		\glstopicSubLoc: moved \space to	
\glstopicColsEnv: new	492		\glstopicSubPreLocSep	492
\glstopicDesc: new	490		\glstopicSubPreLocSep: new	492
\glstopicGroupHeading: new	489		\glstreeChildDescLoc: new	436
\glstopicInit: new	490		\glstreeDescLoc: new	435
\glstopicItem: new	489		\glstreegroupskip: new	433
\glstopicLoc: new	490		\glstreePreHeader: new	433
\glstopicMarker: new	489		\glxtralttreeSymbolDescLocation:	
\glstopicMidSkip: new	491		added check for description	439
\glstopicName: new	489		topic: added penalty if no description .	488
\glstopicParIndent: new	490		topicmcols: added penalty if no	
\glstopicPostSkip: new	491		description	492
\glstopicPreSkip: new	491	1.42 (2020-02-03)		
\glstopicSubIndent: new	490		\@@glxtr@record: moved label	
\glstopicSubItem: new	491		definition outside of conditional	9
\glstopicSubItemBox: new	491		\@ACRfull: added redefinition	93
\glstopicSubItemSep: new	491		\@ACRfullpl: added redefinition	94
\glstopicSubLoc: new	492		\@Acrfull: added redefinition	93
\glstopicSubNameFont: new	492		\@Acrfullpl: added redefinition	93
\glstopicTitleFont: new	489		\@GlsXtrIfFieldValueInCsvList:	
\glstopicwidest: new	490		new	38
all: added glossary-topic	420		\@acrfull: added redefinition	93
topic: new	488		\@acrfullpl: added redefinition	93
topicmcols: new	492		\@domakeglossaries: provided	
1.40 (2019-03-31)			definition for \@domakeglossaries .	131
\glfirstabbrvdefaultfont: changed			\@gls@assign@actual: new	185
definition from \glsabbrvfont to			\@gls@entry@field: redefined	47
\glsabbrvdefaultfont for			\@gls@setup@default@access: added	
consistency	227		\glsdefaultshortaccess	186
\GlsXtrDefaultResourceOptions:			\@gls@setup@default@short@access:	
new	151		renamed to	
long-hyphen-noshort-noreg:			\@gls@setup@default@access . .	186
corrected formatting commands . . .	338		\@glslink: switched from	
\printunsrtabbreviations: new . . .	375		\glsdohyperlink to	
\printunsrtacronyms: new	374		\glxtrdohyperlink	103
\printunsrtindex: new	374		\@glxtr@abbrlists: new	129
\printunsrtindex: new	374		\@glxtr@acronymlists: new	128
\printunsrtnumbers: new	375		\@glxtr@addabbreviationlist: new	129
\printunsrtsymbols: new	375		\@glxtr@base@acrcmd: new	88

\@glxstr@doloadprefix: new	24	removed \relax and updated	
\@glxstr@org@addtoacronymlists:		\@gls@short instead of	
new	128	\glsshorttok	223
\@glxstr@org@setacronymlists: new	129	replaced explicit \spacefactor with	
\@glxstr@entryfmt: added \glslabel		\@	223
and scope	36	\glxstr@writefields: added check for	
General: added \@afterheading	452	order=letter	154
debug: showaccsupp	27	\glxstrAccSuppAbbrSetFirstLongAttrs:	
\forall abbreviations: new ...	129	new	189, 193
\forall acronyms: new	129	\glxstrAccSuppAbbrSetNameLongAttrs:	
\glsdefaultshortaccess: new	185	new	189, 193
\glsdisplaynumberlist: added	375	\glxstrAccSuppAbbrSetNameShortAttrs:	
\glsenablehyper: switched from		new	189, 193
\glsdohyperlink to		\glxstrAccSuppAbbrSetNoLongAttrs:	
\glxstrdohyperlink	102	new	188, 193
\glstrynumberlist: added	375	\glxstrAccSuppAbbrSetTextShortAttrs:	
\GLSfmtfirst: new	368	new	189, 193
\GLSfmtfirstpl: new	369	\glxstralttreeSymbolDescLocation:	
\GLSfmtfull: new	372	switched to using \glstreeDescLoc	439
\Glsfmtfull: switched pdf case to use		\glxstrassignactualsetup: new ...	185
\glspdffmtfull	372	\glxstrbookindexbookmarkprefix:	
\glspdffmtfull: switched pdf case to use		new	460
\glspdffmtfull	371	\GlsXtrDiscardUnsetBuffering: new	113
\GLSfmtfullpl: new	373	\glxstrdohyperlink: new (was former	
\Glsfmtfullpl: switched pdf case to use		redefinition of \glsdohyperlink)	101
\glspdffmtfullpl	373	\glxstrequationlocfmt: new	379
\glspdffmtfullpl: switched pdf case to use		\glxstrfieldformatcsvlist: new ...	37
\glspdffmtfullpl	372	\glxstrfieldformatlist: new	37
\GLSfmtlong: new	370	\glxstrfootnotedescname: new ...	251
\GLSfmtlongpl: new	371	\glxstrfootnotedescsort: new ...	251
\GLSfmtname: new	366	\GLSXtrhiername: switched to using	
\GLSfmtplural: new	368	\GLSfmttext and \GLSfmtname ...	56
\GLSfmttext: new	367	\GLSXtrhiername: switched to using	
\glspdffmtfull: new	371	\glsfmttext, \glsfmtname,	
\glspdffmtfullpl: new	371	\GLSfmttext and \GLSfmtname ...	56
\glseeitemformat: switched to using		\GlsXtrhiername: switched to using	
\glsfmttext and \glsfmtname ...	55	\glsfmttext and \glsfmtname ...	56
\glsshowtarget: added check for		\GlsXtrhiername: switched to using	
\glsshowtargetouter	28	\glsfmttext and \glsfmtname ...	55
\glstreeChildDescLoc: added		\glxstrhiername: switched to using	
\glstreeNoDescSymbolPreLocation		\glsfmttext and \glsfmtname ...	55
.....	436	\GlsXtrIfFieldValueInCsvList: new	37
\glstreegroupheaderskip: new ...	433	\glxstrpdfentryfmt: new	35
\glstreeNoDescSymbolPreLocation:		\glxstrprovideaccsuppcmd: new ...	188
new	435	\glxstrscsuffix: added \protect ..	264
\glxstr@newabbreviation: moved		\GlsXtrSetAltModifier: added check	100
apply abbreviation style to after		\GLSXtrtitlefirst: new	359
category key has been obtained ...	222	\GLSXtrtitlefirstplural: new ...	360
		\GLSXtrtitlefull: new	363

\GLSxtrtitlefullpl: new	364	moved \protect inside	
\GLSxtrtitlelong: new	361	\glxtrscsuffix	276
\GLSxtrtitlelongpl: new	362	short-sc-footnote-desc: new	277
\GLSxtrtitlename: new	356	short-sc-long: added missing text key	266
\GLSxtrtitleplural: new	358	moved \protect inside	
\GLSxtrtitleshort: new	355	\glxtrscsuffix	267
\GLSxtrtitleshortpl: new	356	short-sc-postfootnote: added	
\GLSxtrtitletext: new	357	missing text key	278
\glxtrusealias: new	57	moved \protect inside	
short-em: removed \protect from		\glxtrscsuffix	278
\glxtremsuffix	306	short-sc-postfootnote-desc: new .	279
short-em-desc: removed \protect		short-sm: removed \protect from	
from \glxtremsuffix	307	\glxtrsmsuffix	285
short-em-footnote: added missing text		short-sm-desc: removed \protect	
key	316	from \glxtrsmsuffix	287
removed \protect from		short-sm-footnote: added missing text	
\glxtremsuffix	317	key	292
short-em-footnote-desc: new	318	removed \protect from	
short-em-long: added missing text key	301	\glxtrsmsuffix	292
removed \protect from		short-sm-footnote-desc: new	294
\glxtremsuffix	302	short-sm-long: added missing text key	283
short-em-long-em: added missing text		removed \protect from	
key	303	\glxtrsmsuffix	283
removed \protect from		short-sm-postfootnote: added	
\glxtremsuffix	304	missing text key	294
short-em-postfootnote: added		removed \protect from	
missing text key	319	\glxtrsmsuffix	295
removed \protect from		short-sm-postfootnote-desc: new .	296
\glxtremsuffix	320	\makeglossaries: added	
short-em-postfootnote-desc: new .	321	\@domakeglossaries	131
short-footnote-desc: new	252	let \@makeglossary to \@gobble	
short-hyphen-long-hyphen: added		instead of \relax	132
missing text key	342	removed redefinition of	
short-hyphen-postlong-hyphen:		\makeglossary	132
added missing text key	344	\makenoidxglossaries: added	
short-long: added missing text key ...	247	\@domakeglossaries	63
short-long-user: added missing text		long-em-noshort-em: removed	
key	330	\protect from \glxtremsuffix .	311
short-postfootnote-desc: added		long-em-noshort-em-desc: removed	
missing text key	255	\protect from \glxtremsuffix .	315
new	254	long-em-short-em: added missing text	
short-postlong-user: added missing		key	299
text key	327	removed \protect from	
short-sc: moved \protect inside		\glxtremsuffix	300
\glxtrscsuffix	268	long-hyphen-noshort-desc-noreg:	
short-sc-desc: moved \protect inside		added missing text key	335
\glxtrscsuffix	270	long-hyphen-postshort-hyphen:	
short-sc-footnote: added missing text		added missing text key	339
key	275		

long-hyphen-short-hyphen: added missing text key	333	\s@GlsXtrIfFieldValueInCsvList: new	38
long-noshort-em: removed \protect from \glxxtremsuffix	309	\seealsoname: add check for \alsoname	58
long-noshort-em-desc: removed \protect from \glxxtremsuffix .	313	1.42 (?) postfootnote-desc: new	255
long-noshort-sc: moved \protect inside \glxtrscsuffix	272	1.43 (2020-02-28) \@glxtrentryfmt: changed \def to \edef to avoid infinite recursion ...	36
long-noshort-sc-desc: moved \protect inside \glxtrscsuffix	274	1.44 (2020-03-23) \@glxtr@assign@leveloffset: new	139
long-noshort-sm: removed \protect from \glxtrsmsuffix	289	\@glxtr@leveloffset: new	139
long-noshort-sm-desc: removed \protect from \glxtrsmsuffix .	290	\@glxtr@noidx@do: replaced \ifglshasparent with \@glxtr@ifischild	165
long-only-short-only: added missing text key	348	\@print@unsrt@innerglossary: new	161
removed \protect from \glxtronlysuffix	348	General: added groups key	139
long-postshort-user: added missing text key	324	added leveloffset key	139
long-short: added missing text key ...	245	\doifglossarynoexistsordo: switched to starred form of \ifglossaryexists	53
long-short-em: added missing text key removed \protect from \glxxtremsuffix	298	\glswriteentry: replaced \ifglused with \GlsXtrIfUnusedOrUndefined	98
long-short-sc: added missing text key moved \protect inside \glxtrscsuffix	265	\glxtr@printgloss@checkexists: new	135
long-short-sm: added missing text key removed \protect from \glxtrsmsuffix	281	\glxtralttreeSymbolDescLocation: removed duplicate description	439
long-short-user: added missing text key	323	\ifglossaryexists: added check for starred form	32
footnote: added missing text key	250	\np@glxtr@assign@leveloffset: new	139
footnote-desc: new	252	\p@glxtr@assign@leveloffset: new	139
postfootnote: added missing text key .	253	\pp@glxtr@assign@leveloffset: new	139
prefix: new	24	\printunsrtglossary: added check for \@printgloss@checkexists	157
\RestoreAcronyms: added display style	130	\printunsrtinnerglossary: new ...	159
		printunsrtglossarywrap: new	160

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	
\\$	375
\,	57
\.	19, 217, 432
\@	62, 152, 185, 223
\@@cGLS@	116, 125
\@@cGLSpl@	116, 125
\@@cGls@	116, 124
\@@cGlspl@	116, 125
\@@cgls@	116, 124
\@@cglspl@	116, 124
\@do@wrglossary	8, 10, 133
\@do@wrglossary	13, 15, 16, 30, 80, 97
\@glo@assign@sortkey	140
\@glo@list	6
\@glo@type	158
\@glossarysec	459
\@glossaryseclabel	138
\@gls@expand@field	34
\@glslocalreset	114
\@glslocalunset	113
\@glsreset	113
\@glsunset	112
\@glsxtr@autoindex@escspch	210, 211
\@glsxtr@base@acrcmd@warn	88, 130
\@glsxtr@checkspch	209–212
\@glsxtr@disabledflycommand	68
\@glsxtr@org@postdescription	137
\@glsxtr@record	15, 16
\@glsxtr@recordcounter	15, 16, 155
\@glsxtrfmt	34
\@glsxtrp	107, 108
\@glsxtrpostloctag	70
\@glsxtrpreloctag	70, 71
\@glsxtrwrglossmark	8, 9, 13, 29, 30, 58, 63, 133
\@@newglossaryentry@defcounters	115
\@@newglossaryentry@defunitcounters	123
\@par	439
\@ACRlong	103
\@ACRlongpl	103
\@ACRshort	103
\@ACRshortpl	103
\@Acrlong	103
\@Acrlongpl	103
\@Acrshort	103
\@Acrshortpl	103
\@GLS@	103, 119, 173, 386
\@GLSdesc@	85
\@GLSpl@	103, 119, 120, 173, 387
\@GLSplural@	104
\@GLSsymbol@	86
\@GLSxtr@full	229
\@GLSxtr@fullpl	230
\@GLSxtr@p@acrlong@	103
\@GLSxtr@p@acrlongpl@	103
\@GLSxtr@p@acrshort@	103
\@GLSxtr@p@acrshortpl@	103
\@GLSxtr@p@long@	103
\@GLSxtr@p@longpl@	103
\@GLSxtr@p@plural@	103
\@GLSxtr@p@short@	103
\@GLSxtr@p@shortpl@	103
\@GLSxtr@p@text@	103
\@GLSxtrlong	103, 234
\@GLSxtrlongpl	103, 238
\@GLSxtrp	111
\@GLSxtrshort	103, 232
\@GLSxtrshortpl	103, 236
\@Gls@	103, 118, 119, 172, 386
\@Gls@acrentryname	127
\@Gls@entry@field	40, 41, 94, 109, 110, 206
\@Gls@entryname	127

\@GlsXtrEnableOnTheFly	65	\@cGls@	116, 124
\@GlsXtrIfFieldCmpNum	39	\@cGlspl@	116, 125
\@GlsXtrIfFieldEqNum	39	\@cgl@	116, 124
\@GlsXtrIfFieldEqStr	42	\@cglpl@	116, 124
\@GlsXtrIfFieldEqXpStr	43	\@currentlabelname	138
\@GlsXtrIfFieldNonZero	39	\@dGLS	386
\@GlsXtrIfFieldValueInCsvList	38	\@dGLSpl	386
\@GlsXtrIfXpFieldEqXpStr	43	\@dGls	386
\@GlsXtrStartUnsetBuffering	112	\@dGlspl	386
\@GlsXtrStopUnsetBuffering	113	\@dblfloat	18
\@Glspl@	103, 118, 119, 172, 386	\@dgls	385
\@Glsplural@	104	\@dglspl	386
\@Gls@	104	\@disable@onlypremakeg	132
\@Glsxtr	66, 67	\@do@auxoutstuff	146, 147
\@Glsxtr@full	228	\@do@gl@getcounterprefix	11, 12
\@Glsxtr@fullpl	230	\@do@glsee	60, 61
\@Glsxtr@p@acrlong@	103	\@do@newglossaryentry	127, 128, 225
\@Glsxtr@p@acrlongpl@	103	\@do@seeglossary	15, 16, 28, 63, 133
\@Glsxtr@p@acrshort@	103	\@do@wrglossary	79, 170, 171
\@Glsxtr@p@acrshortpl@	103	\@domakeglossaries	63, 131
\@Glsxtr@p@long@	103	\@dtl@formatlist@handler	37
\@Glsxtr@p@longpl@	103	\@dtl@formatlist@itemsep	37
\@Glsxtr@p@plural@	103	\@dtl@formatlist@lastitem	37
\@Glsxtr@p@short@	103	\@dtl@formatlist@prelastitem	37
\@Glsxtr@p@shortpl@	103	\@dtl@formatlist@prelastitemsep	37
\@Glsxtr@p@text@	103	\@dtlformatlist	37
\@Glsxtrlong	103, 233	\@empty	31,
\@Glsxtrlongpl	103, 237		81, 89–93, 129, 137, 145, 209, 210, 228–238
\@Glsxtrp	110	\@end@glxtr@addunused	61, 62
\@Glsxtrpl	67	\@end@glxtr@gettype	135, 139
\@Glsxtrshort	103, 231	\@end@glxtr@usesee	54
\@Glsxtrshortpl	103, 235	\@end@glxtrifhyphenstart	332
\@acrlong	103	\@endfortrue	37, 205, 241, 385
\@acrlongpl	103	\@firstofone	81,
\@acrshort	103		159, 161, 185, 200, 207, 213, 214, 379, 380
\@acrshortpl	103	\@firstofthree	74,
\@addtoacronymlists	127–130		81, 89–92, 99, 228, 229, 231, 233, 235, 237
\@addtoreset	175	\@firstoftwo	82–
\@afterheading	423,		86, 90, 92, 93, 96, 99, 131, 165, 205,
	424, 434, 435, 437–439, 452–455, 463, 489		206, 219, 220, 228–230, 235–238, 351, 352
\@alt@gl@hyp@opt	99	\@float	18
\@auxout		\@for	6, 25, 37,
	11–13, 63, 71, 100, 117, 125, 132, 133,		62, 80, 114, 115, 127, 129, 130, 132, 135,
	146, 147, 151, 153–155, 166, 167, 385, 463		143, 159, 161, 169, 175, 198, 205, 213, 385
\@bibgl@restoreat	152	\@glo@alias	59, 60
\@cGLS	119	\@glo@assign@sortkey	135
\@cGLS@	116, 119, 125	\@glo@autosee	29
\@cGLSpl	120	\@glo@autoseehook	60
\@cGLSpl@	116, 120, 125	\@glo@category	121

