

glossaries-extra.sty v1.36: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-08-18

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	27
1.3 Modifications to Commands Provided by glossaries	39
1.3.1 Existence Checks	43
1.3.2 Document Definitions	51
1.3.3 Existing Glossary Style Modifications	57
1.3.4 Entry Formatting, Hyperlinks and Indexing	61
1.3.5 Entry Counting	98
1.3.6 Acronym Modifications	113
1.3.7 Indexing and Displaying Glossaries	116
1.4 Link Counting	152
1.5 Integration with glossaries-accsupp	154
1.6 Categories	168
1.7 Abbreviations	194
1.7.1 Abbreviation Styles Setup	214
1.7.2 Predefined Styles (Default Font)	217
1.7.3 Predefined Styles (Small Capitals)	234
1.7.4 Predefined Styles (Fake Small Capitals)	249
1.7.5 Predefined Styles (Emphasized)	263
1.7.6 Predefined Styles (User Parentheses Hook)	284
1.7.7 Predefined Styles (Hyphen)	294
1.7.8 Predefined Styles (No Short on First Use)	308
1.8 Using Entries in Headings	310
1.9 Multi-Lingual Support	330
1.10 glossaries-extra-bib2gls.sty	331
2 Style Adjustments (glossaries-extra-stylemods.sty)	367
2.1 Package Initialisation	367
2.2 List-Like Styles	368
2.3 Longtable Styles	371
2.4 Long Ragged Styles	373
2.5 Supertabular Styles	375
2.6 Super Ragged Styles	377
2.7 Inline Style	379
2.8 Tree Styles	379
2.9 Multicolumn Styles	397

3 bookindex style (glossary-bookindex.sty)	403
3.1 Package Initialisation and Options	403
Glossary	409
Change History	410
Index	429

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/08/18 v1.36 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

```
5 \@ifpackageloaded{glossaries}
6 {%
```

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

```
7   \newcommand{\glxstr@doption}[1]{\setupglossaries{#1}}%
8   \let\@glxstr@declareoption\@gls@declareoption
9 }
10 {%
```

Not already loaded, so pass options to glossaries.

```
11   \newcommand{\glxstr@doption}[1]{%
12     \PassOptionsToPackage{#1}{glossaries}%
13   }%
```

Set the defaults.

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

```

undefaction

```

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

```

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

\glstr@record Does nothing by default.

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

```

\glstr@recordsee Does nothing by default.

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

```

\glstr@defaultnumberformat

```
79 \newcommand*{\glstr@defaultnumberformat}{\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\gls@label\@gls@label
89       \glswriteentry{#1}%
90       {%
91         \ifdefempty{\@glsxtr@thevalue}%
92         {%
93           \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
94           \else
95             \let\theHglentrycounter\@glsxtr@theHvalue
96             \fi
97             \glsxtr@saveentrycounter
98             \let\@@do@@wrglossary\@glsxtr@dorecord
99           }%
100          {%
101            \let\theHglentrycounter\@glsxtr@thevalue
102            \let\theHglentrycounter\@glsxtr@theHvalue
103            \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
104          }%
105          \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
106            \glsxtr@@do@wrglossary{#1}%
107          \else
108            \@glsxtrwrglossmark
109
110            Increment associated counter.
111            \glsxtr@inc@wrglossaryctr{#1}%
112            \@@do@@wrglossary
113          \fi
114        }%
115      \fi
116    \endgroup
117  }

```

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

```

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

```

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

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

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

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

Save the label in case it's needed.

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

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

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

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

```
132     \@gls@setdefault@glslink@opts
```

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings.

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

Increment associated counter.

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

Check if noindex option has been used.

```
137     \ifKV@glslink@noindex
138     \else
139       \glswriteentry{#2}%
140       {%
```

Check if thevalue has been set.

```
141         \ifdefempty{\@glsxtr@thevalue}%
142         {%
```

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

```
143         \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
144         \else
145           \let\theHglsentrycounter\@glsxtr@theHvalue
146         \fi
```

Save the entry counter.

```
147         \glsxtr@saveentrycounter
```

Temporarily redefine \@@do@@wrglossary for use with \glxtr@@do@@wrglossary.

```
148      \let\@@do@@wrglossary\glxtr@dorecord
149      }%
150      {%
```

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.)