<code>\@glo@check@sortallowed</code>	135	<code>\@gls@currentlettergroup</code>	
<code>\@glo@counterprefix</code> ...	11, 12, 31, 145, 380		143, 158, 161, 164, 165
<code>\@glo@countunit</code>	121	<code>\@gls@declareoption</code>	5
<code>\@glo@default@sorttype</code>	135	<code>\@gls@default@longpl</code>	223, 224
<code>\@glo@desc</code>	48, 49	<code>\@gls@deffile</code>	62
<code>\@glo@descplural</code>	48, 49	<code>\@gls@doautomake</code>	135, 154
<code>\@glo@group</code>	14	<code>\@gls@doautomake@err</code>	154, 155
<code>\@glo@label</code> 14, 33, 54, 59–61, 94, 102, 442–448		<code>\@gls@enablesavenonumberlist</code>	62
<code>\@glo@location</code>	14	<code>\@gls@encapchar</code>	209
<code>\@glo@loclist</code>	14	<code>\@gls@entry@count</code>	116, 117
<code>\@glo@name</code>	208	<code>\@gls@entry@field</code>	33, 34, 40,
<code>\@glo@no@assign@sortkey</code>	140		41, 59, 60, 94, 108, 110, 111, 115, 116, 157
<code>\@glo@parent</code>	443, 444	<code>\@gls@entry@unitcount</code>	125
<code>\@glo@see</code>	54, 57, 60, 61	<code>\@gls@field@font</code>	81–88
<code>\@glo@seealso</code>	59, 60	<code>\@gls@field@link</code>	81–88, 94, 95
<code>\@glo@sort</code>	208	<code>\@gls@firstaccess</code>	183, 184, 188
<code>\@glo@sorttype</code>	135, 143	<code>\@gls@firstpluralaccess</code> ...	183, 184, 188
<code>\@glo@text</code>	74	<code>\@gls@get@counterprefix</code>	31
<code>\@glo@thislettergrp</code>	164, 165	<code>\@gls@getcounterprefix</code>	11
<code>\@glo@thisvalue</code>	322	<code>\@gls@getgrouptitle</code>	142, 158, 161
<code>\@glo@tmp</code>	31, 33, 57, 94, 145	<code>\@gls@grptitle</code>	100, 143
<code>\@glo@type</code>		<code>\@gls@hyp@opt</code>	94, 95, 100,
.. 61, 100, 127, 129, 132, 136, 137, 139,			119, 120, 167, 171–173, 227–238, 385, 386
140, 143, 144, 146, 147, 150, 151, 158–161		<code>\@gls@hyp@opt@cs</code>	99
<code>\@glo@types</code>	196, 197, 442–448	<code>\@gls@ifaccessattribute@set</code>	186
<code>\@glossary@default@style</code> 68, 136, 160, 457		<code>\@gls@ifinlist</code>	163
<code>\@glossarystyle</code>	136, 160, 161	<code>\@gls@increment@currcount</code>	116
<code>\@glossentrysymbol</code>	212	<code>\@gls@increment@currunitcount</code> ...	124
<code>\@gls@</code>	103, 118, 119, 171, 386	<code>\@gls@initaccesskeys</code>	222
<code>\@gls@@automake@immediate</code>	132	<code>\@gls@keymap</code>	14, 33, 54, 59, 94, 153, 205
<code>\@gls@@link</code>	74	<code>\@gls@label</code> .	8, 9, 11, 12, 63, 99, 133, 155, 241
<code>\@gls@ReturnAfterFi</code>	145, 380	<code>\@gls@levelchar</code>	209
<code>\@gls@actualchar</code>	209	<code>\@gls@link</code>	35, 73–75, 89–93, 228–238
<code>\@gls@actuallylong</code>	185, 186	<code>\@gls@link@checkfirsthyper</code>	74, 131
<code>\@gls@actuallylongpl</code>	185–187	<code>\@gls@link@label</code>	77, 170
<code>\@gls@actualshort</code>	185–187	<code>\@gls@link@nocheckfirsthyper</code>	
<code>\@gls@actualshortpl</code>	185–187		73, 89–93, 228–238
<code>\@gls@adjustmode</code>	80	<code>\@gls@link@opts</code>	77
<code>\@gls@alt@hyp@opt</code>	100	<code>\@gls@list</code>	143
<code>\@gls@alt@hyp@opt@char</code>	99, 100	<code>\@gls@local@increment@currcount</code> ...	116
<code>\@gls@alt@hyp@opt@keys</code>	99, 100	<code>\@gls@local@increment@currunitcount</code> 124	
<code>\@gls@assign@actual</code>	186	<code>\@gls@location</code>	165, 166
<code>\@gls@automake</code>	135	<code>\@gls@loclist</code>	140, 141, 165, 166
<code>\@gls@between</code>	143	<code>\@gls@long</code>	223
<code>\@gls@checkedmkidx</code>	209–212	<code>\@gls@longaccess</code>	184, 186
<code>\@gls@checkmkidxchars</code>	58, 208	<code>\@gls@longaccesspl</code>	184, 187
<code>\@gls@codepage</code>	147	<code>\@gls@longpl</code>	185, 186, 221, 223, 224
<code>\@gls@counter</code>		<code>\@gls@map</code>	205
..... 9, 11, 12, 26, 31, 76, 77, 80, 97, 170		<code>\@gls@nameaccess</code>	183, 184, 187

\@gls@nohyperlist	49–51	\@glslocalreset	114
\@gls@noidx@do	143	\@glslocalunset	113, 114
\@gls@noidx@getgrouptitle	158, 161	\@glslongextra@begintab	469–483, 485–487
\@gls@noidx@nosanitizesort	134	\@glslongextrawidestname	467, 468
\@gls@noidx@sanitizesort	134	\@glsnextpages	136, 160
\@gls@noidx@loclist@finalsep	140	\@glsnonextpages	136, 160
\@gls@noidx@loclist@prev	140	\@glsnumberformat
\@gls@noidx@loclist@sep	140	... 9, 11, 12, 77, 80, 97, 170, 205, 208, 380	
\@gls@noref@warn	133, 144	\@glsorder	132
\@gls@org@glsnoidxdisplayloc	141	\@glspl@	103, 118, 119, 172, 386
\@gls@org@glsseeformat	141	\@glsplural@	104
\@gls@org@longpl	224	\@glspl@	104
\@gls@org@shortpl	224	\@glspl@	104
\@gls@pluralaccess	183, 184, 187	\@glspl@	104
\@gls@preglossaryhook	137, 161, 214	\@glspl@	104
\@gls@prevlevel	449–451, 455–457	\@glspl@	104
\@gls@quotechar	209	\@glspl@	104
\@gls@reference	63, 64, 132, 133	\@glspl@	104
\@gls@restoreat	62	\@glspl@	104
\@gls@saveentrycounter	15, 16, 30, 78, 80, 170	\@glspl@	104
\@gls@see@noindex	29, 151, 152	\@glspl@	104
\@gls@setdefault@glslink@opts	9, 35, 78, 98	\@glspl@	104
\@gls@setsort	78, 80	\@glspl@	104
\@gls@setup@default@access	224	\@glspl@	104
\@gls@setupsort@none	15	\@glspl@	104
\@gls@short	223, 224	\@glspl@	104
\@gls@shortaccess	184, 186–188	\@glspl@	104
\@gls@shortaccesspl	184, 186–188	\@glspl@	104
\@gls@shortpl	185, 186, 221, 224	\@glspl@	104
\@gls@sort	164	\@glspl@	104
\@gls@textaccess	183, 184, 187	\@glspl@	104
\@gls@thisHloc	31	\@glspl@	104
\@gls@thislabel	80, 114, 385–387	\@glspl@	104
\@gls@thisloc	31	\@glspl@	104
\@gls@thisval	205, 206	\@glspl@	104
\@gls@tmp	43, 44, 143, 185, 186	\@glspl@	104
\@gls@tmpb	211, 212	\@glspl@	104
\@gls@type	129, 130, 133, 135, 241, 442–448	\@glspl@	104
\@gls@write@entrycounts	116	\@glspl@	104
\@gls@write@entryunitcounts	125	\@glspl@	104
\@gls@write@entryunitcounts@do	126	\@glspl@	104
\@gls@writedef	63	\@glspl@	104
\@gls@xref	13, 58	\@glspl@	104
\@glsabbrv@current@abbreviation	223, 238	\@glspl@	104
\@glsacronymlists	127–130	\@glspl@	104
\@glsdoifexistsorwarn	17, 201–204, 206	\@glspl@	104
\@glsentry	63, 117, 125, 126	\@glspl@	104
\@glslink	79, 100, 102	\@glspl@	104
		\@glspl@	104

<code>\@glsunset</code>	112, 113	<code>\@glsxtr@check@bibgls@nameref</code>	152
<code>\@glswidestname</code>	441, 449	<code>\@glsxtr@checkgroup</code>	159, 162
<code>\@glsxtr</code>	66, 67	<code>\@glsxtr@counterrecordhook</code>	11–13
<code>\@glsxtr@do@wrglossary</code>	133	<code>\@glsxtr@csname</code>	122, 124
<code>\@glsxtr@abbreviationsdef</code>	20, 29, 30	<code>\@glsxtr@current@style</code>	68, 457
<code>\@glsxtr@abbrlists</code>	129	<code>\@glsxtr@currentunitcount</code>	122, 124
<code>\@glsxtr@accessdisplay</code>	205–207	<code>\@glsxtr@currunitcount</code>	123, 125
<code>\@glsxtr@acronymlists</code>	129, 130	<code>\@glsxtr@debugnr</code>	27
<code>\@glsxtr@activate@initialtagging</code> ..		<code>\@glsxtr@debugval</code>	27
.....	214, 215	<code>\@glsxtr@declareoption</code> ...	5, 18–20, 24, 26
<code>\@glsxtr@addabbreviationlist</code>	225	<code>\@glsxtr@defaultnoglossarywarning</code> ..	24
<code>\@glsxtr@addunitcounter</code>	121	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glsxtr@addunused</code>	62	7, 9, 77, 80, 97, 205, 208
<code>\@glsxtr@addunusedxrefs</code>	61, 62	<code>\@glsxtr@defpostpunc</code>	19, 20, 27
<code>\@glsxtr@altmodifier</code>	100	<code>\@glsxtr@deprecated@abbrstyle</code>	
<code>\@glsxtr@assign@leveloffset</code>	139	273, 275, 277,
<code>\@glsxtr@attrval</code>		279, 290, 292, 293, 296, 311, 314, 318, 321	
.....	78, 199–204, 206, 208, 212, 213	<code>\@glsxtr@dialect</code>	44, 45
<code>\@glsxtr@autoindex@at</code>	208–210	<code>\@glsxtr@disabledflycommand</code>	67
<code>\@glsxtr@autoindex@doextra@esc</code> ...	208	<code>\@glsxtr@display@loc</code>	144
<code>\@glsxtr@autoindex@encap</code>	208–210	<code>\@glsxtr@do@@wrrindex</code>	99
<code>\@glsxtr@autoindex@esc</code>	209, 211, 212	<code>\@glsxtr@do@autoadd</code>	77
<code>\@glsxtr@autoindex@escat</code>	209, 210	<code>\@glsxtr@do@glstdisablehyperinlist</code> ..	96
<code>\@glsxtr@autoindex@escencap</code> ...	209, 210	<code>\@glsxtr@do@inc@linkcount</code>	175
<code>\@glsxtr@autoindex@esclevel</code> ...	209, 210	<code>\@glsxtr@do@nameref@record</code>	11, 12
<code>\@glsxtr@autoindex@escquote</code> ...	209, 211	<code>\@glsxtr@do@record@wrglossary</code> ...	8, 15
<code>\@glsxtr@autoindex@level</code>	209, 210	<code>\@glsxtr@do@redef@forglsentries</code>	7
<code>\@glsxtr@autoindex@setname</code>	208	<code>\@glsxtr@do@style</code>	25, 374
<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 54, 59	<code>\@glsxtr@do@titlecaps@warn</code>	
<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 54, 59	200–203, 206, 214
<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 54, 59	<code>\@glsxtr@doabbreviationsdef</code>	20
<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 54, 59	<code>\@glsxtr@doaccsupp</code>	24, 27
<code>\@glsxtr@autoindex@crossrefs</code> ..	15, 17, 54, 59	<code>\@glsxtr@docdefsetting</code>	17, 63
<code>\@glsxtr@base@acrcmd</code>	88–94, 129, 130	<code>\@glsxtr@docdefval</code>	17, 62–64
<code>\@glsxtr@bibgls@removespaces</code>	380	<code>\@glsxtr@doccounterrecord</code>	13
<code>\@glsxtr@bookindex@atendgroup</code>		<code>\@glsxtr@doglossary</code>	159, 161, 162
.....	460, 461, 463	<code>\@glsxtr@doiflabelinlist</code>	163
<code>\@glsxtr@bookindex@atsubendgroup</code> ..	462	<code>\@glsxtr@doloadprefix</code>	24, 27, 30
<code>\@glsxtr@bookindex@atsubsubendgroup</code>	462	<code>\@glsxtr@doloctag</code>	70, 71
<code>\@glsxtr@bookindex@between</code> ..	460, 461, 463	<code>\@glsxtr@dorecord</code>	8, 10
<code>\@glsxtr@bookindex@sep</code>	460–462	<code>\@glsxtr@dorecordnodefer</code>	8, 10
<code>\@glsxtr@bookindex@subatendgroup</code> ..		<code>\@glsxtr@dosee@alsoindex@glossary</code> ..	16
.....	460, 461, 463	<code>\@glsxtr@doseeglossary</code>	16, 29
<code>\@glsxtr@bookindex@subbetween</code> ..	460–462	<code>\@glsxtr@dostylewarn</code>	241
<code>\@glsxtr@bookindex@subsep</code>	460–462	<code>\@glsxtr@enabletagging</code>	213
<code>\@glsxtr@bookindex@subsubatendgroup</code>		<code>\@glsxtr@end@</code>	65
.....	461, 463	<code>\@glsxtr@endescspch</code>	209–212
<code>\@glsxtr@bookindex@subsubbetween</code> ..		<code>\@glsxtr@entrycount@org@localreset</code>	116
.....	460–462	<code>\@glsxtr@entrycount@org@localunset</code>	116
<code>\@glsxtr@bookindex@groupskip</code> ...	461, 463		
<code>\@glsxtr@cat</code>	115, 127, 169, 213		

\@glsxtr@entrycount@org@reset	116	\@glsxtr@noidx@entrynumberlist	134
\@glsxtr@entrycount@org@unset	116	\@glsxtr@noidx@numberlistloop	134
\@glsxtr@entryunitcount@org@localreset		\@glsxtr@nomissingglstextnr	24
.....	124	\@glsxtr@nomissingglstextval	24
\@glsxtr@entryunitcount@org@localunset		\@glsxtr@noop@recordcounter	13, 16
.....	124	\@glsxtr@nopostpunc	137
\@glsxtr@entryunitcount@org@reset	124	\@glsxtr@nopostpunc@postdesc	137
\@glsxtr@entryunitcount@org@unset	124	\@glsxtr@notfoundinlist	220
\@glsxtr@equationsfalse	18	\@glsxtr@op@recordcounter	15, 16
\@glsxtr@err@undefaction	7, 16	\@glsxtr@optlist	67
\@glsxtr@field@linkdefs	73, 75	\@glsxtr@org@starttoc	351
\@glsxtr@floatsfalse	18	\@glsxtr@org@GLS@	74
\@glsxtr@format@overridefalse	207	\@glsxtr@org@GLSpl@	74
\@glsxtr@format@overridetrue	207	\@glsxtr@org@Gls@	73, 74
\@glsxtr@foundinlist	220	\@glsxtr@org@Glspl@	74
\@glsxtr@full	227	\@glsxtr@org@Glsxtrtitlefirst	352, 353
\@glsxtr@fullpl	229	\@glsxtr@org@Glsxtrtitlefirstplural	
\@glsxtr@get@prefixedlabel	386, 387	352, 353
\@glsxtr@gettype	135	\@glsxtr@org@Glsxtrtitlefull	352, 354
\@glsxtr@glossdescfont	199–201	\@glsxtr@org@Glsxtrtitlefullpl	352, 354
\@glsxtr@glossnamefont	201–204, 206, 207	\@glsxtr@org@Glsxtrtitlelong	352, 353
\@glsxtr@glosssymbolfont	212, 213	\@glsxtr@org@Glsxtrtitlelongpl	352, 353
\@glsxtr@gobbleto@endescspch	211, 212	\@glsxtr@org@Glsxtrtitlename	352, 353
\@glsxtr@groupheading	159, 162, 164, 165	\@glsxtr@org@Glsxtrtitleplural	352, 353
\@glsxtr@idx@displaynumberlist	134	\@glsxtr@org@Glsxtrtitleshort	352, 353
\@glsxtr@idx@entrynumberlist	134	\@glsxtr@org@Glsxtrtitleshortpl	352, 353
\@glsxtr@if@record@only	131, 154	\@glsxtr@org@Glsxtrtitletext	352, 353
\@glsxtr@ifcsstart	65	\@glsxtr@org@MakeUppercase	352, 353
\@glsxtr@ifischild	165	\@glsxtr@org@addtoacronymlists	127, 130
\@glsxtr@ifnum@mmode	76	\@glsxtr@org@checkfirsthyper	96, 131
\@glsxtr@ifpunctoken	219	\@glsxtr@org@currentfieldvalue	44
\@glsxtr@ifunitcounter	121	\@glsxtr@org@delimN	71
\@glsxtr@insert@dots	221	\@glsxtr@org@delimR	71
\@glsxtr@insert@dots@next	221	\@glsxtr@org@doseeglossary	29, 133
\@glsxtr@insertdots	186, 223	\@glsxtr@org@gloautoosee	29
\@glsxtr@label	37, 62, 175, 176, 198, 199	\@glsxtr@org@glolinkprefix	77, 79
\@glsxtr@labelprefixes	384, 385	\@glsxtr@org@gls@	73
\@glsxtr@leveloffset	139, 165	\@glsxtr@org@glsdohypertarget	139
\@glsxtr@loadstyles	420, 421	\@glsxtr@org@glsignore	71
\@glsxtr@local@textformat	77–79	\@glsxtr@org@glspl@	73
\@glsxtr@locale	44, 45	\@glsxtr@org@glsxtrtitlefirst	352, 353
\@glsxtr@longnewglossaryentry	48	\@glsxtr@org@glsxtrtitlefirstplural	
\@glsxtr@mark@wordseps	222	352, 353
\@glsxtr@mark@wordseps@next	222	\@glsxtr@org@glsxtrtitlefull	352, 353
\@glsxtr@markwordseps	223, 224	\@glsxtr@org@glsxtrtitlefullpl	352, 354
\@glsxtr@mixed@assign@sortkey	135	\@glsxtr@org@glsxtrtitlelong	352, 353
\@glsxtr@newglslike	166, 167	\@glsxtr@org@glsxtrtitlelongpl	352, 353
\@glsxtr@noidx@displaynumberlist	134	\@glsxtr@org@glsxtrtitlename	352, 353
\@glsxtr@noidx@do	164		

<code>\@glsxtr@org@glsxtrtitleorpdforheading</code>	<code>\@glsxtr@record@setting@nameref</code> ...
..... 352, 353 11, 12, 15, 31, 152, 153
<code>\@glsxtr@org@glsxtrtitleplural</code>	<code>\@glsxtr@record@setting@off</code> . 63, 100, 166
352, 353	<code>\@glsxtr@record@setting@only</code> 15, 31
<code>\@glsxtr@org@glsxtrtitleshort</code> . 352, 353	<code>\@glsxtr@recordsee</code> 15, 29, 58
<code>\@glsxtr@org@glsxtrtitleshortpl</code> 352, 353	<code>\@glsxtr@redef@for@gl@sentries</code> 7, 30
<code>\@glsxtr@org@glsxtrtitletext</code> .. 352, 353	<code>\@glsxtr@redefstyles</code> 24, 25, 374
<code>\@glsxtr@org@makeglossaries</code> ... 131, 132	<code>\@glsxtr@reg@glosslist</code> 132–135, 140
<code>\@glsxtr@org@markboth</code> 351	<code>\@glsxtr@restore@postpunc</code> 137
<code>\@glsxtr@org@markright</code> 350, 351	<code>\@glsxtr@rglstrigger@record</code> ... 171–173
<code>\@glsxtr@org@newacronymstyle</code> .. 128, 130	<code>\@glsxtr@s@longnewglossaryentry</code> 48
<code>\@glsxtr@org@postdescription</code> .. 137, 215	<code>\@glsxtr@save@preloctag</code> 70, 71
<code>\@glsxtr@org@see@noindex</code> 151, 152	<code>\@glsxtr@setentrycountunsetattr</code> ... 115
<code>\@glsxtr@org@setacronymlists</code> 130	<code>\@glsxtr@setentryunitcountunsetattr</code> 126
<code>\@glsxtr@org@setacronymstyle</code> .. 128, 130	<code>\@glsxtr@setupshortcuts</code> 22, 23, 29, 30
<code>\@glsxtr@org@theHvalue</code> 8–10	<code>\@glsxtr@shortcutsnr</code> 22
<code>\@glsxtr@org@unset@buffer</code> 112, 113	<code>\@glsxtr@shortcutsval</code> 22, 154
<code>\@glsxtr@org@prefix</code> 11	<code>\@glsxtr@swaptwo</code> 220
<code>\@glsxtr@org@printglossary</code> 67, 138	<code>\@glsxtr@tag</code> 214
<code>\@glsxtr@org@warndep</code> 223	<code>\@glsxtr@taggingcs</code> 213
<code>\@glsxtr@p@acrlong@</code> 103	<code>\@glsxtr@textformat</code> 78, 79
<code>\@glsxtr@p@acrlongpl@</code> 103	<code>\@glsxtr@theHentrycounter</code> 11
<code>\@glsxtr@p@acrshort@</code> 103	<code>\@glsxtr@theHvalue</code> ... 8–10, 75, 77–80, 170
<code>\@glsxtr@p@acrshortpl@</code> 103	<code>\@glsxtr@theentrycounter</code> 11
<code>\@glsxtr@p@long@</code> 103	<code>\@glsxtr@thevalue</code> 8–10, 75, 77–80, 170
<code>\@glsxtr@p@longpl@</code> 103	<code>\@glsxtr@thisloctag</code> 71
<code>\@glsxtr@p@plural@</code> 103	<code>\@glsxtr@titlelabel</code> 141–143, 164
<code>\@glsxtr@p@short@</code> 103	<code>\@glsxtr@tmp</code> 25, 76, 145
<code>\@glsxtr@p@shortpl@</code> 103	<code>\@glsxtr@type</code> 199
<code>\@glsxtr@p@text@</code> 103	<code>\@glsxtr@unitcountlist</code> 121
<code>\@glsxtr@pagetag</code> 70, 71	<code>\@glsxtr@unset</code> 112, 113
<code>\@glsxtr@pagetag</code> 70, 71	<code>\@glsxtr@unset@buffer</code> 112, 113
<code>\@glsxtr@prefix</code> 385	<code>\@glsxtr@unsrt@getgroup@title</code> .. 158, 161
<code>\@glsxtr@prevunitcount</code> 123	<code>\@glsxtr@use@equation@counter</code> . 9, 76, 78
<code>\@glsxtr@printglossnr</code> 138	<code>\@glsxtr@usesee</code> 54
<code>\@glsxtr@printglossopts</code> . 67, 135, 138, 160	<code>\@glsxtr@warn@onexists@ordo</code> 7, 15, 16
<code>\@glsxtr@printglossval</code> 138	<code>\@glsxtr@warn@undefaction</code> 7, 15, 16
<code>\@glsxtr@printunsrtglossaryskipentry</code>	<code>\@glsxtr@wrglossary@location@hyperlink</code>
..... 159, 162 26
<code>\@glsxtr@provide@addstoragekey</code> 34	<code>\@glsxtr@wrglossnr</code> 75
<code>\@glsxtr@provide@storagekey</code> 33	<code>\@glsxtr@wrglossval</code> 75
<code>\@glsxtr@providenewgls</code> 167	<code>\@glsxtr@buffer@nodup@unset</code> 112
<code>\@glsxtr@record</code> .. 15, 16, 73–75, 80, 228–238	<code>\@glsxtr@buffer@unset</code> 112
<code>\@glsxtr@record@noglossarywarning</code> .. 15	<code>\@glsxtr@dialecthook</code> 374
<code>\@glsxtr@record@only@setup</code> 16	<code>\@glsxtr@docdef@false</code> 64
<code>\@glsxtr@record@setting</code> . 8, 10–12, 15,	<code>\@glsxtr@entryfmt</code> 35
31, 58, 63, 64, 100, 132, 149, 151–154, 166	<code>\@glsxtr@fmt</code> 34
<code>\@glsxtr@record@setting@alsoindex</code> .	<code>\@glsxtr@glossentry</code> 155
..... 8, 10, 58, 132	<code>\@glsxtr@glossentryother</code> 157

<code>\@glxtrhypernameprefix</code>	138, 139, 163	<code>\@no@glxtrindexaliased</code>	97, 98
<code>\@glxtrifhasfield</code>	37, 38, 42, 43	<code>\@no@makeglossaries</code>	150
<code>\@glxtrifhyphenstart</code>	332	<code>\@nocounterr</code>	176
<code>\@glxtrindexaliased</code>	97	<code>\@nopostdesc</code>	137, 160
<code>\@glxtrindexcrossrefsfalse</code>	17	<code>\@onelevel@sanitize</code>	
<code>\@glxtrindexcrossrefstrue</code>	18		11, 13, 58, 67, 142, 143, 164
<code>\@glxtrinmark</code>	351	<code>\@onlypreamble</code>	68,
<code>\@glxtrlong</code>	103, 233		71, 125, 152, 155, 176, 208, 210, 211, 213
<code>\@glxtrlongpl</code>	103, 236	<code>\@org@glossaryentrynumbers</code>	
<code>\@glxtrnewgls</code>	168		136, 137, 160, 161
<code>\@glxtrnewgls@inner</code>	166, 167	<code>\@org@newglossaryentryprehook</code>	48, 49
<code>\@glxtrnewgls@innercsname</code>	167	<code>\@print@unsrt@glossary</code>	158
<code>\@glxtrnotinmark</code>	351	<code>\@print@unsrt@innerglossary</code>	160
<code>\@glxtrp</code>	108, 109	<code>\@printgloss@checkexists</code>	136, 157, 158
<code>\@glxtrp@opt</code>	106, 107	<code>\@printgloss@checkexists@allowignored</code>	
<code>\@glxtrpl</code>	66, 67		158
<code>\@glxtrpostloctag</code>	70, 72	<code>\@printgloss@setsort</code>	135, 136, 160
<code>\@glxtrpreloctag</code>	70, 72	<code>\@printglossary</code>	67, 158
<code>\@glxtrsetaliasnoindex</code>	97, 98	<code>\@printunsrt@glossary@handler</code>	159, 162
<code>\@glxtrshort</code>	103, 231	<code>\@printunsrtglossary</code>	158
<code>\@glxtrshortpl</code>	103, 235	<code>\@rGLS</code>	172
<code>\@glxtrundeftag</code>	6, 31	<code>\@rGLS@</code>	172
<code>\@glxtrwrglossmark</code>	27	<code>\@rGLSpl</code>	173
<code>\@gobble</code>	7,	<code>\@rGLSpl@</code>	173
	16, 18, 81, 129, 132, 162, 163, 221, 460–463	<code>\@rGls</code>	172
<code>\@gobbletwo</code>	130, 223	<code>\@rGls@</code>	172
<code>\@gtempa</code>	376	<code>\@rGlspl</code>	172
<code>\@ifdefinable</code>	376	<code>\@rGlspl@</code>	172
<code>\@ifglossaryexists</code>	32	<code>\@rc@ifdefinable</code>	376
<code>\@ifnextchar</code>	99, 139	<code>\@rgls</code>	171
<code>\@ifpackageloaded</code>	5, 20, 154,	<code>\@rgls@</code>	171
	176, 199, 201, 203, 205, 207, 374, 388, 419	<code>\@rglspl</code>	171
<code>\@ifstar</code>	32–34, 38, 39,	<code>\@rglspl@</code>	171
	42, 43, 48, 49, 51, 65, 99, 112, 113, 158, 213	<code>\@sGlsXtrEnableOnTheFly</code>	65
<code>\@ifundefined</code>	373, 376	<code>\@secondofthree</code>	81–
<code>\@ignored@glossaries</code>	49–52		83, 89–92, 95, 228, 230, 232, 234, 235, 237
<code>\@input</code>	152	<code>\@secondoftwo</code>	
<code>\@input@</code>	146		74, 81, 84–93, 96, 102, 131, 138,
<code>\@istfilename</code>	132		165, 205, 220, 228, 229, 231–238, 253,
<code>\@makeglossary</code>	132		255, 278, 280, 295, 297, 320, 322, 352, 353
<code>\@mfu@domakefirsttuc</code>	214	<code>\@sglsxtr@provide@storagekey</code>	33
<code>\@mfu@nocaplist</code>	214	<code>\@star@or@long</code>	376
<code>\@ne</code>	117, 126, 167	<code>\@starttoc</code>	351
<code>\@newglossaryentry@defcounters</code>	115, 123	<code>\@thirdofthree</code>	81–84, 89, 90,
<code>\@newglossaryentryposthook</code>	14, 33, 59, 94		92, 93, 95, 229, 230, 232, 234, 236, 238, 352
<code>\@newglossaryentryprehook</code>		<code>\@thirdoftwo</code>	84–88
	14, 33, 48, 49, 59, 94	<code>\@this@key</code>	205
<code>\@nil</code>	145, 164, 380	<code>\@tracklang@lang</code>	45
<code>\@nnil</code>	209, 211, 212, 219–222	<code>\@warn@nomakeglossaries</code>	147

\@xdy@main@language	146	\ac	21
\@xdycrossrefhook	58	\ACF	21
\@xdy@language	146, 147	\Acf	21
\@xdy@locationclassorder	58	\acf	21
[package	378	\ACFP	22
\\	145, 380	\Acfp	21
		\acfp	21
		\ACL	21
_	64, 148, 149	\Acl	21
		\acl	21
		\ACLP	21
		\Aclp	21
		\aclp	21
		\ACP	21
		\Acp	21
		\acp	21
		\ACRfull	93
		\Acrfull	93
		\acrfull	93
		\ACRfullfmt	93, 128
		\Acrfullfmt	93, 128
		\acrfullfmt	93, 128
		\ACRfullpl	94
		\Acrfullpl	93
		\acrfullpl	93
		\ACRfullplfmt	94, 128
		\Acrfullplfmt	94, 128
		\acrfullplfmt	93, 128
		\ACRlong	91
		\Acrlong	91
		\acrlong	91
		\ACRlongpl	92
		\Acrlongpl	92
		\acrlongpl	92
		\acronymentry	128
		\acronymfont	89–93, 105, 106, 130
		\acronymname	20
		\acronymsort	128
		\acronymtype	20, 127, 128, 130, 374
		\acrpluralsuffix	128, 154
		\ACRshort	89
		\Acrshort	89
		\acrshort	88
		\ACRshortpl	90
		\Acrshortpl	90
		\acrshortpl	90
		\ACS	21
		\Acs	21
		\acs	21

A

\AA	403
\aa	403
\AB	21
\Ab	21
\ab	20
abbreviation styles:	
footnote	252
long-hyphen-postshort-hyphen ...	338, 341
long-hyphen-short-hyphen	334, 339
long-postshort-user	326
long-short	185
long-short-user	324
nolong-short	260
short	258
short-em-footnote	318
short-em-postfootnote	321
short-footnote	252, 294
short-hyphen-long-hyphen	343, 344
short-hyphen-postlong-hyphen ...	344, 346
short-long-user	327
short-nolong	257, 259
short-nolong-desc	259
short-postfootnote	254
short-postlong-user	329
short-sc-footnote	277, 279
short-sm-postfootnote	296
\abbreviationsname	20
\abbrvpluralsuffix	
.....	154, 224, 245, 247, 250, 253,
.....	256, 258, 261, 265, 267, 268, 270, 272,
.....	274, 276, 278, 281, 283, 285, 287, 289,
.....	290, 292, 295, 298, 300, 302, 304, 306,
.....	307, 309, 311, 313, 315, 317, 320, 324,
.....	325, 327, 331, 334, 336, 339, 342, 345, 348
\ABP	21
\Abp	21
\abp	20
\AC	21
\Ac	21

<code>\ACSP</code>	21
<code>\Acsp</code>	21
<code>\acsp</code>	21
<code>\actualchar</code>	211
<code>\addtolength</code>	450
<code>\advance</code>	117, 126, 139, 152, 167
<code>\AF</code>	21
<code>\Af</code>	21
<code>\af</code>	20
<code>\AFP</code>	21
<code>\Afp</code>	21
<code>\afp</code>	21
<code>\AL</code>	21
<code>\Al</code>	21
<code>\al</code>	20
<code>\ALP</code>	21
<code>\Alp</code>	21
<code>\alp</code>	20
<code>\alsoname</code>	58
amsmath package	12, 26
<code>\AnyTrackedLanguages</code>	374, 419
<code>\appto</code> 14, 25, 33, 54, 58–60, 94, 99, 115, 123, 159, 162, 164, 207, 219, 221–223, 384, 420	
<code>\apptoglossarypreamble</code>	381–383
<code>\arabic</code>	26, 463
<code>\AS</code>	21
<code>\As</code>	21
<code>\as</code>	20
<code>\ASP</code>	21
<code>\Asp</code>	21
<code>\asp</code>	20
<code>\AtBeginDocument</code>	27, 31, 69, 154, 385
<code>\AtEndDocument</code>	61, 116, 125, 146, 147

B

babel package	208, 210, 219
<code>\begin</code>	143, 148, 149, 159, 161, 422, 424–431, 452, 454, 455, 457, 460, 469–483, 485, 486, 492, 493
<code>\begingroup</code> .. 8, 9, 35, 37, 77, 80, 97, 156– 158, 160, 175, 185, 212, 376, 379, 380, 385	
<code>\bgroup</code>	48, 49, 136
<code>bib2gls</code>	26, 35, 47, 96, 98, 100, 152, 159, 162, 164, 166, 170, 171, 373, 374, 376, 378, 381, 383, 385, 387, 494
<code>\bibglsgdelimN</code>	375
<code>\bibglshrefchar</code>	152
<code>\bibglslastDelimN</code>	375
<code>\bottomrule</code>	467, 471– 473, 475, 476, 478, 480, 481, 483, 484, 486

C

<code>\c@wrglossary</code>	26
<code>\catcode</code>	62, 152
category attributes:	
<code>accessinsertdots</code>	186
<code>apospplural</code>	224
<code>discardperiod</code>	218
<code>entrycount</code>	111, 114–116, 126
<code>externallocation</code>	378
<code>firstshortaccess</code>	188, 250, 252
<code>firsttuc</code>	203
<code>glossdesc</code>	199
<code>glossdescfont</code>	199
<code>glossname</code>	201
<code>glossnamefont</code>	201, 203
<code>headuc</code>	354
<code>indexname</code>	208
<code>indexonlyfirst</code>	98
<code>insertdots</code>	223
<code>linkcount</code>	174
<code>linkcountmaster</code>	174
<code>markshortwords</code>	223
<code>markwords</code>	223, 224, 332, 333, 342
<code>nameshortaccess</code>	187, 188
<code>nohyper</code>	96
<code>nohyperfirst</code>	82–84
<code>noshortplural</code>	224
<code>regular</code>	72, 120, 244–247, 249, 250, 252, 253, 255, 257, 259, 260, 262, 263, 266, 268, 269, 271, 273, 275, 277, 278, 280, 282–284, 286, 287, 290–292, 294–296, 299–306, 308, 310, 312–314, 316, 317, 319, 321, 323, 330, 332, 334–336, 338, 342, 343, 348, 350
<code>textformat</code>	78
<code>textshortaccess</code>	187
<code>\cdot</code>	27
<code>\centering</code>	459
<code>\cGLS</code>	21, 115, 126
<code>\cGls</code>	21, 114, 126
<code>\cgls</code>	20, 21, 114, 126
<code>\cGLSformat</code>	119
<code>\cGlsformat</code>	118
<code>\cglsformat</code>	118, 120
<code>\cGLSpl</code>	21, 115, 126
<code>\cGlspl</code>	21, 115, 126
<code>\cglspl</code>	20, 21, 114, 126
<code>\cGLSplformat</code>	119
<code>\cGlsplformat</code>	118

\cglspformat	118, 120	183–186, 209–214, 219–224, 227–238,	
\char	142	241, 332, 333, 335, 338, 342, 344, 378,	
\columnwidth	69	380, 384, 385, 419, 433, 449–451, 455–	
\count@	117, 125, 126	457, 460, 461, 467, 468, 488, 489, 492, 493	
\csappto	47	\defglsentryfmt	49–52
\csdef 33, 41, 47, 94, 95, 116, 121, 122, 124, 188, 193, 205, 216, 217, 242, 243, 253, 255, 278, 280, 295, 296, 319, 321, 324, 326, 327, 329, 339, 341, 345, 347, 421, 440		\define@boolkey	17, 18, 75, 96, 139
\cseappto	52	\define@choicekey	7, 15, 17, 19, 22, 24, 27, 75, 138
\csedef	122, 491	\define@key	14, 19, 24, 25, 33, 59, 75, 76, 79, 94, 138, 139, 183, 184, 220, 221
\csgdef	42, 49–52, 64, 70, 116, 122, 124, 125, 129, 440, 441, 463	\DefineAcronymSynonyms	22, 23
\cslet	41, 48, 49, 136	\delimN	71
\csletcs	42, 243	\delimR	71
\csname 6, 34, 50, 51, 58, 63, 68, 72, 74, 77, 80, 89–95, 97, 107, 122, 133, 143, 146, 147, 150, 151, 159–161, 167, 169, 170, 175, 199, 223, 228–238, 244, 449		\descriptionname	467, 470, 472, 473, 475, 476, 478, 479, 481, 483, 484, 486
\cspreto	47	\detokenize	65
\csuse	9, 12, 35, 36, 41, 47, 50, 60, 61, 70, 94, 95, 108– 110, 121–125, 136, 142–144, 155, 157, 163–165, 194, 205, 216, 217, 242, 243, 379, 381, 384, 440, 441, 443, 444, 467, 491	\dimen@	131, 440–448, 468
\csxdef	54, 59, 122, 125	\dimen@i	443–445
\currentglossary 136, 156, 157, 160, 460, 463		\dimen@ii	440, 441, 443–445, 468
\CurrentOption	27, 420, 421	\dimeexpr	69, 439, 468, 469, 490
\CurrentTrackedLanguage	419	\disable@keys	20, 30, 31, 64, 151
\CurrentTrackedLanguageTag	153	\do	6, 25, 37, 62, 80, 114, 115, 127, 129, 130, 132, 135, 143, 159, 161, 169, 175, 198, 205, 213, 385
\CurrentTrackedScript	419	\do@glsl@link@checkfirsthyper	35, 73–75, 78, 89–93, 228–238
\CurrentTrackedTag	374, 419	\do@glslsdisablehyperinlist	78, 97
\CustomAbbreviationFields 225, 245–248, 250, 252, 254, 256, 258, 261, 263, 264, 266–268, 270, 272, 275, 277, 278, 280–286, 288, 292, 294, 296, 298– 301, 303, 305, 307, 309, 311, 314, 317– 319, 321, 323, 324, 326, 327, 329–331, 333, 335–337, 339, 341–344, 346, 348, 349		doc package	211
D		\dolistcsloop	36
datatool-base package	12	\DTLifinlist .. 38, 77, 129, 133, 134, 140, 385	
\DeclareAcronymList	127	\DTLifint	142
\DeclareOption	5, 420	E	
\DeclareOptionX	5, 27	\eappto	13, 25, 49– 52, 129, 159, 162, 165, 186–188, 208, 420
\def	9, 11, 13–17, 28, 31, 34, 35, 37, 39, 40, 48–51, 54, 58, 59, 62, 65–67, 69, 72–94, 100, 102–106, 112, 117–120, 127, 133, 135–140, 142– 146, 158–161, 164, 165, 167, 170–173,	\edef	6, 8, 9, 11, 36, 45, 49–52, 57, 60–63, 77, 78, 80, 96, 97, 100, 102, 121, 122, 124, 129, 132, 133, 140, 142, 144–147, 152, 156, 157, 170, 175, 186, 187, 199–202, 204–206, 209, 211, 212, 220, 242, 380, 385, 419, 443, 444, 460–462, 469–483, 485, 486, 492, 493
\declssetwidest	442–448	\egroup	48, 49, 137
\else	8–13, 15, 17–20, 22–24, 28, 29, 31, 35, 40, 46, 62, 64, 65, 70, 72, 74, 79, 97, 98, 117, 129, 131, 132, 134, 136–138, 142, 144, 145, 148, 150, 152, 160, 165, 170, 171, 207–209, 211, 212, 220–222, 224, 231–238, 240, 241, 245, 246, 248, 250,		