```
151      \let\theglentrycounter\glxtr@thevalue
152      \let\theHglentrycounter\glxtr@theHvalue
153      \let\@@do@@wrglossary\glxtr@dorecordnodefer
154      }%
155      \ifx\glxtr@record@setting\glxtr@record@setting@alsoindex
156      \glxtr@@do@@wrglossary{#2}%
157      \else
```

No need to escape special characters.

```
158      \@@do@@wrglossary
159      \fi
160      }%
161      \fi
162      \endgroup
163      }%
164      }
```

glslink@prekeys

```
165 \newcommand{\@glxtr@glslink@prekeys}{\glslinkpresetkeys}
```

lslink@postkeys

```
166 \newcommand{\@glxtr@glslink@postkeys}{\glslinkpostsetkeys}
```

lossadd@prekeys

```
167 \newcommand{\@glxtr@glossadd@prekeys}{\glsaddpresetkeys}
```

ossadd@postkeys

```
168 \newcommand{\@glxtr@glossadd@postkeys}{\glsaddpostsetkeys}
```

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

```
169 \newcommand*\@glxtr@dorecord{%
170   \global\let\@glsrecordlocref\theglentrycounter
171   \let\@glxtr@orgprefix\@glo@counterprefix
172   \ifx\theglentrycounter\theHglentrycounter
173     \def\@glo@counterprefix{}%
174   \else
175     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
176       {\theglentrycounter}{\theHglentrycounter}}%
177     }%
178     \@do@gls@getcounterprefix
179   \fi
```

Don't protect the `\@glsrecordloc` 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.

```
180 \protected@write\@auxout{}\string\glsxtr@record
181   {\@gls@label}\@glo@counterprefix{\@gls@counter}\@glsnumberformat}%
182   {\@glsrecordloc}}%
183 \@glsxtr@counterrecordhook
184 \let\@glo@counterprefix\@glsxtr@orgprefix
185 }
```

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

```
186 \newcommand*\@glsxtr@dorecordnodefer{%
187   \ifx\theglsentrycounter\theHglentrycounter
188     \protected@write\@auxout{}\string\glsxtr@record
189       {\@gls@label}\@gls@counter}\@glsnumberformat}%
190     {\theglsentrycounter}}%
191   \else
192     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
193       {\theglsentrycounter}\theHglentrycounter}%
194     }%
195     \@do@gls@getcounterprefix
196     \protected@write\@auxout{}\string\glsxtr@record
197       {\@gls@label}\@glo@counterprefix{\@gls@counter}\@glsnumberformat}%
198     {\theglsentrycounter}}%
199   \fi
200 \@glsxtr@counterrecordhook
201 }
```

`r@recordcounter`

```
202 \newcommand*\@glsxtr@recordcounter{%
203   \@glsxtr@noop@recordcounter
204 }
```

`p@recordcounter`

```
205 \newcommand*\@glsxtr@noop@recordcounter}[1]{%
206   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
207     requires record=only or record=alsoindex package option}{}%
208 }
```

`op@recordcounter`

```
209 \newcommand*\@glsxtr@op@recordcounter}[1]{%
210   \eappto\@glsxtr@counterrecordhook{\noexpand\@glsxtr@docounterrecord{#1}}%
211 }
```

`lsxtr@recordsee` Deal with `\glssee` in record mode. (This doesn't increment the associated counter.)

```

212 \newcommand*{\@glsxtr@recordsee}[2]{%
213   \@@glsxtrwrglossmark
214   \def\@gls@xref{#2}%
215   \@onelevel@sanitize\@gls@xref
216   \protected@write\@auxout{}\string\glsxtr@recordsee{#1}{\@gls@xref}}%
217 }

```

srtglossaryunit

```

218 \newcommand{\printunsrtglossaryunit}{%
219   \print@noop@unsrtglossaryunit
220 }

```

tr@setup@record Initialise.

```

221 \newcommand*{\glsxtr@setup@record}{\let\@do@wrglossary\glsxtr@do@wrglossary}

```

saveentrycounter Only store the entry counter information if the indexing is on.