251, 253, 254, 256–262, 265, 267, 269–
 277, 279, 282, 284–293, 295, 296, 298–
 302, 304, 306–318, 320, 321, 324–326,
 328, 331–333, 335, 338–342, 344, 346,
 348, 349, 380–384, 422, 425–432, 434,
 436, 438, 450, 457, 459–462, 470–487, 491
 \emph 185, 297
 \empty 144, 145, 185, 380, 384, 385
 \encapchar 211
 \end 143, 148,
 149, 159, 161, 422, 424–431, 453–455,
 457, 461, 470–477, 479, 480, 482–487, 493
 \end@getprefix 31
 \end@glxtr@display@loc 144
 \endcsname 6, 34, 50, 51, 58, 63, 68,
 72, 74, 77, 80, 89–95, 97, 107, 122, 133,
 143, 146, 147, 150, 151, 159–161, 167,
 169, 170, 175, 199, 223, 228–238, 244, 449
 \endfoot 467, 470, 472,
 473, 475, 476, 478, 479, 481, 482, 484, 486
 \endgroup ... 8, 10, 35, 37, 77, 80, 97, 156–
 158, 160, 175, 185, 186, 213, 376, 380, 385
 \endhead 467, 470, 472,
 473, 475, 476, 478, 479, 481, 482, 484, 486
 \ensuremath 27
 entry categories:
 abbreviation 223, 238
 general 193, 195
 index 197
 nameshortaccess 307
 \entryname 467, 468, 470, 472,
 473, 475, 476, 478, 479, 481, 483, 484, 486
 \epreto 208
 \equal 150, 217
 equation (counter) 18, 76
 \escapechar 376
 etoolbox package 5
 \expandafter 27,
 34, 35, 37, 38, 44, 54, 57, 61, 62, 65–67,
 76, 94, 95, 99, 107, 113, 120, 121, 129,
 133–135, 140, 143–145, 159–162, 164,
 165, 167, 174, 186, 199, 202, 203, 205,
 207, 208, 211, 219, 223, 224, 332, 376, 380
 \expandonce 128, 165,
 185–187, 209, 246, 336, 469–483, 485–487
 \ExtraCustomAbbreviationFields
 186–188, 222, 225

F

\fi 7–13, 15–20, 22–24, 27–
 29, 31, 35, 40, 46, 54, 58, 59, 61, 63–65,
 68–70, 72, 74, 75, 78, 79, 98, 117, 125,
 126, 129, 131, 134–139, 142, 144, 145,
 147, 148, 150–154, 159, 160, 162, 165,
 170, 171, 207–210, 212, 220–222, 224,
 231–238, 240, 241, 245, 246, 248, 250,
 251, 253, 254, 256–262, 265, 267, 269–
 277, 279, 282–293, 295, 296, 298–302,
 304, 306–318, 320, 321, 324–326, 328,
 331–333, 335, 338–342, 344, 346, 348,
 349, 374, 380–384, 422, 425–432, 434,
 436, 438, 440–448, 450, 457, 459–462,
 468, 470–482, 484, 485, 487, 488, 491–493
 first use 494
 flag 494
 text 494
 \firstacronymfont 130, 131
 fontspec package 154
 \footnote 249
 \forall abbreviations 129
 \forall glossaries
 61, 158, 196, 197, 199, 442–448
 \forall glossaries 63, 117, 126
 \ForEachTrackedDialect 374, 419
 \foreignlanguage 44, 45
 \for glossaries 6, 61, 197, 199, 442–448
 \forlistcsloop 36, 37, 126, 143
 \forlistloop 113, 140, 141, 214
 \futurelet 219

G

\gdef 71, 210, 211
 \Genacrfullformat 128
 \genacrfullformat 128
 \GenericAcronymFields 128
 \Genplacrfullformat 128
 \genplacrfullformat 128
 \GetTrackedDialectFromLanguageTag .. 44
 \GetTrackedDialectToMapping 45
 \glo@grabfirst 164
 \glo@name 202, 203, 207
 \gloaliaslabel 101
 \global 11, 48, 49, 62, 137, 161, 165
 \glo link prefix . 76, 77, 79, 101, 102, 138, 154
 glossaries package 12, 15, 28–30, 46,
 47, 54, 58–60, 131, 132, 135, 157, 184, 421
 glossaries-accsupp package
 23, 24, 27, 176, 185, 224

glossaries-extra package	2, 47, 419	tree	435
glossaries-extra-bib2gls package		treegroup	437
.....	15, 16, 31, 374, 419, 468	treehypergroup	437
glossaries-extra-stylemods package		treenoname	437
.....	24, 215, 374, 382	treenonamegroup	438
glossaries-prefix package	24, 27, 385	treenonamehypergroup	439
glossaries-stylemods package	458	glossary-bookindex package	421
glossaries.sty package	49	glossary-hypernav package	100
\GlossariesExtraWarning	6,	glossary-long package	426
.....	16, 18, 24, 31, 44–47, 65, 67, 78, 88, 130,	glossary-longbooktabs package	426
	133, 145, 148, 151, 152, 158, 161, 199–	glossary-longextra package	381–383
	201, 203, 204, 206, 213, 214, 243, 376, 384	glossary-tree package	382, 383, 433, 490
\GlossariesExtraWarningNoLine	\glossaryentrynumbers
.....	18, 117, 126, 154	72, 136, 137, 160, 161, 165, 166
\GlossariesWarning	\glossaryheader
.....	70, 127, 134, 136, 140, 141, 160, 241	143, 159, 161, 422–431, 433,
\GlossariesWarningNoLine		434, 436–439, 449, 451–454, 456, 461,
133, 147			470, 471, 473–477, 479, 480, 482–488, 492
glossary styles:		\glossaryname
altlist	423	136
altlistgroup	423	\glossarypostamble
altlisthypergroup	424	143, 159, 161, 164
alttree	381, 382, 439, 440, 449, 467	\glossarypreamble
alttreegroup	450	143, 158, 161
alttreehypergroup	451	\glossarysection
index	433	143, 144, 150, 151, 158, 161, 163
indexgroup	434	\glossarytitle
indexhypergroup	434	50,
inline	432		51, 136, 143, 144, 150, 151, 158, 160, 161
list	422, 423	\glossarytoctitle
listdotted	421	50, 51,
listdottedstyle	422		136, 138, 143, 144, 150, 151, 158, 160, 161
listgroup	423	\glossentry
listhypergroup	423	136, 161, 166, 421–431, 433, 436,
longragged-booktabs	469		438, 449, 461, 470, 471, 473, 474, 476,
mcolalttree	455		477, 479, 480, 482, 484, 485, 487, 488, 492
mcolalttreegroup	456	\Glossentrydesc
mcolalttreehypergroup	456	421, 422, 424–432, 435–438, 465, 491
mcolalttreespannav	457	\Glossentryname
mcolindexgroup	452	489
mcolindexhypergroup	452	\glossentryname
mcolindexspannav	452	156,
mcoltreegroup	453		421–431, 434, 436, 438, 450, 458, 465, 491
mcoltreehypergroup	453	\glossentrynameother
mcoltreenonamegroup	454	157
mcoltreenonamehypergroup	454	\glossentrysymbol
mcoltreenonamespannav	455	426,
mcoltreespannav	454		428, 430, 432, 435, 437, 439, 465, 489, 491
name-desc	472	\glossxtrsetpopts
sublistdotted	422	215
		\GLS
		115, 126, 169
		\Gls
		66, 114, 126, 169
		\gls
		46, 66, 68, 114, 126, 134, 148, 169
		\gls@assign@desc
		48, 49
		\gls@assign@field
		14, 33, 94
		\gls@checkseeallowed
		62, 64, 133
		\gls@codepage
		147
		\gls@defdocnewglossaryentry
		62, 115, 123

<code>\gls@defglossaryentry</code> ...	48, 49, 63, 66, 67	<code>\Glsaccesslong</code>	91, 226,
<code>\gls@dotocitle</code>	136, 160		234, 245, 256, 261, 262, 265, 271–274,
<code>\gls@glossary</code>	58		282, 288, 289, 291, 298, 300, 309–315,
<code>\gls@grplabel</code>	100		324–326, 334, 336, 337, 340, 343, 348, 349
<code>\gls@ifnotmeasuring</code>	12, 114	<code>\glsaccesslong</code>	91, 92,
<code>\gls@level</code>	165		225, 233, 234, 245, 248, 250, 251, 254,
<code>\gls@noidxglossary</code>	133		256, 258, 260–262, 265, 267, 269–277,
<code>\gls@org@glossaryentryfield</code> ...	136, 161		279, 282, 284–291, 293, 296, 298, 300,
<code>\gls@org@glossarysubentryfield</code>			302, 304, 306–318, 320, 321, 324, 325,
	136, 137, 161		328, 331, 334, 336, 337, 340, 343, 348, 349
<code>\gls@orgTrackLangRequireDialectPrefix</code>		<code>\Glsaccesslongpl</code>	92,
	419		226, 237, 246, 256, 261, 262, 265, 272–
<code>\gls@save@numberlist</code>	69, 70, 72		274, 282, 288–291, 299, 301, 309–316,
<code>\gls@set@xr@key</code>	59		324–326, 334, 336, 337, 340, 343, 348, 349
<code>\gls@tmplen</code>	442–448, 450, 468, 469	<code>\glsaccesslongpl</code>	92, 93,
<code>\gls@type</code>	133		226, 237, 238, 245, 248, 251, 254, 256,
<code>\glsabbrvdefaultfont</code> ...	227, 245, 247,		258, 260–262, 265, 267, 269–277, 279,
	250, 253, 256, 258, 261, 322, 333, 336, 347		282, 284–293, 296, 298, 300, 302, 304,
<code>\glsabbrvmfont</code>			306–308, 310–318, 320, 321, 324, 325,
	297–307, 309, 311, 313, 315, 317, 319–321		328, 331, 334, 336, 337, 340, 343, 348, 349
<code>\glsabbrvfont</code>		<code>\GLSaccessname</code>	84
	104, 105, 130, 231, 232, 235, 236,	<code>\Glsaccessname</code>	84
	238, 240, 241, 244–253, 255–258, 261,	<code>\glsaccessname</code>	84
	263, 265, 267, 268, 270, 272, 274, 276,	<code>\GLSaccessplural</code>	83
	278, 281, 283, 285, 287, 289, 290, 292,	<code>\Glsaccessplural</code>	83
	295, 298, 300, 302, 304, 306, 307, 309,	<code>\glsaccessplural</code>	83
	311, 313, 315, 317, 320, 324, 325, 327,	<code>\Glsaccessshort</code>	
	330–332, 334, 336, 339, 340, 342, 345, 348		.. 89, 232, 241, 248, 251, 254, 258, 260,
<code>\glsabbrvhyphenfont</code>			267, 269, 270, 276, 277, 279, 284, 285,
	333–335, 339–343, 345, 346		287, 293, 295, 296, 302, 304, 306, 307,
<code>\glsabbrvonlyfont</code>	347–350		317, 318, 320, 321, 328, 331, 340, 345, 346
<code>\glsabbrvscfont</code>		<code>\glsaccessshort</code> ..	89, 225, 226, 231, 232,
	264–268, 270, 272, 274–278, 280		240, 245, 246, 248, 250, 251, 253, 254,
<code>\glsabbrvsmfont</code>			256–262, 265, 267, 269–271, 273, 274,
	281–287, 289, 290, 292, 294–296		276, 279, 282, 283, 285–291, 293, 295,
<code>\glsabbrvuserfont</code>	322–331		296, 298, 300, 302, 304, 306–310, 312,
<code>\GLSaccessdesc</code>	85		314, 315, 317, 318, 320, 324–326, 328,
<code>\Glsaccessdesc</code>	84, 200, 212		331, 334, 336, 337, 340, 343, 345, 346, 349
<code>\glsaccessdesc</code>	84, 200, 217, 218	<code>\Glsaccessshortpl</code>	
<code>\GLSaccessdescplural</code>	85		.. 90, 235, 241, 248, 251, 254, 258, 260,
<code>\Glsaccessdescplural</code>	85		267, 269, 270, 276, 277, 279, 284, 286,
<code>\glsaccessdescplural</code>	85		287, 293, 295, 296, 302, 304, 306, 308,
<code>\GLSaccessfirst</code>	82		318, 320, 321, 328, 331, 340, 345, 346, 349
<code>\Glsaccessfirst</code>	82	<code>\glsaccessshortpl</code>	90, 226, 235, 236, 241,
<code>\glsaccessfirst</code>	82		245, 246, 248, 250, 251, 254, 256–260,
<code>\GLSaccessfirstplural</code>	84		262, 265, 267, 269–274, 276, 279, 282,
<code>\Glsaccessfirstplural</code>	83		284–291, 293, 295, 296, 298–302, 304,
<code>\glsaccessfirstplural</code>	83		306–310, 312, 314–318, 320, 324, 326,
			328, 331, 334, 337, 340, 343, 345, 346, 349

<code>\GLSaccesssymbol</code>	86	<code>\glsdisplaynumberlist</code>	134, 140
<code>\Glsaccesssymbol</code>	85, 213	<code>\glsdohyperlink</code>	100, 381
<code>\glsaccesssymbol</code>	85, 213, 217, 218	<code>\glsdohypertarget</code>	102, 138
<code>\GLSaccesssymbolplural</code>	86	<code>\glsdoifexists</code>	17, 28, 41, 48, 52, 54–58, 73, 74, 80, 88–93, 102, 114, 133, 140, 141, 156, 157, 228–238, 375
<code>\Glsaccesssymbolplural</code>	86	<code>\glsdoifexistsordo</code>	35, 36, 75
<code>\glsaccesssymbolplural</code>	86	<code>\glsdoifexistsorwarn</code>	17, 199, 200, 212, 213
<code>\GLSaccessstext</code>	81	<code>\glsdoifnoexists</code>	48
<code>\Glsaccessstext</code>	81	<code>\glsdonohyperlink</code>	79, 102, 103
<code>\glsaccessstext</code>	81	<code>\glsdosanitizesort</code>	134
<code>\glsacrshortcutstrue</code>	22, 23	<code>\glsenableentrycount</code>	114, 117, 125
<code>\glsacspacemax</code>	131	<code>\glsenableentryunitcount</code>	116, 126
<code>\glsadd</code>	35, 62, 77, 80, 148	<code>\glsentrycounter</code>	26, 145, 380
<code>\glsadd options</code>		<code>\GlsEntryCounterLabelPrefix</code>	46
<code>theHvalue</code>	10	<code>\glsentrycurrcount</code>	115, 117, 123
<code>thevalue</code>	9, 10	<code>\Glsentrydesc</code>	180, 191, 201
<code>\glsaddpostsetkeys</code>	10, 80	<code>\glsentrydesc</code>	180, 191, 201
<code>\glsaddpresetkeys</code>	10, 80	<code>\Glsentrydescplural</code>	181, 192
<code>\glsaddstoragekey</code>	60, 193, 377, 378	<code>\glsentrydescplural</code>	181, 191, 192
<code>\glsbackslash</code>	65	<code>\Glsentryfirst</code>	120, 173, 174, 178, 190
<code>\glscapscase</code>	74, 81–93, 95, 228–240	<code>\glsentryfirst</code>	120, 173, 178, 190, 368
<code>\glscategory</code>		<code>\Glsentryfirstplural</code>	121, 174, 179, 191
... 72, 81, 96, 104, 105, 194–196, 199–		<code>\glsentryfirstplural</code>	
206, 212, 213, 216, 217, 228–232, 235, 236		... 120, 173, 178, 179, 190, 191, 369	
<code>\glscategorylabel</code>		<code>\glsentryfmt</code>	49–52, 129
... 96, 184, 187, 188, 220, 222–224, 242,		<code>\Glsentryfull</code>	128
245–248, 250, 252–256, 258, 263–268,		<code>\glsentryfull</code>	128
270, 275, 277–286, 288, 292, 294–296,		<code>\Glsentryfullpl</code>	128
298, 299, 301, 303, 305, 307, 309, 311,		<code>\glsentryfullpl</code>	128
312, 316, 318, 319, 321, 323, 324, 326,		<code>\glsentryitem</code>	156, 157, 421–
327, 329–331, 333, 334, 337, 339, 341–349		431, 433, 436, 438, 450, 461, 465, 489, 491	
<code>\glsclsebrace</code>	58, 149, 150	<code>\Glsentrylong</code> ...	105, 106, 120, 174, 183, 192
<code>\glscounter</code>	9, 18	<code>\glsentrylong</code>	105, 106, 120, 173, 182, 183, 192, 253, 255, 278, 280, 295, 297, 319, 321, 327, 329, 344, 370, 371
<code>\glscurrententrylabel</code>	70, 71, 136, 146, 156, 157, 159, 161–163, 215, 216	<code>\Glsentrylongpl</code> ...	105, 106, 121, 183, 192
<code>\glscurrentfieldvalue</code>		<code>\glsentrylongpl</code>	
... 35–40, 42–44, 55, 56, 322, 376, 377		... 105, 106, 120, 183, 192, 193, 370, 371	
<code>\glscustomtext</code> ..	73–75, 89–93, 228–238, 240	<code>\Glsentrylongplural</code>	174
<code>\glsdefaultshortaccess</code>	186, 187	<code>\glsentrylongplural</code>	173
<code>\glsdefaulttype</code>		<code>\Glsentryname</code>	176, 190, 201, 203, 204
... 6, 20, 47, 135, 136, 148, 158, 160, 163		<code>\glsentryname</code>	155, 156, 176, 190, 208, 365, 366, 442–448, 464
<code>\glsdescriptionaccessdisplay</code> ..	180, 200	<code>\glsentrynumberlist</code> ...	134, 141, 447, 448
<code>\glsdescriptionpluralaccessdisplay</code>	181	<code>\glsentrypdfsymbol</code>	212
<code>\glsdescwidth</code>	424–431, 467–469	<code>\Glsentryplural</code>	177, 190
<code>\glsdetoklabel</code>	8, 9, 32, 36–42, 46–49, 52–54, 57, 61, 63– 65, 77, 80, 97, 101, 115, 116, 121–126, 133, 136, 140, 141, 156, 157, 161, 164, 165, 168–170, 199, 202, 203, 207, 443, 444	<code>\glsentryplural</code> ...	177, 178, 190, 367, 368
<code>\glsdisp</code>	387	<code>\glsentryprevcount</code>	115, 117, 123

<code>\glsentryprevmaxcount</code>	124	<code>\glsfirstabbrvscfont</code>	264–280
<code>\glsentryprevtotalcount</code>	123	<code>\glsfirstabbrvsmfont</code>	281–296
<code>\Glsentryshort</code>	104, 105, 181, 192	<code>\glsfirstabbrvuserfont</code> .	323–328, 330–332
<code>\glsentryshort</code>	104, 105, 131, 181, 182, 192, 325, 326, 338, 364, 365, 371	<code>\glsfirstaccessdisplay</code>	178
<code>\Glsentryshortpl</code>	105, 106, 182, 192	<code>\glsfirstlongdefaultfont</code>	245, 248, 256, 258, 261, 264–275, 281– 292, 298, 299, 302, 303, 306–310, 313, 314
<code>\glsentryshortpl</code>	104–106, 182, 192, 364, 365, 371	<code>\glsfirstlongemfont</code>	300, 301, 304, 305, 311, 312, 315, 316
<code>\Glsentrysymbol</code>	179, 191	<code>\glsfirstlongfont</code>	225, 226, 245–250, 253, 256, 258, 260–263, 265, 267, 269, 270, 272, 274, 276, 278, 281, 283, 285, 287, 289, 290, 292, 295, 298, 300, 302, 304, 306, 307, 309, 311, 313, 315, 317, 320, 324, 325, 328, 331, 334, 336, 340, 343, 345, 348
<code>\glsentrysymbol</code>	179, 191, 212, 445–447	<code>\glsfirstlongfootnotefont</code>	250–255, 275–280, 292–297, 317–321
<code>\Glsentrysymbolplural</code>	180, 191	<code>\glsfirstlonghyphenfont</code>	333–345
<code>\glsentrysymbolplural</code>	180, 191	<code>\glsfirstlongonlyfont</code>	348, 349
<code>\Glsentrytext</code>	177, 190	<code>\glsfirstlonguserfont</code>	323–332
<code>\glsentrytext</code>	102, 177, 190, 366, 367	<code>\GLSfirstplural</code>	359, 360
<code>\glsentrytype</code>	156, 225	<code>\Glsfirstplural</code>	360
<code>\Glsentryuseri</code>	86	<code>\glsfirstplural</code>	359, 360
<code>\glsentryuseri</code>	87	<code>\glsfirstpluralaccessdisplay</code> ..	178, 179
<code>\Glsentryuserii</code>	87	<code>\GLSfmtname</code>	56
<code>\glsentryuserii</code>	87	<code>\Glsfmtname</code>	55, 56
<code>\Glsentryuseriii</code>	87	<code>\glsfmtname</code>	55, 56
<code>\glsentryuseriii</code>	87	<code>\GLSfmttext</code>	56
<code>\Glsentryuseriv</code>	87	<code>\Glsfmttext</code>	55, 56
<code>\glsentryuseriv</code>	87	<code>\glsfmttext</code>	55, 56
<code>\Glsentryuserv</code>	88	<code>\glsforeachincategory</code>	241
<code>\glsentryuserv</code>	88	<code>\glsgenentryfmt</code>	72
<code>\Glsentryuservi</code>	88	<code>\glsgetattribute</code> .	78, 101, 117, 121–123, 146, 170, 175, 199–202, 204, 206, 208, 212
<code>\glsentryuservi</code>	88	<code>\glsgetcategoryattribute</code>	194
<code>\glsextrapostnamehook</code>	205	<code>\glsgetgrouptitle</code>	423, 424
<code>\glsfieldfetch</code>	101	<code>\glsgetwidestname</code>	440
<code>\glsfieldxdef</code>	198, 199	<code>\glsgroupheading</code>	165, 422–431, 433, 434, 436–439, 449, 451–457, 462, 470, 471, 473, 474, 476, 477, 479, 480, 482, 484, 485, 487, 488, 492
<code>\glsFindWidestLevelTwo</code>	384	<code>\glsgroupskip</code>	165, 422, 425–432, 434, 436, 438, 450, 462, 470, 472–476, 478, 479, 481, 482, 484, 485, 487, 489, 493
<code>\glsFindWidestTopLevelName</code>	383	<code>\glshasattribute</code>	78, 101, 117, 122, 124, 126, 146, 170, 174, 199–204, 206, 208, 212, 245–247, 249, 250, 252, 253, 255, 257, 259, 260, 262,
<code>\glsfindwidesttoplevelname</code>	442		
<code>\GLSfirst</code>	358, 359		
<code>\Glsfirst</code>	359		
<code>\glsfirst</code>	359		
<code>\glsfirstabbrvdefaultfont</code>	227, 245, 248, 250, 253, 256, 258, 261, 336		
<code>\glsfirstabbrvmfont</code>	298–321		
<code>\glsfirstabbrvfont</code>	130, 225, 226, 245–254, 256–262, 265, 267, 269, 270, 272, 274, 276, 278, 281, 283, 285, 287, 289, 290, 292, 295, 298, 300, 302, 304, 306, 307, 309, 311, 313, 315, 317, 320, 324, 325, 327, 331, 334, 336, 337, 340, 342, 345, 348		
<code>\glsfirstabbrvhyphenfont</code>	333–335, 338, 340, 342–346		
<code>\glsfirstabbrvonlyfont</code>	348, 349		

263, 265–268, 275, 277, 278, 280–284, 292, 294, 295, 297–305, 313, 316, 317, 319–321, 323, 325–327, 329–332, 334– 336, 338, 339, 341–343, 345, 347, 348, 350	\glslinkpresetkeys 10, 78, 170
\glslinkvar 99	
\glslistchildpostlocation 422	
\glslistchildprelocation 422, 423	
\glslistdesc 422, 423	
\glslistdottedwidth 421	
\glslistgroupheaderfmt 423, 424	
\glslistgroupskip 422	
\glslistnavigationitem 423, 424	
\glslistprelocation 422, 423	
\glslocalunset 74, 171	
\glslongaccessdisplay 182, 183	
\glslongdefaultfont . 227, 245, 248, 249, 256, 258, 261, 265, 267, 269, 270, 272, 274, 281, 283, 285, 287, 289–291, 298, 302, 306, 307, 309, 310, 313, 323, 333, 347	
\glslongemfont .. 297, 300, 303, 304, 311, 315	
\glslongextraDescAlign 469–480, 482, 483, 485–487	
\glslongextraDescFmt 466, 470, 471, 473, 474, 476, 477, 479, 481, 482, 484, 485, 487	
\glslongextraDescNameHeader 473	
\glslongextraDescNameTabularFooter 472	
\glslongextraDescNameTabularHeader 472, 473	
\glslongextraDescSymNameHeader 485	
\glslongextraDescSymNameTabularFooter 484, 485	
\glslongextraDescSymNameTabularHeader 484, 485	
\glslongextraGroupHeading 470, 471, 473, 474, 476, 477, 479, 480, 482, 484, 485, 487	
\glslongextraHeaderFmt . 467, 468, 470, 472, 473, 475, 476, 478–481, 483, 484, 486	
\glslongextraLocationAlign 471, 474, 477, 480, 483, 486, 487	
\glslongextraLocationDescNameHeader 474	
\glslongextraLocationDescNameTabularFooter 473, 474	
\glslongextraLocationDescNameTabularHeader 473, 474	
\glslongextraLocationDescSymNameHeader 487	
\glslongextraLocationDescSymNameTabularFooter 486	
\glslongextraLocationDescSymNameTabularHeader 486	
\glslongextraLocationFmt 471, 474, 477, 481, 484, 487	
\glsifcategoryattribute 195	
\glsifcategoryattribute 96, 184, 187, 188, 195, 196, 223, 224	
\glsifnotregular 81	
\glsifnotregularcategory 196	
\glsifplural . 74, 81–86, 89–93, 218, 228–239	
\glsifregular 72, 81, 120, 121, 173, 174	
\glsifregularcategory 196	
\glsifusetranslator 50	
\glsignore 71	
\glsinlinedescformat 432	
\glsinlinesubdescformat 432	
\glsinsert 74, 81, 89–93, 228–240, 332, 339, 341, 345, 347	
\glskeylisttok 127, 128, 222, 225	
\glslabel 8, 9, 35, 36, 52, 72, 75–79, 96, 97, 101, 102, 131, 170, 174, 175, 216–218, 238–240, 253, 255, 278, 280, 295, 297, 319, 321, 325–327, 329, 339, 341, 345, 347	
\glslabeltok 127, 222, 225, 245–247, 249– 253, 255–268, 270, 272, 275–278, 280– 285, 287, 289, 292, 294, 295, 297–307, 309, 311, 313, 315–321, 323–332, 334– 336, 338, 339, 341–343, 345, 347, 348, 350	
\glsletentryfield 209	
\glslink 128, 387	
\glslink options	
counter 10	
format 207	
hyper 350	
hyperoutside 75, 76	
noindex 8, 9, 96, 350	
textformat 78	
theHvalue 78	
thevalue 78, 166	
wrgloss 8, 75	
\glslinkcheckfirsthyperhook 96	
\glslinkpostsetkeys 10, 78, 170	

<code>\glslongextraLocationSymDescNameHeader</code>	<code>\glslongextraSubNameFmt</code> 470, 472–474, 476, 477, 479, 481, 482, 484, 485, 487
..... 484	
<code>\glslongextraLocationSymDescNameTabularFooter</code>	<code>\glslongextraSubSymbolFmt</code> 476, 478, 479, 481, 482, 484, 485, 487
..... 482, 483	
<code>\glslongextraLocationSymDescNameTabularHeader</code>	<code>\glslongextraSymbolAlign</code> 475–483, 485–487
..... 482, 483	
<code>\glslongextraLocSetDescWidth</code> .. 471, 474	<code>\glslongextraSymbolFmt</code> 466, 476, 477, 479, 481, 482, 484, 485, 487
<code>\glslongextraNameAlign</code>	<code>\glslongextraSymDescNameHeader</code> 482
..... 469–480, 482, 483, 485–487	<code>\glslongextraSymDescNameTabularFooter</code>
<code>\glslongextraNameDescHeader</code> 481, 482
..... 470	
<code>\glslongextraNameDescLocationHeader</code> 471	<code>\glslongextraSymDescNameTabularHeader</code>
<code>\glslongextraNameDescLocationTabularFooter</code> 481, 482
..... 470, 471	<code>\glslongextraSymLocSetDescWidth</code> ...
<code>\glslongextraNameDescLocationTabularHeader</code> 477, 480, 483, 486
..... 470, 471	<code>\glslongextraSymSetDescWidth</code>
<code>\glslongextraNameDescSymHeader</code> 476 469, 475, 478, 479, 481, 482, 485
<code>\glslongextraNameDescSymLocationHeader</code>	<code>\glslongextraTabularVAlign</code>
..... 477 469, 471, 472,
<code>\glslongextraNameDescSymLocationTabularFooter</code>	474, 475, 477, 478, 480, 481, 483, 485, 486
..... 476, 477	<code>\glslongextraUpdateWidest</code> 381–383
<code>\glslongextraNameDescSymLocationTabularHeader</code>	<code>\glslongextraUpdateWidestChild</code> 381–384
..... 476, 477	<code>\GlsLongExtraUseTabularfalse</code> 469
<code>\glslongextraNameDescSymTabularFooter</code>	<code>\glslongfont</code>
..... 475 105, 227, 233, 234, 237, 238, 245,
<code>\glslongextraNameDescSymTabularHeader</code>	246, 248, 250, 251, 253, 256–258, 260,
..... 475	261, 263, 265, 267, 269, 270, 272, 274,
<code>\glslongextraNameDescTabularFooter</code>	276, 278, 281, 283, 285, 287, 289, 290,
..... 467, 470	292, 295, 298, 300, 302, 304, 306, 307,
<code>\glslongextraNameDescTabularHeader</code>	309, 311, 313, 315, 317, 320, 324, 325,
..... 467, 470	328, 331, 334, 336, 340, 343, 345, 348, 349
<code>\glslongextraNameFmt</code> ... 470, 471, 473, 474, 476, 477, 479, 481, 482, 484, 485, 487	<code>\glslongfootnotefont</code>
<code>\glslongextraNameSymDescHeader</code> 479	249, 250, 253, 276, 278, 292, 295, 317, 320
<code>\glslongextraNameSymDescLocationHeader</code>	<code>\glslonghyphenfont</code>
..... 480	333, 334, 336, 337, 339, 340, 342, 343, 345
<code>\glslongextraNameSymDescLocationTabularFooter</code>	<code>\glslongonlyfont</code>
..... 479, 480	347, 348
<code>\glslongextraNameSymDescLocationTabularHeader</code>	<code>\glslongpltok</code> 224, 225, 245–247, 249, 250,
..... 479, 480	252, 261, 263, 264, 266, 268, 272, 275,
<code>\glslongextraNameSymDescTabularFooter</code>	277, 281–284, 289, 292, 294, 298–305,
..... 478	309, 311, 315, 317, 319, 323, 324, 326–
<code>\glslongextraNameSymDescTabularHeader</code>	330, 332, 334–337, 339, 341–343, 348, 349
..... 478, 479	<code>\glslongpluralaccessdisplay</code> 183
<code>\glslongextraSetDescWidth</code>	<code>\glslongtok</code> 127, 128, 222, 223,
..... 468–470, 472, 473	225, 245–253, 256, 257, 260, 261, 263–
<code>\glslongextraSubDescFmt</code> 470, 472–474, 476, 478, 479, 481, 482, 484, 485, 487	266, 268, 272, 275, 277, 278, 281–285,
<code>\glslongextraSubLocationFmt</code>	289, 292, 294, 298–305, 309, 311, 315,
..... 472, 474, 478, 481, 484, 487	317–319, 323, 324, 326, 327, 329, 330,
	332, 334–337, 339, 341–343, 345, 348, 349
	<code>\glslonguserfont</code> 323–326, 328, 331
	<code>\glsmcols</code> 452, 454, 455, 457

<code>\GLSname</code>	356	<code>\glssseeformat</code>	54, 57, 63, 133, 141
<code>\Glsname</code>	356	<code>\glssseelist</code>	57
<code>\glsname</code>	356	<code>\glsssetabbrvfmt</code> ...	72, 81, 104, 105, 199– 204, 206, 212, 213, 228–232, 235, 236, 238
<code>\glsnameaccessdisplay</code> ..	176, 201, 202, 204	<code>\glsssetAttribute</code>	245, 247, 249, 250, 252, 253, 255–268, 270, 272, 275–278, 280–285, 287, 289, 292, 294, 295, 297–307, 309, 311, 313, 315–317, 319–321, 323, 325, 327, 329–332, 334– 336, 338, 339, 341–343, 345, 347, 348, 350
<code>\glsnamefont</code>	201, 203, 204, 206, 468	<code>\glsssetcategoryattribute</code>	115, 127, 131, 169, 175, 176, 188, 189, 194, 195, 197, 198, 213
<code>\glsnavhyperlink</code>	143	<code>\glsssetnoexpandfield</code>	14
<code>\glsnavhyperlinkname</code>	100	<code>\glsssettoctitle</code>	136, 160
<code>\glsnavhypertarget</code>	423, 424, 434, 437, 439, 451–457	<code>\glsssetwidest</code>	382, 383
<code>\glsnavigation</code>	423, 424, 434, 437, 439, 451–457	<code>\glssshortaccessdisplay</code>	181, 182
<code>\glsnextpages</code>	136, 160	<code>\glssshortaccsupp</code>	188
<code>\glsnoidxdisplayloc</code>	141, 378	<code>\glssshortpltok</code>	224, 225, 245–247, 249, 250, 252–256, 258, 264, 266, 268, 270, 275, 277, 278, 280–286, 292, 294, 296, 298–305, 307, 317–319, 321, 323, 324, 326, 327, 329, 330, 332, 334, 335, 339, 341–343, 345, 346, 348, 350
<code>\glsnoidxdisplayloclisthandler</code>	140	<code>\glssshortpluralaccessdisplay</code>	182
<code>\glsnoidxloclist</code>	141, 165, 166	<code>\glssshorttok</code>	127, 128, 222, 224, 225, 244–258, 263, 264, 266, 268, 270, 272, 275, 277, 278, 280– 286, 289, 292, 294, 296, 298–305, 307, 309, 311, 317–319, 321, 323, 324, 326– 330, 332–335, 337, 339, 341–343, 345–349
<code>\glsnoidxnumberlistloophandler</code>	141	<code>\glssshowtargetfont</code>	28
<code>\glsnonnextpages</code>	136, 160	<code>\glssshowtargetouter</code>	28
<code>\glsnonnumberlistfalse</code>	70	<code>\glssubentryitem</code>	156, 421–432, 434, 436, 438, 450, 462, 466
<code>\glsnonnumberlisttrue</code>	70	<code>\glssymbolaccessdisplay</code>	179
<code>\glsnopostdotfalse</code>	137	<code>\glssymbolpluralaccessdisplay</code> .	179, 180
<code>\glsnopostdottrue</code>	137	<code>\glstarget</code>	156, 157, 421–432, 434, 436, 438, 450, 461, 462, 465, 466, 489, 491
<code>\glsnumberlistloop</code>	134	<code>\GLStext</code>	357
<code>\glsnumlistlastsep</code>	140, 375	<code>\Glstext</code>	357
<code>\glsnumlistsep</code>	140, 375	<code>\glstext</code>	357
<code>\glsopenbrace</code>	58, 149, 150	<code>\glstextaccessdisplay</code>	177
<code>\glsorder</code>	132, 154	<code>\glstextformat</code>	75, 78, 79
<code>\glspagelistwidth</code> 425, 427, 429, 431, 467, 469		<code>\glstextup</code>	264
<code>\glspar</code>	164, 489	<code>\glstopic@postchildren</code>	492, 493
<code>\glspatchLToutput</code>	470, 471, 473–475, 477, 479, 480, 482, 483, 485, 486	<code>\glstopic@prechildren</code>	488, 492, 493
<code>\glspdfdfmtfull</code>	372	<code>\glstopic@prevlevel</code>	488, 489, 492, 493
<code>\glspdfdfmtfullpl</code>	372, 373	<code>\glstopicAssignSubIndent</code>	489, 493
<code>\glspenaltygroupskip</code> ...	470, 472, 473, 475, 476, 478, 479, 481, 482, 484, 485, 487		
<code>\GLSpl</code>	115, 126, 169		
<code>\Glspl</code>	67, 115, 126, 169		
<code>\glspl</code>	67, 114, 126, 169		
<code>\GLSplural</code>	358		
<code>\Glsplural</code>	358		
<code>\glsplural</code>	358		
<code>\glspluralaccessdisplay</code>	177, 178		
<code>\glspluralsuffix</code>	154, 223, 227		
<code>\glspostdescription</code> ...	19, 20, 137, 215, 421, 422, 424–432, 435–438, 465, 490, 491		
<code>\glspostinline</code>	432		
<code>\glspostlinkhook</code> 73–75, 89–93, 107, 228–238			
<code>\glsprestandardsort</code>	134		
<code>\glsresetentrylist</code>	143, 159, 161		
<code>\glsssee</code>	59–61		