```

222 \newcommand*{\glsxtr@indexonly@saveentrycounter}{%
223   \ifKV@glslink@noindex
224   \else
225     \glsxtr@saveentrycounter
226   \fi
227 }

```

addloclistfield

```

228 \newcommand*{\glsxtr@addloclistfield}{%
229   \key@ifundefined{glossentry}{loclist}%
230   {%
231     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
232     \appto\@gls@keymap{,{loclist}{loclist}}%
233     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
234     \appto\@newglossaryentryposthook{%
235       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
236     }%
237     \glssetnoexpandfield{loclist}%
238   }%
239   {%

```

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

```

240   \key@ifundefined{glossentry}{location}%
241   {%
242     \define@key{glossentry}{location}{\def\@glo@location{##1}}%
243     \appto\@gls@keymap{,{location}{location}}%
244     \appto\@newglossaryentryprehook{\def\@glo@location{}}%
245     \appto\@newglossaryentryposthook{%
246       \gls@assign@field{\@glo@label}{location}{\@glo@location}%
247     }%
248     \glssetnoexpandfield{location}%
249   }%
250   {%

```

Add a key to store the group heading.

```

251 \key@ifundefined{glossentry}{group}%
252 {%
253   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
254   \appto\@gls@keymap{,{group}{group}}%
255   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
256   \appto\@newglossaryentryposthook{%
257     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
258   }%
259   \glssetnoexpandfield{group}%
260 }%
261 {}%
262 }

```

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

```

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

```

`setting@alsoindex`

```

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

```

`rd@setting@only`

```

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

```

`ord@setting@off`

```

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

```

`record` Now define the record package option.

```

267 \define@choicekey{glossaries-extra.sty}{record}
268 [ \@glsxtr@record@setting\glsxtr@record@nr ]%
269 {off,only,alsoindex}%
270 [only]%
271 {%
272   \ifcase\glsxtr@record@nr\relax

```

Don't record.

```

273   \def\glsxtr@setup@record{%
274     \renewcommand*{\@do@seeglossary}{\@glsxtr@doseeglossary}%
275     \renewcommand*{\@glsxtr@record}[3]{}%
276     \let\@@do@wrglossary\glsxtr@@do@wrglossary
277     \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
278     \let\glsxtrundefaction\@glsxtr@err@undefaction
279     \let\glsxtr@warnonexistsordo\@gobble
280     \let\@@glsxtr@recordcounter\@glsxtr@noop@recordcounter
281     \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
282     \undef\glsxtrsetaliasnoindex
283   }%
284   \or

```

Only record (don't index).

```
285 \def\glxtr@setup@record{%
286   \glxtr@autoseeindexfalse
287   \let\@do@seeglossary\glxtr@recordsee
288   \let\glxtr@record\@glxtr@record
289   \let\@do@wrglossary\glxtr@do@record@wrglossary
290   \let\glxtr@saveentrycounter\relax
291   \let\glxtrundefaction\glxtr@warn@undefaction
292   \let\glxtr@warnonexistssordo\glxtr@warn@onexistssordo
293   \glxtr@addloclistfield
294   \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
295   \let\@glxtr@recordcounter\glxtr@op@recordcounter
296   \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

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

```
297 \def\glxtrsetaliasnoindex{}%
```

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

```
298 \ifdef\glxtr@setupsort@none{\glxtr@setupsort@none}{}%
```

Warn about using `\printglossary`:

```
299 \def\glxtrNoGlossaryWarning{\glxtr@record@noglossarywarning}%
```

Load glossaries-extra-bib2gls:

```
300 \RequirePackage{glossaries-extra-bib2gls}%
301 }%
302 \or
```

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

```
303 \def\glxtr@setup@record{%
304   \renewcommand*{\@do@seeglossary}{\glxtr@do@alsoindex@glossary}%
305   \let\glxtr@record\@glxtr@record
306   \let\@do@wrglossary\glxtr@do@alsoindex@wrglossary
307   \let\glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
308   \let\glxtrundefaction\glxtr@warn@undefaction
309   \let\glxtr@warnonexistssordo\glxtr@warn@onexistssordo
310   \glxtr@addloclistfield
311   \let\@glxtr@recordcounter\glxtr@op@recordcounter
312   \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
313   \undef\glxtrsetaliasnoindex
314 }%
315 \fi
316 }
```

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).

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

Need to provide conditional commands that are backward compatible:

if@glxtrdocdef

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

lsxtrdocdeftrue

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

sxtrdocdeffalse

```
320 \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.