<code>\glstopicAssignWidest</code>	490	<code>\glstreesubitem</code>	433, 462
<code>\glstopicCols</code>	492, 493	<code>\glstreesubsubitem</code>	433, 462
<code>\glstopicColsEnv</code>	492, 493	<code>\glstreesymbol</code>	434, 436
<code>\glstopicDesc</code>	489	<code>\GlsstrLetField</code>	41
<code>\glstopicGroupHeading</code>	488, 492	<code>\glstype</code>	74, 77, 89–93, 170, 228–238
<code>\glstopicInit</code>	488, 492	<code>\glset</code>	62, 74, 118, 119, 171
<code>\glstopicItem</code>	488, 492	<code>\glupdatewidest</code>	381, 382
<code>\glstopicLoc</code>	489	<code>\gluserdescription</code>	323, 324, 327, 330
<code>\glstopicMarker</code>	489	<code>\glswrite</code>	58, 132
<code>\glstopicMidSkip</code>	489	<code>\glswriteentry</code>	8, 9
<code>\glstopicParIndent</code>	488, 492	<code>\Glsxtr</code>	67
<code>\glstopicPostSkip</code>	489	<code>\glxtr</code>	67
<code>\glstopicPreSkip</code>	489	<code>\glxtr@do@wrglossary</code>	8, 10, 13, 16
<code>\glstopicSubIndent</code>	490	<code>\glxtr@addloccllistfield</code>	15, 16
<code>\glstopicSubItem</code>	489, 493	<code>\glxtr@addunused</code>	61
<code>\glstopicSubItemBox</code>	491	<code>\glxtr@applyabbrvfmt</code>	238
<code>\glstopicSubItemSep</code>	491	<code>\glxtr@applyabbrvstyle</code>	223, 242
<code>\glstopicSubLoc</code>	491	<code>\glxtr@beginbookindex</code>	460
<code>\glstopicSubNameFont</code>	491	<code>\glxtr@counterrecord</code>	155
<code>\glstopicSubPreLocSep</code>	491	<code>\glxtr@dblfloat</code>	18
<code>\glstopicTitle</code>	489	<code>\glxtr@do@alsoindex@wrglossary</code>	16
<code>\glstopicTitleFont</code>	489	<code>\glxtr@do@autoadd</code>	9, 77, 78
<code>\glstopicwidest</code>	490, 491	<code>\glxtr@doption</code>	5, 18, 19, 26, 27, 30
<code>\glstreechilddesc</code>	436	<code>\glxtr@endbookindex</code>	461
<code>\glstreeChildDescLoc</code>	434, 436	<code>\glxtr@fields</code>	153
<code>\glstreechildpredesc</code>	436	<code>\glxtr@float</code>	18
<code>\glstreechildprelocation</code>	436, 438	<code>\glxtr@grptitle</code> ...	434, 437–439, 451–457
<code>\glstreechildsymbol</code>	434, 436	<code>\glxtr@headentry@p</code>	108, 109
<code>\glstreedefaultnamefmt</code>	433	<code>\glxtr@hyperoutsidefalse</code>	76
<code>\glstreedesc</code>	435	<code>\glxtr@hyperoutsidettrue</code>	75, 76
<code>\glstreeDescLoc</code>	434, 436, 438, 439	<code>\glxtr@ifnextpunc</code>	219
<code>\glstreegroupheaderfmt</code>		<code>\glxtr@ifpunctoken</code>	219
.....	434, 437–439, 451–457, 459	<code>\glxtr@inc@linkcount</code>	78, 175
<code>\glstreegroupheaderskip</code>		<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 26, 30
.....	434, 435, 437–439, 451–457	<code>\glxtr@indexonly@saveentrycounter</code>	16, 30
<code>\glstreegroupskip</code> ..	433, 434, 436, 438, 450	<code>\glxtr@keylist</code>	66, 67
<code>\glstreeindent</code>	436, 438, 449, 450	<code>\glxtr@label</code>	464
<code>\glstreeitem</code>	433, 453, 461, 462	<code>\glxtr@langtag</code>	153
<code>\glstreenamebox</code>	450	<code>\glxtr@linkprefix</code>	153, 154
<code>\glstreenamefmt</code>		<code>\glxtr@loaddialect</code>	374, 419
.....	432, 434, 436, 438, 440, 442–450	<code>\glxtr@locationhypertext</code>	380
<code>\glstreenavigationfmt</code>	434, 437, 439, 451–457	<code>\glxtr@makeglossaries</code>	132
<code>\glstreeNoDescSymbolPreLocation</code>	435, 436	<code>\glxtr@newabbreviation</code>	130, 222
<code>\glstreenonamechilddesc</code>	438	<code>\glxtr@next</code>	220
<code>\glstreenonamedesc</code>	438	<code>\glxtr@org@@do@wrglossary</code>	30
<code>\glstreenonamesymbol</code>	438	<code>\glxtr@org@dohyperlink</code>	100
<code>\glstreepredesc</code>	435, 437	<code>\glxtr@org@getgrouptitle</code>	142
<code>\glstreePreHeader</code> ..	434, 437–439, 451–457	<code>\glxtr@org@newignoredglossary</code>	49
<code>\glstreeprelocation</code>	433, 435		

\glxtr@orgmakenoidxglossaries	63	\glxtractivatenopost	136
\glxtr@pluralsuffixes	153, 154	\glxtraddallcrossrefs	61
\glxtr@printgloss@checkexists	137	\glxtralias	97
\glxtr@printgloss@groupstrue	139	\glxtrAltTreeIndent	439, 440
\glxtr@process	159, 161, 162	\glxtralttreeInit	449, 455, 457
\glxtr@provideignoredglossary	51	\glxtrAltTreePar	439
\glxtr@punctlist	219	\glxtrAltTreeSetHangIndent	439, 450
\glxtr@record	11, 12, 153	\glxtrAltTreeSetSubHangIndent	450
\glxtr@record@nameref	12, 153	\glxtralttreeSubSymbolDescLocation	450
\glxtr@record@nr	15, 16	\glxtralttreeSymbolDescLocation	440, 450
\glxtr@recordsee	13	\glxtrassignactualsetup	185
\glxtr@renewcommand	376	\glxtrassignfieldfont	81–88
\glxtr@resource	151, 153	\glxtrautoindex	208
\glxtr@s@newignoredglossary	49	\glxtrautoindexassignsort	208
\glxtr@s@provideignoredglossary	51	\glxtrautoindexentry	208
\glxtr@saveentrycounter	8, 10, 13, 97	\glxtrautoindexesc	208
\glxtr@setaccessdisplay	206	\glxtrbibaddress	377
\glxtr@setbookindexmark	463	\glxtrbibauthor	377
\glxtr@setup@record	15, 16, 30, 31	\glxtrbibbooktitle	377
\glxtr@shortcutsval	153, 154	\glxtrbibchapter	377
\glxtr@texencoding	154	\glxtrbibedition	377
\glxtr@undefaction@nr	7	\glxtrbibhowpublished	377
\glxtr@undefaction@val	7	\glxtrbibinstitution	377
\glxtr@usesee	54	\glxtrbibjournal	378
\glxtr@warnonexistsordo	7, 15, 16, 53	\glxtrbibmonth	378
\glxtr@writefields	151	\glxtrbibnote	378
\glxtrabbreviationfont	72	\glxtrbibnumber	378
\glxtrabbrvfootnote	250–253, 255, 275–278, 280, 292–295, 297, 317–319, 321	\glxtrbiborganization	378
\glxtrabbrvpluralsuffix	154, 186, 227, 245, 247, 250, 253, 256, 258, 261, 264, 281, 297, 323, 333, 336, 339, 345, 347	\glxtrbibpages	378
\glxtrabbrvtype	20, 225	\glxtrbibpublisher	378
\glxtrAccSuppAbbrSetFirstLongAttrs	245, 247, 264, 266, 281, 283, 298, 299, 301, 303, 323, 324, 327, 330, 333, 339, 342, 344, 348	\glxtrbibschooll	378
\glxtrAccSuppAbbrSetNameLongAttrs	252, 254, 270, 277, 279, 285, 294, 296, 318, 321	\glxtrbibseries	378
\glxtrAccSuppAbbrSetNameShortAttrs	263, 288, 309, 311, 312, 337	\glxtrbibtitle	378
\glxtrAccSuppAbbrSetNoLongAttrs	250, 252, 256, 258, 268, 275, 278, 286, 292, 294, 305, 307, 316, 319	\glxtrbibtype	378
\glxtrAccSuppAbbrSetTextShortAttrs	246, 248, 265, 267, 282, 284, 299, 301, 303, 305, 326, 329, 331, 334, 341, 343, 346, 349	\glxtrbibvolume	378
		\glxtrbookindexatendgroup	461
		\glxtrbookindexatsubendgroup	462
		\glxtrbookindexatsubsubendgroup	462
		\glxtrbookindexbetween	461
		\glxtrbookindexbookmark	463
		\glxtrbookindexbookmarkprefix	463
		\glxtrbookindexcols	460, 461
		\glxtrbookindexcolspread	460
		\glxtrbookindexfirstmarkfmt	464
		\glxtrbookindexformatheader	463
		\glxtrbookindexgroupskip	463
		\glxtrbookindexlastmarkfmt	464
		\glxtrbookindexlocation	459, 461

\glxstrbookindexmulticolseenv ..	460, 461	\glxstrfieldlistgadd	155
\glxstrbookindexname	458, 461	\glxstrfieldtitlecase	200–203, 206
\glxstrbookindexparentchildsep	459–461	\glxstrfieldtitlecaseecs	199
\glxstrbookindexparentsubchildsep .		\glxstrfieldxifinlist	163
.....	460–462	\glxstrfirstscfont	264
\glxstrbookindexprelocation ...	458, 461	\glxstrfirstsmfont	281
\glxstrbookindexsubbetween	462	\GlsXtrFmtDefaultOptions	35
\glxstrbookindexsublocation	462	\glxstrfmtdisplay	35
\glxstrbookindexsubname	462	\glxstrfmtexternalnameref	380
\glxstrbookindexsubprelocation ...	462	\GlsXtrFmtField	35, 36
\glxstrbookindexsubsubbetween	462	\glxstrfmtinternalnameref	379, 380
\glxstrbookindexthepage	463, 464	\glxstrfootnotedesname	
\glxstrcat	66, 67	252, 254, 277, 280, 294, 296, 318, 321
\glxstrchecknohyperfirst	82–84	\glxstrfootnotedescsort	
\glxstrcombiningdiacriticIIIRules .	391	252, 254, 277, 280, 294, 296, 318, 321
\glxstrcombiningdiacriticIIRules ..	391	\glxstrfootnotename	
\glxstrcombiningdiacriticIRules ...	391	250, 253, 275, 278, 292, 294, 317, 319
\glxstrcombiningdiacriticIVrules ..	391	\GlsXtrForeignTextField	44
\glxstrComputeTreeIndent	450	\GlsXtrFormatLocationList	70, 72, 447, 448
\glxstrComputeTreeSubIndent	450	\GLSxtrfull	21, 93, 362, 363
\glxstrcounterprefix	145	\Glsxtrfull	21, 93, 363
\glxstrcurrencyrules	393	\glxstrfull	20, 21, 93, 362
\glxstrcurrentgrptitle	463	\Glsxtrfullformat	
\GlsXtrDefaultResourceOptions	151	. 226, 240, 242, 243, 245, 248, 251, 254,	
\Glsxtrdefaultsubsequentfmt ...	241, 242	257, 259, 262, 265, 267, 269, 271, 273,	
\glxstrdefaultsubsequentfmt ...	240, 242	275, 276, 279, 282, 284, 286, 287, 290,	
\Glsxtrdefaultsubsequentplfmt .	241, 242	291, 293, 295, 298, 300, 302, 304, 306,	
\glxstrdefaultsubsequentplfmt .	241, 242	308, 310, 312, 314, 316, 317, 320, 324,	
\GlsXtrDefineAbbreviationShortcuts .	23	325, 328, 331, 334, 337, 340, 343, 346, 348	
\GlsXtrDefineAcShortcuts	23	\glxstrfullformat	
\GlsXtrDefineOtherShortcuts	23	. 226, 240, 242, 243, 245, 248, 250, 253,	
\glxstrdetoklocation	169	257, 259, 262, 265, 267, 269, 271, 273,	
\glxstrdiscardperiod	216	275, 276, 279, 282, 283, 286, 287, 290,	
\glxstrdisplayendloc	144	291, 293, 295, 298, 300, 302, 304, 306,	
\glxstrdisplayendlochook	145	308, 310, 312, 314, 316, 317, 320, 324,	
\glxstrdisplaysingleloc	144, 145	325, 328, 331, 334, 337, 340, 343, 345, 348	
\glxstrdisplaystartloc	144	\GLSxtrfullpl	21, 22, 94, 363, 364
\glxstrdoautoindexname	98, 99, 205	\Glsxtrfullpl	21, 93, 364
\glxstrdohyperlink	100, 102, 103	\glxstrfullpl	21, 93, 363
\glxstrdopostpunc		\Glsxtrfullplformat	
....	253, 255, 278, 280, 295, 297, 319, 321	. 226, 240, 242, 243, 246, 248, 251, 254,	
\glxstrdowrglossaryhook	99	257, 259, 262, 265, 267, 269, 271, 273,	
\GlsXtrDualField	377	275, 276, 279, 282, 284, 286, 287, 290,	
\glxxtremsuffix	298, 300, 302,	292, 293, 295, 299, 301, 302, 304, 307,	
304, 306, 307, 309, 311, 313, 315, 317, 320		308, 310, 312, 314, 316, 318, 320, 324,	
\GlsXtrEnableEntryCounting	126	325, 328, 331, 334, 337, 340, 343, 346, 348	
\GlsXtrEnableEntryUnitCounting	115	\glxstrfullplformat	
\GlsXtrEnableOnTheFly	65, 68	240, 242, 243, 245, 248, 250, 253,
\glxxtrendfor	37	257, 259, 262, 265, 267, 269, 271, 273,	

275, 276, 279, 282, 284, 286, 287, 290, 291, 293, 295, 298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 317, 320, 324, 325, 328, 331, 334, 337, 340, 343, 346, 348	\glxtrifinmark 80, 108–111, 351–353
\glxtrifnextpunc 219, 220	\glxtrifnextpunc 219, 220
\glxtrifperiod 216, 218, 219	\glxtrifperiod 216, 218, 219
\glxtrifrecordtrigger 171–173	\glxtrifrecordtrigger 171–173
\GlsXtrIfUnusedOrUndefined 98	\GlsXtrIfUnusedOrUndefined 98
\glxtrifwasfirstuse 81–84, 89–93, 96, 131, 217, 218, 228, 231–238, 253, 255, 278, 280, 295–297, 319–322, 325–327, 329, 339, 341, 345, 347	\glxtrifwasfirstuse 81–84, 89–93, 96, 131, 217, 218, 228, 231–238, 253, 255, 278, 280, 295–297, 319–322, 325–327, 329, 339, 341, 345, 347
\glxtrigenabbrvfmt 72	\glxtrinclinlinkcounter 175
\glxtrigeneralpuncIIrules 393	\glxtrindexaliased 97, 98
\glxtrigeneralpuncIrules 393	\glxtrindexseealso 60
\glxtrigetgroupitle 143, 434, 437–439, 451–457, 463	\glxtrinithyperoutside 77
\glxtrigroupfield 164	\glxtrinitwrgloss 77, 170
\Glsxtrheadfirst 353	\glxtrinitwrglossbeforefalse 75
\glxtrheadfirst 353	\glxtrinitwrglossbeforetrue 75
\Glsxtrheadfirstplural 353	\Glsxtrinlinefullformat 226, 228, 242, 243, 251, 254, 256, 258, 260, 262, 269–271, 273, 274, 277, 279, 285, 287–289, 291, 293, 296, 306, 307, 309, 310, 312, 314, 315, 318, 320, 326, 328, 337, 340, 346, 349
\glxtrheadfirstplural 353	\glxtrinlinefullformat 226, 228, 229, 242, 243, 251, 254, 256, 258, 260, 261, 269– 271, 273, 274, 276, 279, 285, 287–289, 291, 293, 296, 306–308, 310, 312, 313, 315, 318, 320, 325, 328, 336, 340, 346, 349
\Glsxtrheadfull 353	\Glsxtrinlinefullplformat 227, 230, 242, 243, 251, 254, 256, 258, 260, 262, 269, 270, 272–274, 277, 279, 286–288, 290, 291, 293, 296, 306, 308–310, 312, 314, 316, 318, 321, 326, 328, 337, 340, 346, 349
\glxtrheadfull 353	\glxtrinlinefullplformat 226, 229, 230, 242, 243, 251, 254, 256, 258, 260, 262, 269– 271, 273, 274, 276, 279, 285, 287–289, 291, 293, 296, 306–308, 310, 312, 314, 315, 318, 320, 325, 328, 336, 340, 346, 349
\Glsxtrheadfullpl 353	\glxtrinsertinsidefalse 244
\glxtrheadfullpl 353	\GlsXtrInternalLocationHyperlink 26, 146
\Glsxtrheadlong 353	\glxstrLatinA 395–400
\glxtrheadlong 353	\glxstrLatinAELigature 397, 399, 400
\Glsxtrheadlongpl 353	\glxstrLatinE 395–400
\glxtrheadlongpl 353	\glxstrLatinEszettSs 396–398, 400
\Glsxtrheadname 353	\glxstrLatinEszettSz 397, 399
\glxtrheadname 156, 353	\glxstrLatinEth 396–399
\Glsxtrheadplural 353	\glxstrLatinH 395–400
\glxtrheadplural 353	\glxstrLatinI 395–400
\Glsxtrheadshort 353	
\glxtrheadshort 352	
\Glsxtrheadshortpl 353	
\glxtrheadshortpl 352	
\Glsxtrheadtext 353	
\glxtrheadtext 353	
\glxtrhiernamesep 55, 56	
\glxtrhyperlink 26, 101	
\glxtrhyphensuffix 334, 342	
\glxtridentifyglslake 167	
\glxtrifcounttrigger 117–119	
\glxtrifcustomdiscardperiod 216	
\glxtrifemptyglossary . 144, 150, 158, 161	
\GlsXtrIfFieldEqNum 156	
\GlsXtrIfFieldNonZero 376	
\glxtrifhasfield 44, 55, 56, 97, 376, 377, 458	
\glxtrifhyphenstart 333, 335, 338, 342, 344	
\glxtrifindexing 98	

<code>\glxstrLatinInsularG</code>	399	<code>\glxstrMathItalicPhi</code> ...	405, 406, 409, 410
<code>\glxstrLatinK</code>	395–400	<code>\glxstrMathItalicPi</code>	405, 409, 410
<code>\glxstrLatinL</code>	395–400	<code>\glxstrMathItalicPsi</code> ...	405, 406, 409, 410
<code>\glxstrLatinM</code>	395–400	<code>\glxstrMathItalicRho</code>	405, 409, 410
<code>\glxstrLatinN</code>	395–400	<code>\glxstrMathItalicSigma</code>	405, 409, 410
<code>\glxstrLatinO</code>	395–400	<code>\glxstrMathItalicTau</code>	405, 409, 410
<code>\glxstrLatinOELigature</code>	397, 400	<code>\glxstrMathItalicTheta</code>	405, 409, 410
<code>\glxstrLatinP</code>	395–400	<code>\glxstrMathItalicUpsilon</code> ...	405, 409, 410
<code>\glxstrLatinS</code>	395–400	<code>\glxstrMathItalicXi</code>	405, 409, 410
<code>\glxstrLatinT</code>	395–400	<code>\glxstrMathItalicZeta</code>	405, 409, 410
<code>\glxstrLatinThorn</code>	400	<code>\glxstrmultisupplocation</code>	378
<code>\glxstrLatinX</code>	396–400	<code>\glxstrnamerefink</code>	379
<code>\GlsXtrLocationField</code>	165	<code>\glxstrnewabbrevpresetkeyhook</code>	224
<code>\glxstrlocationhyperlink</code>	145	<code>\glxstrnewnumber</code>	22
<code>\glxstrlocrangefmt</code>	144, 145	<code>\glxstrnewsymbol</code>	22
<code>\GLSxtrlong</code>	21, 91, 360, 361	<code>\glxstrNoGlossaryWarning</code>	15, 24, 146
<code>\Glsxtrlong</code>	21, 91, 361	<code>\GlsXtrNoGlsWarningAutoMake</code>	150
<code>\glxstrlong</code>	20, 21, 91, 360	<code>\GlsXtrNoGlsWarningBuildInfo</code>	150
<code>\glxstrlonghyphen</code>	340	<code>\GlsXtrNoGlsWarningCheckFile</code>	150
<code>\glxstrlonghyphennoshort</code>	336, 337	<code>\GlsXtrNoGlsWarningEmptyMain</code>	150
<code>\glxstrlonghyphenshort</code>	334	<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	150
<code>\glxstrlongnoshortdescname</code> .	261, 314, 336	<code>\GlsXtrNoGlsWarningEmptyStart</code>	150
<code>\glxstrlongnoshortname</code>		<code>\GlsXtrNoGlsWarningHead</code>	150
.....	263, 272, 289, 309, 311, 337	<code>\GlsXtrNoGlsWarningMisMatch</code>	150
<code>\GLSxtrlongpl</code>	21, 92, 361, 362	<code>\GlsXtrNoGlsWarningNoOut</code>	151
<code>\Glsxtrlongpl</code>	21, 92, 362	<code>\GlsXtrNoGlsWarningTail</code>	151
<code>\glxstrlongpl</code>	20, 21, 92, 361	<code>\glxstrnopostpunc</code>	137
<code>\glxstrlongshortdescname</code>		<code>\glxstronlydescname</code>	349
.....	246, 266, 282, 299, 301, 335, 341	<code>\glxstronlydescsort</code>	349
<code>\glxstrlongshortdescsort</code>		<code>\glxstronlyname</code>	348
....	246, 266, 282, 299, 301, 329, 335, 341	<code>\glxstronlysuffix</code>	348
<code>\glxstrlongshortname</code>		<code>\glxstrorg@ifKV@glslink@hyper</code>	73
.....	245, 264, 281, 298, 300, 323, 324, 333, 339	<code>\glxstrorglong</code>	185, 186, 222, 246, 336
<code>\glxstrlongshortuserdescname</code> ..	326, 329	<code>\glxstrorgshort</code>	185, 186, 222, 246
<code>\glxstrmarkhook</code>	350, 351	<code>\GLSxtrp</code>	108
<code>\glxstrMathItalicAlpha</code>	405, 408, 409	<code>\Glsxtrp</code>	107
<code>\glxstrMathItalicBeta</code>	405, 408, 409	<code>\glxstrp</code>	107, 109
<code>\glxstrMathItalicChi</code> ...	405, 406, 409, 410	<code>\glxstrparen</code>	217,
<code>\glxstrMathItalicDelta</code>	405, 408, 410	218, 225, 226, 245–249, 251, 254, 256–
<code>\glxstrMathItalicEpsilon</code> ...	405, 408, 410	258, 260–262, 264–274, 276, 277, 279,
<code>\glxstrMathItalicEta</code>	405, 409, 410	281–291, 293, 296, 298–310, 312, 314–
<code>\glxstrMathItalicGamma</code>	405, 408, 409	316, 318, 320–322, 333–338, 342–344, 349
<code>\glxstrMathItalicIota</code>	405, 409, 410	<code>\glxstrpdfentryfmt</code>	35
<code>\glxstrMathItalicKappa</code>	405, 409, 410	<code>\Glsxtrpl</code>	67
<code>\glxstrMathItalicLambda</code>	405, 409, 410	<code>\glxstrpl</code>	67
<code>\glxstrMathItalicMu</code>	405, 409, 410	<code>\glxstrpostdescription</code> .	137, 197, 215, 432
<code>\glxstrMathItalicNu</code>	405, 409, 410	<code>\glxstrposthyphenlong</code>	345, 347
<code>\glxstrMathItalicOmega</code> .	405, 406, 409, 410	<code>\glxstrposthyphenshort</code>	339, 341
<code>\glxstrMathItalicOmicron</code> ...	405, 409, 410		

<code>\glsxtrposthyphensubsequent</code>	<code>\glsxtrshorthyphenlong</code>	343
..... 339, 341, 345, 347	<code>\glsxtrshortlongdescname</code>	
<code>\glsxtrpostlink</code> 248, 268, 284, 303, 305, 343, 346	
216	<code>\glsxtrshortlongdescsort</code>	
<code>\glsxtrpostlinkendsentence</code> 248, 268, 284, 303, 305, 331, 343, 346	
216	<code>\glsxtrshortlongname</code>	
<code>\glsxtrpostlinkhook</code>	247, 266, 283, 301, 303, 327, 330, 342, 344	
216	<code>\glsxtrshortlonguserdescname</code> ..	329, 331
<code>\glsxtrpostlocalreset</code>	<code>\glsxtrshortnolongname</code> ..	256, 268, 285, 305
114, 116, 124	<code>\GLSxtrshortpl</code>	21, 90, 354–356
<code>\glsxtrpostlocalunset</code>	<code>\Glsxtrshortpl</code>	21, 90, 355
113, 116, 124	<code>\glsxtrshortpl</code>	20, 21, 90, 354
<code>\glsxtrpostlongdescription</code>	<code>\glsxtrsmfont</code>	280
49	<code>\glsxtrsmsuffix</code>	
<code>\glsxtrpostnamehook</code> 281, 283, 285, 287, 289, 290, 292, 295	
202–204, 207	<code>\GlsXtrStandaloneEntryName</code>	156
<code>\GlsXtrPostNewAbbreviation</code> ..	<code>\GlsXtrStandaloneEntryOther</code>	157
225, 242,	<code>\GlsXtrStandaloneGlossaryType</code> ..	156, 157
243, 245–247, 249, 250, 252, 253, 255–	<code>\GlsXtrStandaloneSubEntryItem</code> ..	156, 157
263, 265–268, 270, 272, 275, 277, 278,	<code>\Glsxtrsubsequentfmt</code>	
280–286, 289, 292, 294–296, 298–305, 239, 242, 261, 272, 274,	
307, 309, 311, 313, 315–317, 319, 321,	289, 291, 310, 311, 313, 315, 336, 340, 345	
323, 324, 326, 327, 329, 330, 332, 334–	<code>\glsxtrsubsequentfmt</code>	
336, 338, 339, 341–343, 345, 347, 348, 350 239, 242, 261, 272, 274,	
<code>\glsxtrpostreset</code>	289, 290, 309, 311, 313, 315, 336, 340, 345	
113, 116, 124	<code>\Glsxtrsubsequentplfmt</code>	
<code>\glsxtrpostunset</code> 239, 242, 261, 272, 274,	
112, 116, 124	289, 291, 310, 311, 313, 315, 336, 340, 345	
<code>\glsxtrprelocation</code>	<code>\glsxtrsubsequentplfmt</code>	
..... 422, 424–426, 428, 430, 433, 458 239, 242, 261, 272, 274,	
<code>\glsxtrprotectlinks</code>	289, 291, 309, 311, 313, 315, 336, 340, 345	
101, 103	<code>\glsxtrsupplocationurl</code>	146, 378, 380
<code>\glsxtrprovideaccsuppcmd</code>	<code>\glsxtrtagfont</code>	215
188, 189	<code>\GLSxtrtitlefirst</code>	368
<code>\GlsXtrRecordCounter</code>	<code>\Glsxtrtitlefirst</code>	352, 353, 368, 369
13	<code>\glsxtrtitlefirst</code>	352, 353, 368
<code>\glsxtrrecordtriggervalue</code>	<code>\GLSxtrtitlefirstplural</code>	369
170	<code>\Glsxtrtitlefirstplural</code>	352, 353, 369
<code>\GlsXtrRecordWarning</code>	<code>\glsxtrtitlefirstplural</code>	352, 353, 369
151	<code>\GLSxtrtitlefull</code>	372
<code>\glsxtrregularfont</code>	<code>\Glsxtrtitlefull</code>	352–354, 372
72, 81	<code>\glsxtrtitlefull</code>	352, 353, 371, 372
<code>\glsxtrresourcecount</code>	<code>\GLSxtrtitlefullpl</code>	373
152	<code>\Glsxtrtitlefullpl</code>	352–354, 373
<code>\glsxtrresourcefile</code>	<code>\glsxtrtitlefullpl</code>	352–354, 372
152	<code>\GLSxtrtitlelong</code>	370
<code>\glsxtrresourceinit</code>	<code>\Glsxtrtitlelong</code>	352, 353, 370
151	<code>\glsxtrtitlelong</code>	352, 353, 370
<code>\glsxtrrestoremarkhook</code>	<code>\GLSxtrtitlelongpl</code>	371
351		
<code>\glsxtrrestorepostpunc</code>		
137		
<code>\glsxtrscfont</code>		
264		
<code>\glsxtrscsuffix</code>		
.... 265, 267, 268, 270, 272, 274, 276, 278		
<code>\GlsXtrSetActualChar</code>		
211		
<code>\glsxtrsetaliasnoindex</code>		
15, 16, 97, 98		
<code>\GlsXtrSetEncapChar</code>		
211		
<code>\GlsXtrSetEscChar</code>		
211		
<code>\glsxtrsetfieldifexists</code>		
41, 42		
<code>\glsxtrsetglossarylabel</code>		
138		
<code>\GlsXtrSetLevelChar</code>		
211		
<code>\glsxtrsetpopts</code>		
106		
<code>\glsxtrsetupfulldefs</code>		
228–		
230, 253, 255, 278, 280, 295, 297, 320, 322		
<code>\GLSxtrshort</code>		
21, 89, 110, 111, 354, 355		
<code>\Glsxtrshort</code>		
21, 89, 355		
<code>\glsxtrshort</code>		
20, 21, 88, 354		
<code>\glsxtrshortdescname</code> ...		
258, 270, 286, 307		
<code>\glsxtrshorthyphen</code>		
345, 346		

<code>\Glsxtrtitlelongpl</code>	352, 353, 371	<code>\GlsXtrUseAbbrStyleFmts</code>	
<code>\glxtrtitlelongpl</code>	352, 353, 370, 371	247, 249, 252, 255, 257, 259,
<code>\GLSxtrtitlename</code>	366		260, 263, 264, 266, 268, 271, 277, 280,
<code>\Glsxtrtitlename</code>	352, 353, 366		283, 285, 288, 294, 297, 299, 301, 303,
<code>\glxtrtitlename</code>	352, 353, 365		305, 308, 313, 316, 319, 322, 327, 329,
<code>\glxtrtitleorpdforheading</code>			330, 332, 335, 336, 338, 341, 344, 347, 350
.....	28, 155–157, 352, 353	<code>\GlsXtrUseAbbrStyleSetup</code>	257, 259, 260,
<code>\GLSxtrtitleplural</code>	368		262, 263, 271, 274, 288, 290, 308, 313, 316
<code>\Glsxtrtitleplural</code>	352, 353, 367	<code>\glxtrusefield</code>	375
<code>\glxtrtitleplural</code>	352, 353, 367	<code>\glxtruserfield</code>	322
<code>\Glsxtrtitleshort</code>	352, 353, 365	<code>\glxtruserparen</code>	323–332
<code>\glxtrtitleshort</code>	352, 353, 364	<code>\glxtrusersuffix</code>	324, 325, 327, 331
<code>\Glsxtrtitleshortpl</code>	352, 353, 365	<code>\glxtruseseealsoformat</code>	57, 58
<code>\glxtrtitleshortpl</code>	352, 353, 364, 365	<code>\glxtruseseeformat</code>	54, 57
<code>\GLSxtrtitletext</code>	367	<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	223, 243
<code>\Glsxtrtitletext</code>	352, 353, 366, 367	<code>\GlsXtrWarning</code>	66, 67
<code>\glxtrtitletext</code>	352, 353, 366	<code>\glxtrword</code>	222
<code>\GlsXtrTotalRecordCount</code>	169	<code>\glxtrwordsep</code> ..	222, 333, 335, 338, 342, 344
<code>\glxtrtreetopindent</code>	440, 449	<code>\glxtrwrglossmark</code>	27
<code>\glxtrundefaction</code>	7, 15, 16, 33, 49, 50, 52, 53		
<code>\glxtrundeftag</code>	31, 140, 141		
<code>\GlsXtrUnknownDialectWarning</code>	45		
<code>\glxtrunsrtdo</code>	163		
<code>\glxtrUpAlpha</code>	403, 404, 408, 409		
<code>\glxtrUpBeta</code>	403, 404, 408, 409		
<code>\glxtrUpChi</code>	404, 409, 410		
<code>\glxtrUpDelta</code>	403, 404, 408, 410		
<code>\glxtrUpDigamma</code>	403, 405, 409		
<code>\glxtrUpEpsilon</code>	403, 404, 409, 410		
<code>\glxtrUpEta</code>	403, 404, 409, 410		
<code>\glxtrUpGamma</code>	403, 404, 408, 410		
<code>\glxtrUpIota</code>	404, 409, 410		
<code>\glxtrUpKappa</code>	404, 409, 410		
<code>\glxtrUpLambda</code>	404, 409, 410		
<code>\glxtrUpMu</code>	404, 409, 410		
<code>\glxtrUpNu</code>	404, 409, 410		
<code>\glxtrUpOmega</code>	404, 409, 410		
<code>\glxtrUpOmicron</code>	404, 409, 410		
<code>\glxtrUpPhi</code>	404, 409, 410		
<code>\glxtrUpPi</code>	404, 409, 410		
<code>\glxtrUpPsi</code>	404, 409, 410		
<code>\glxtrUpRho</code>	404, 409, 410		
<code>\glxtrUpSigma</code>	404, 409, 410		
<code>\glxtrUpTau</code>	404, 409, 410		
<code>\glxtrUpTheta</code>	403, 404, 409, 410		
<code>\glxtrUpUpsilon</code>	404, 409, 410		
<code>\glxtrUpXi</code>	404, 409, 410		
<code>\glxtrUpZeta</code>	403, 404, 409, 410		

H

<code>\hangindent</code>	436,
	438, 439, 449, 451, 456, 457, 488, 490, 492
<code>\hbox</code>	421
<code>\hfill</code>	421
<code>\href</code>	101
<code>\hsize</code>	69
<code>\hss</code>	421
<code>\hyperlink</code>	16, 102, 378
<code>\hyperpage</code>	207
<code>\hyperref</code>	101, 146, 376, 381
<code>hyperref package</code>	102, 207, 350, 364, 378

I

<code>\if</code>	65
<code>\if@display</code>	12
<code>\if@glxtr@autoseeindex</code>	29, 54, 59
<code>\if@glxtr@equations</code>	9, 78
<code>\if@glxtr@floats</code>	18
<code>\if@glxtr@format@override</code>	208
<code>\if@glxtr@docdefrestricted</code>	63
<code>\if@glxtr@indexcrossrefs</code>	17, 61
<code>\ifblank</code>	33, 66, 67, 132
<code>\ifbool</code>	32, 48
<code>\ifcase</code> ..	7, 16, 22, 24, 27, 64, 75, 138, 434, 461
<code>\ifcsdef</code>	33, 41, 47, 49–52, 76, 78,
	94, 95, 107–111, 121, 136, 142, 143, 156,
	164, 167–169, 174, 199–202, 204–206,
	212, 217, 223, 238, 242, 379, 424–431, 490
<code>\ifcsstring</code>	33, 195, 241

<code>\ifcsundef</code>	32, 40, 45, 47, 49–52, 64, 68, 70, 103, 115, 116, 121–125, 142, 143, 146, 163, 175, 176, 188, 195, 241, 243, 244, 421, 440, 441, 449, 463, 490	<code>\ifglssused</code>	61, 62, 96, 117, 126, 131, 238, 442, 443, 445, 446, 448
<code>\ifcsvoid</code>	60, 194	<code>\ifglsxindy</code>	146, 148
<code>\ifdef</code>	15, 22, 28, 29, 32, 35, 40, 41, 44, 46, 47, 53, 58, 59, 62, 68, 69, 96, 97, 100, 102, 108–110, 132, 135, 136, 140, 141, 152–154, 157, 167, 185, 197, 198, 211, 212, 215, 322, 352, 364–373, 376, 381–384, 419, 421–424, 432–435, 437–439, 451–457, 459, 463, 464	<code>\ifglsxtr@hyperoutside</code>	79
<code>\ifdefempty</code>	7–9, 38, 40, 44, 45, 49–51, 54, 57, 78, 80, 115, 127, 129, 132, 135, 145, 151, 159, 161, 165, 169, 170, 186–188, 213, 222, 238, 384, 460	<code>\ifglsxtr@printgloss@groups</code> ...	159, 162
<code>\ifdefequal</code>	43, 44, 63, 100, 127, 129, 150, 164, 166, 205	<code>\ifglsxtr@trinitwrglossbefore</code> ...	75, 79, 170
<code>\ifdefstring</code> .	6, 26, 42, 43, 154, 208, 214, 459	<code>\ifglsxtrininsertinside</code> 231–238, 240, 241, 245, 246, 248, 250, 251, 253, 254, 256–262, 265, 267, 269–277, 279, 282–293, 295, 296, 298–302, 304, 306–318, 320, 321, 324–326, 328, 331, 333, 335, 338–342, 344, 346, 348, 349
<code>\ifdefvoid</code>	54, 59–61, 101, 121, 142, 146, 165, 166, 380	<code>\ifHy@hyperindex</code>	207
<code>\ifdim</code>	69, 131, 440–448, 468, 491	<code>\ifinlist</code>	113
<code>\IfFileExists</code>	25, 146, 150, 151, 154, 419, 420	<code>\ifinlistcs</code>	37, 64
<code>\ifglossaryexists</code>	53, 160, 374, 375	<code>\ifinner</code>	28
<code>\ifglssacronym</code>	20, 150, 374	<code>\ifKV@glslink@hyper</code>	73, 77, 79
<code>\ifglssacrshortcuts</code>	22	<code>\ifKV@glslink@local</code>	74, 171
<code>\ifglssautomake</code>	135, 150, 154	<code>\ifKV@glslink@noindex</code> ..	8, 9, 13, 35, 97, 98
<code>\ifglssentrycounter</code>	46	<code>\ifmmode</code>	12, 28
<code>\ifglssentryexists</code>	9, 32, 52, 53, 66, 67, 70, 81, 165, 195, 215, 216, 385	<code>\ifnum</code>	17, 40, 62, 63, 117, 125, 126, 142, 152, 165, 170, 381–383, 436, 438, 449, 450, 460, 461, 488, 492, 493
<code>\ifglssfieldeq</code>	193	<code>\ifst@rred</code>	12
<code>\ifglsshdesc</code> ...	435, 436, 488, 489, 491, 492	<code>\ifstreempty</code>	156, 167, 176, 379, 381–385
<code>\ifglsshfield</code>	35, 36, 322	<code>\ifstrequal</code>	19, 24, 77, 100, 379
<code>\ifglsshlong</code>	120, 121, 173, 174	<code>\ifthenelse</code>	150, 217
<code>\ifglsshparent</code> .	156, 157, 159, 162, 442–444	<code>\IfTrackedDialectHasMapping</code>	45
<code>\ifglsshasshort</code>	55, 56, 72, 81	<code>\IfTrackedLanguageFileExists</code>	374
<code>\ifglsshassymbol</code>	217, 218, 435–437, 439, 489, 491	<code>\ifundef</code>	16, 26, 38–40, 44, 132, 214, 215, 378, 419, 468
<code>\ifglssindexonlyfirst</code>	98	<code>\ifx</code> ...	8, 10–12, 15, 31, 58, 68, 70, 78, 129, 132, 136, 139, 144, 145, 152–154, 160, 208, 209, 211, 220–222, 224, 332, 380, 457
<code>\ifGlsLongExtraUseTabular</code>	469, 471, 472, 474, 475, 477, 478, 480, 481, 483, 484, 486	<code>\immediate</code>	62, 117, 125, 146, 147, 154
<code>\ifglssnogroupskip</code>	422, 425–432, 434, 436, 438, 450, 459, 470, 472–474, 476, 478, 479, 481, 482, 484, 485, 487	<code>\index</code>	208
<code>\ifglssnonumberlist</code>	72	<code>\indexspace</code>	422, 433, 459, 463
<code>\ifglssnopostdot</code>	19, 137	<code>\input</code>	373
<code>\ifglsssanitizesort</code>	134	<code>\inputencodingname</code>	154
<code>\ifglsssubentrycounter</code>	46	<code>\InputIfFileExists</code>	62
		<code>\istfilename</code>	132
		<code>\item</code>	148, 149, 421–424, 433, 434, 452, 453
			J
		<code>\jobname</code>	62, 146, 148–152, 154
			K
		<code>\key@ifundefined</code>	13, 14, 33, 34, 94, 158, 161, 164
		<code>\KV@glslink@hyperfalse</code> ...	82, 96, 102, 103

<code>\KV@glslink@hypertrue</code>	102	<code>\makenoidxglossaries</code>	149
<code>\KV@glslink@noindexfalse</code>	80, 96, 97	<code>\MakeTextUppercase</code>	352
<code>\KV@glslink@noindextrue</code>	97, 103	<code>\MakeUppercase</code>	352, 353
L			
<code>\L</code>	400, 403	<code>\marginpar</code>	28
<code>\l</code>	403	<code>\markboth</code>	351
<code>\label</code>	26, 138	<code>\markright</code>	351
<code>\large</code>	489	<code>\mathit</code>	388
<code>\LaTeX</code>	148, 149	<code>\mathrm</code>	387–389
<code>\leaders</code>	421	<code>\maxdimen</code>	69
<code>\leavevmode</code>	49, 77	<code>\mbox</code>	423, 424, 449
<code>\let</code>	5, 7–11, 13, 15, 16, 18, 20–22, 28–30, 35, 37, 43– 45, 48, 49, 62–64, 68, 70, 71, 73–75, 77– 93, 95–100, 102, 103, 107, 112, 113, 115, 116, 124–133, 135–143, 151, 152, 154, 158–162, 165, 166, 170, 175, 185–187, 200, 201, 203–210, 213–215, 219–224, 228–238, 240–242, 253, 255, 278, 280, 295, 297, 320, 322, 350–354, 375, 376, 379, 380, 419, 433, 439, 442, 453, 460–463	<code>\MessageBreak</code>	64, 68, 117, 126, 132, 134, 136, 151, 160, 241
<code>\letabbreviationstyle</code>	251, 252, 254, 255, 257, 259, 262, 263, 270, 271, 286, 288, 307, 308	<code>mfirstuc package</code>	214
<code>\letcs</code>	33, 38–40, 54, 57, 61, 76, 78, 94, 140–142, 164, 165, 199–207, 212, 464	<code>\mfirstucMakeUppercase</code>	41, 81– 90, 92, 93, 95, 104–106, 108, 110, 111, 120, 128, 174, 176–183, 190–193, 202, 203, 207, 229, 230, 232, 234, 236, 238–240
<code>\levelchar</code>	211	<code>\mfu@checkword@arg</code>	214
<code>\linewidth</code>	468	<code>\mfu@checkword@do</code>	214
<code>\listadd</code>	121	<code>\midrule</code>	467, 470, 472, 473, 475, 476, 478, 480, 481, 483, 484, 486
<code>\listbreak</code>	214	N	
<code>\listcsadd</code>	36	<code>\NeedsTeXFormat</code> ...	5, 374, 420, 458, 465, 488
<code>\listcseadd</code>	36, 122	<code>\new@atom@glossaryentry</code>	62
<code>\listcsgadd</code>	36, 64	<code>\new@command</code>	376
<code>\listcsxadd</code>	36, 122	<code>\new@glossaryentry</code>	64, 134
<code>\listxadd</code>	112, 113	<code>\new@ifnextchar</code>	34, 94, 95, 119, 120, 167, 171–173, 219, 227–238, 386, 387
<code>\loadglsentries</code>	64, 148	<code>\newabbr</code>	21, 22
<code>\long</code>	48, 49	<code>\newabbreviation</code>	21, 22
M		<code>\newabbreviationhook</code>	225
<code>\m@ne</code>	376	<code>\newabbreviationstyle</code> 244, 246–249, 252, 254, 255, 257–268, 270–273, 275, 277–279, 281–286, 288, 290, 292, 294, 296, 297, 299, 301, 303, 305, 307–309, 311–314, 316, 318, 319, 321, 323, 324, 326, 327, 329–331, 333, 334, 336, 337, 339, 341–344, 346, 348, 349
<code>\MakeAcronymsAbbreviations</code>	130	<code>\newacronym</code>	127, 130
<code>\makeatletter</code>	62, 146, 152, 210	<code>\newacronymhook</code>	127
<code>\makeatother</code>	210	<code>\newacronymstyle</code>	128, 130
<code>\makebox</code>	421, 450, 491	<code>\newcommand</code>	5–15, 17–57, 59– 62, 65–68, 70–72, 75–77, 79–82, 88, 94, 95, 97–103, 106, 108–115, 117, 119–127, 129–131, 136–142, 144–153, 155–159, 161–186, 188–199, 205–222, 225–238, 240–244, 246–249, 251, 255, 257, 260,
<code>\makefirstuc</code>	214		
<code>makeglossaries</code>	139		
<code>\makeglossaries</code>	131, 147, 149, 150, 155		
<code>makeindex</code>	494		
<code>makeindex</code>	16, 131, 494		

263, 264, 280, 281, 297, 322, 323, 326, 328, 333, 335, 338, 339, 342, 344, 347, 349, 351–381, 383–387, 390–418, 420, 422, 432, 433, 435–441, 449, 458–460, 463–473, 475, 476, 478–484, 486, 489–492	171–173, 206, 212, 213, 215, 227–238, 332, 351, 352, 354–364, 385–387, 442–448
\newcount 139, 152, 166	\newsym 22
\newcounter 26, 174	\newterm 197
\newentry 22	\newtoks 221
\newenvironment 160	\newwrite 62, 132
\newglossary 20, 132	\nfss@text 28
\newglossaryentry 22, 62, 64, 115, 123, 127, 197, 198, 225	\nobreak 422–424, 433, 489
\newglossaryentry options	\NoCaseChange 108–111, 157, 354–364
access 187	\noexpand ... 11, 13, 25, 57, 60–62, 77, 127, 128, 146, 147, 152, 159, 162, 163, 165, 175, 185, 186, 209, 210, 225, 376, 385, 420, 460–462, 469–483, 485, 486, 492, 493
alias 18, 54, 57, 59–61	\nofiles 150
desc 180, 191	\noindent ... 150, 437–439, 452–455, 457, 489
descplural 181, 191, 192	\nopagebreak 433, 459, 463, 488, 493
description 515	\nopostdesc 49, 66, 67, 137, 160, 197
first ... 101, 178, 190, 244, 358–360, 368, 494	\np@glstr@assign@leveloffset 139
firstaccess 188	\ns@GLSxtrfull 229
firstplural 178, 179, 190, 191, 244, 359, 360, 369, 494	\ns@GLSxtrfull 228
group 164, 165	\ns@glstrfull 227
loclist 36	\ns@GLSxtrfullpl 230
long 183, 192, 370	\ns@GLSxtrfullpl 230
longplural 183, 193, 370	\ns@glstrfullpl 229
name 55, 176, 190, 208, 356, 365	\ns@GLSxtrlong 234
plural 177, 190, 244, 357, 358, 367	\ns@glstrlong 233
see 17, 18, 29, 54, 59, 61, 64, 133	\ns@GLSxtrlong 233
seealso 18, 54, 57, 59–61, 505	\ns@GLSxtrlongpl 238
short 55, 182, 192, 221	\ns@glstrlongpl 237
shortaccess 186	\ns@GLSxtrlongpl 236
shortaccessplural 186	\ns@GLSxtrshort 232
shortplural 182, 192, 221	\ns@glstrshort 231
symbol 179, 191	\ns@GLSxtrshort 231
symbolplural 179, 180, 191	\ns@GLSxtrshortpl 236
text 55, 101, 177, 190, 244, 246, 357, 366	\ns@glstrshortpl 235
textaccess 187	\ns@GLSxtrshortpl 234
user2 46	\null 24
\newglossarystyle 460, 469, 471, 472, 474, 475, 477, 478, 480, 481, 483, 484, 486, 488, 492	\number 62, 122–125, 152, 165, 167
\newif 75, 207, 244, 469	\numexpr 122, 125, 165
\newlength 439, 440, 490	
\newnum 22	
\newrobustcmd .. 34, 35, 37, 38, 41–44, 57– 59, 63, 76, 80, 94, 95, 106–108, 119, 120, 137, 142, 156, 157, 163, 164, 167, 168,	

O

\O 400, 403
\o 403
\openout 62
\or 7, 16, 22, 23, 27, 64, 75, 434, 462
\org@glossaryentrynumbers 69, 70, 136, 160
\org@glossarytitle 136, 160
\org@ifKV@glslink@hyper 77, 79

P	
\p@glshyp@opt	99
\p@glstr@assign@leveloffset	139
package options:	
abbreviations	20
accsupp	24, 176
acronym	20
automake	135, 148, 154
immediate	154
true	154
autoseeindex	29
false	29
counter	
wrglossary	26
debug	
showtargets	102
docdef	16, 17, 63, 64, 115, 123
atom	62
false	63, 64
restricted	17
true	62, 64
docdefs	
restricted	63
equations	9, 78
indexcounter	376
nonumberlist	69
nopostdot	19
false	19
numbers	22
order	
letter	154
postdot	19
prefix	24
record	7, 14, 15, 63, 73, 131, 151, 228–238, 503
alsoindex	8, 10, 154
nameref	12, 30, 31, 152, 155
only	8, 149
shortcuts	22
ac	22
all	22
false	22
none	22
true	22
sort	
use	80
style	25
stylemods	25
symbols	22, 198
undefaction	52, 53
error	6
warn	6
xindy	58
\PackageError	6, 13, 25, 29, 63, 64, 67, 68, 76, 94, 95, 98, 100, 107, 108, 115, 116, 123, 125, 126, 130, 132, 134, 143, 155, 162, 164, 167, 217, 241–244, 421
\PackageWarning	18
\PackageWarningNoLine	18
\pagelistname	470, 473, 476, 480, 483, 486
\pageref	26, 46
\par	149–151, 423, 424, 433, 436–439, 449, 451, 453–457, 459, 488, 490, 492
\parindent	433, 436, 438–440, 450–452, 454–457, 460, 488, 490, 492
\parskip	433, 436, 438, 452, 454, 455, 460
\PassOptionsToPackage	5, 24
\pdfbookmark	459, 460
\pdfstringdef	185
\pp@glstr@assign@leveloffset	139
\preglossarypreamble	47
\preto	97, 385
\print@noop@unsrtglossaryunit	13, 16
\print@op@unsrtglossaryunit	15, 16
\printabbreviations	20
\printglossaries	133, 149
\printglossary	20, 133, 149, 151
\printglossary options	
nonumberlist	72
type	135
\printnoidxglossaries	149
\printnoidxglossary	133, 134, 149
\printnumbers	22, 198
\printsymbols	22, 198
\printunsrtglossary	149, 151, 158, 374, 375
\printunsrtglossaryentryprocesshook	159, 162
\printunsrtglossaryhandler	163
\printunsrtglossarypredoglossary	159, 162
\printunsrtglossaryskipentry	159, 162
\printunsrtglossaryunit	15, 16, 164
\printunsrtglossaryunitsetup	163
\ProcessOptions	421
\ProcessOptionsX	27
\protect	28, 108–111, 190–193, 222, 225, 226, 244–258, 260–264, 266, 268–278, 280–286, 288–312,

314–319, 321, 323, 324, 326–330, 332, 334–337, 339, 341–343, 345–350, 354–364	\rGLS 169
\protected@csedef 41, 42, 143, 440, 441	\rGLs 169
\protected@csxdef 42, 142, 440, 441	\rgls 169
\protected@edef 11, 31, 43, 44, 68, 76, 77, 100, 127, 138, 141–143, 163, 164, 185, 186, 208, 225, 380	\rGLSformat 173
\protected@write 11–13, 63, 71, 100, 132, 133, 151, 153–155, 166, 167, 385, 463	\rGlsformat 172
\providecommand 20–22, 34, 58, 71, 96, 97, 100, 117, 125, 131, 132, 146, 147, 153, 166, 374–376, 385, 387–389, 421, 458	\rglsformat 171, 174
\ProvidesFile 373	\rGLSpl 169
\ProvidesPackage .. 5, 374, 420, 458, 465, 488	\rGlSpl 169
	\rglspl 169
	\rGLSplformat 173
	\rGlsplformat 172
	\rglsplformat 171, 174
	\robustify 375
	\romannumeral 440, 441, 449, 490, 491
Q	S
\quad 491	\s@glsxtrfmt 34
\quotechar 211	\s@gls@hyp@opt 99
R	\s@glsxtr@enabletagging 213
\raggedright 426, 427, 430, 431, 461, 467	\s@glsxtrfmt 34
\refstepcounter 26	\s@GlsXtrIfFieldCmpNum 39
\relax 7, 15–17, 21, 22, 24, 27, 29, 30, 35, 40, 62–64, 68, 69, 71, 75, 77, 78, 99, 107, 116, 117, 125, 133, 136, 137, 139, 141, 142, 144, 151, 152, 154, 160, 161, 165, 166, 170, 209–211, 214, 217, 221, 222, 332, 376, 381–383, 434, 436, 438, 442–451, 456, 457, 460–463, 488, 490–493	\s@GlsXtrIfFieldEqNum 39
relsize package 280	\s@GlsXtrIfFieldEqStr 42
\renewcommand . 6, 7, 15–20, 22–32, 47–53, 55, 62–65, 67, 68, 70–74, 77, 94, 96–98, 100, 102, 106, 113–117, 120, 121, 123– 139, 141, 143–147, 161, 163, 169, 197, 199–204, 212–216, 226, 227, 242, 243, 245–351, 375, 384, 419, 421–434, 436– 439, 449–457, 461, 462, 470–489, 492, 493	\s@GlsXtrIfFieldEqXpStr 43
\renewenvironment 422, 424–431, 433, 436, 438, 449, 452, 454, 455, 457, 460, 469–475, 477–483, 485, 486, 488, 492	\s@GlsXtrIfFieldNonZero 39
\renewglossarystyle 421–431, 433, 434, 436–439, 449, 451–457	\s@GlsXtrIfFieldValueInCsvList 38
\renewrobustcmd 80, 102	\s@glsxtrifhasfield 38, 43, 44
\RequireGlossariesExtraLang ... 374, 419	\s@GlsXtrIfXpFieldEqXpStr 43
\RequirePackage 5, 15, 24, 25, 27, 420, 458, 465, 488	\s@GlsXtrStartUnsetBuffering 112
\reserved@a 219	\s@GlsXtrStopUnsetBuffering 113
\reserved@b 219	\s@ifglossaryexists 32
\reserved@d 220	\s@printunsrtglossary 158, 163
\RestoreAcronyms 127, 130	\seealsoname 57, 59
	\seename 54, 57
	\setabbreviationstyle 130, 246, 257
	\SetAcronymLists 129, 130
	\setacronymstyle 128, 130, 131
	\SetDefaultAcronymDisplayStyle 130
	\setentrycounter 144, 378, 380
	\SetGenericNewAcronym 130
	\setglossarystyle 25, 136, 160, 421, 423, 424, 434, 437–439, 451–457, 460
	\setkeys 9, 25, 30, 35, 78, 80, 98, 127, 136, 160, 170, 223, 224
	\setlength 69, 433, 436, 438, 439, 450, 452, 454, 455, 460, 468, 469, 490, 491
	\settowidth 131, 440–449, 468, 490
	\setupglossaries 5, 30
	\sfcode 19, 217, 432

<code>\small</code>	28, 57	289, 292, 294–307, 309, 311, 313, 315–
<code>\smallskip</code>	491	321, 323–339, 341–343, 345–350, 380, 491
<code>soul package</code>	112	
<code>\space</code>	6, 13, 58, 64, 65, 68, 88, 98, 115–117, 123, 125–127, 130–134, 136, 147, 150, 151, 155, 160, 162, 164, 167, 217, 218, 221, 226, 246, 421, 422, 432, 435, 437, 439, 440, 449, 458, 489, 491, 492	
<code>\spacefactor</code>	19, 217, 432	
<code>\stepcounter</code>	175	
<code>\string</code>	6, 11– 13, 31, 58, 63–65, 68, 71, 78, 88, 94, 95, 98, 100, 107, 108, 115–117, 123, 125– 127, 130, 132–136, 146–151, 153–155, 160, 162, 164, 166, 167, 201, 203, 204, 206, 208, 217, 375, 376, 385, 390–419, 463	
<code>\strut</code>	421–432, 438, 466	
<code>\subglossentry</code>	136, 161, 165, 421–432, 434, 436, 438, 450, 461, 470, 472–474, 476, 477, 479, 481, 482, 484, 485, 487, 488, 493	
<code>\subitem</code>	433, 434	
<code>\subsubitem</code>	433, 434	
<code>\symbolname</code>	468, 475, 476, 478, 479, 481, 483, 484, 486	
T		
<code>\tabcolsep</code>	468, 469	
<code>\tablehead</code>	428–431	
<code>\tabletail</code>	428–431	
<code>\tabularnewline</code>	424–432, 467, 470–487	
<code>\TeX</code>	148	
<code>\texorpdfstring</code>	35, 40, 41, 108–110, 212, 352, 364–373	
<code>\textbf</code>	185, 432, 467, 489, 492	
<code>textcase package</code>	350	
<code>\textit</code>	185	
<code>\textmd</code>	185	
<code>\textrm</code>	185	
<code>\textsc</code>	185, 264	
<code>\textsf</code>	185	
<code>\textsl</code>	185	
<code>\textsmaller</code>	280	
<code>\texttt</code>	28, 147–150, 185	
<code>\the</code>	127, 128, 145, 152, 211, 212, 225, 244–268, 270, 272, 275–278, 280–287,	
<code>\theH</code>	31	
<code>\theHglentrycounter</code> .	8, 10–13, 78, 80, 170	
<code>\theindex</code>	207	
<code>\thewrglossary</code>	26	
<code>\this@dialect</code>	374, 419	
<code>\toks@</code>	145, 211, 212, 380	
<code>\toprule</code>	467, 470, 472, 473, 475, 476, 478, 479, 481, 483, 484, 486	
<code>\TrackedDialectClosestSubMatch</code>	44	
<code>tracklang package</code>	44, 153, 389	
<code>\TrackLangGetDefaultScript</code>	419	
<code>\TrackLangIfHasDefaultScript</code>	419	
<code>\TrackLangRequireDialectPrefix</code>	419	
<code>\triangleright</code>	57	
U		
<code>\u</code>	375	
<code>\undef</code>	16, 62, 213	
<code>\underline</code>	215	
<code>\unskip</code>	49, 62, 421	
<code>upgreek package</code>	388	
<code>\usepackage</code>	149, 150	
W		
<code>\warn@nomakeglossaries</code>	133	
<code>\warn@noprintglossary</code>	133, 137, 161	
<code>wrglossary (counter)</code>	25, 26	
<code>\write</code>	58, 117, 125, 132, 146, 147, 154	
X		
<code>\x</code>	175, 380, 385	
<code>\xcapitalisewords</code>	199	
<code>\xdef</code>	136, 161, 163, 376	
<code>\xifinlist</code>	121	
<code>\xifinlistcs</code>	37	
<code>xindy</code>	494	
<code>xindy</code>	16, 131, 494	
<code>xkeyval package</code>	5	
<code>\XKV@checkchoice</code>	72	
<code>\XKV@plfalse</code>	72	
<code>\XKV@resa</code>	72	
<code>\XKV@sttrue</code>	72	