```
321 \define@choicekey{glossaries-extra.sty}{docdef}
322 [\@glxtr@docdefsetting\@glxtr@docdefval]%
323 {false,true,restricted,atom}[true]%
324 {%
325   \ifnum\@glxtr@docdefval>1\relax
326     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
327   \else
328     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}%
329   \fi
330 }
```

ocdefrestricted

```
331 \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).

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

indexcrossrefs Automatically index cross references at the end of the document

```
333 \define@boolkey{glossaries-extra.sty}{@glxtr}{indexcrossrefs}[true]{%
334   \if@glxtrindexcrossrefs
335   \else
336     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
337   \fi
338 }
```

Switch off since this can increase the build time.

```
339 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
340 \newcommand*{\@glxtr@autoindexcrossrefs}{\@glxtr@indexcrossrefstrue}
```

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

```
341 \define@boolkey{glossaries-extra.sty}{@glxtr@}{autoseeindex}[true]{%
342 }
343 \@glxtr@autoseeindextrue
```

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

347 \@glxtr@declareoption{nowarn}{%
348 \let\GlossariesExtraWarning\@gobble
349 \let\GlossariesExtraWarningNoLine\@gobble
350 \glxtr@doption{nowarn}%
351 }
```

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

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

postdot Shortcut for nopostdot=false

```
353 \@glxtr@declareoption{postdot}{%
354 \glxtr@doption{nopostdot=false}%
355 \renewcommand*{\@glxtr@defpostpunc}{%
356 \renewcommand*{\glspostdescription}{%
357 \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
358 }%
359 }
```

nopostdot Needs to redefine \@glxtr@defpostpunc

```
360 \define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
361 \glxtr@doption{nopostdot=#1}%
362 \renewcommand*{\@glxtr@defpostpunc}{%
363 \renewcommand*{\glspostdescription}{%
364 \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
365 }%
366 }
```

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

```
367 \define@key{glossaries-extra.sty}{postpunc}{%
368 \glxtr@doption{nopostdot=false}%
```



```

369 \ifstrequal{#1}{dot}%
370 {%
371   \renewcommand*{\@glsxtr@defpostpunc}{%
372     \renewcommand*{\glspostdescription}{.\spacefactor\sffcode‘\. }%
373   }%
374 }%
375 {%
376   \ifstrequal{#1}{comma}%
377   {%
378     \renewcommand*{\@glsxtr@defpostpunc}{%
379       \renewcommand*{\glspostdescription}{,}%
380     }%
381   }%
382   {%
383     \ifstrequal{#1}{none}%
384     {%
385       \glsxtr@dooption{nopostdot=true}%
386       \renewcommand*{\@glsxtr@defpostpunc}{%
387         \renewcommand*{\glspostdescription}{}%
388       }%
389     }%
390     {%
391       \renewcommand*{\@glsxtr@defpostpunc}{%
392         \renewcommand*{\glspostdescription}{#1}%
393       }%
394     }%
395   }%
396 }%
397 }

```

`\glsxtrabbrvtype` Glossary type for abbreviations.

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

`\bbreviationsdef` Set by abbreviations option.

```
399 \newcommand*{\@glsxtr@abbreviationsdef}{}%
```

`\bbreviationsdef`

```

400 \newcommand*{\@glsxtr@doabbreviationsdef}{%
401   \@ifpackageloaded{babel}%
402   {\providecommand{\abbreviationsname}{\acronymname}}%
403   {\providecommand{\abbreviationsname}{Abbreviations}}%
404   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
405   \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
406   \newcommand*{\printabbreviations}[1][1]{%
407     \printglossary[type=\glsxtrabbrvtype,##1]%
408   }%
409   \disable@keys{glossaries-extra.sty}{abbreviations}%

```

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

```

410 \ifglsacronym
411 \else
412   \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
413 \fi
414 }%

```

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

```

415 \@glsxtr@declareoption{abbreviations}{%
416   \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
417 }

```

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

```

418 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
419   \newcommand*{\ab}{\cgl}s}%
420   \newcommand*{\abp}{\cgl spl}%
421   \newcommand*{\as}{\glsxtrshort}%
422   \newcommand*{\asp}{\glsxtrshortpl}%
423   \newcommand*{\al}{\glsxtrlong}%
424   \newcommand*{\alp}{\glsxtrlongpl}%
425   \newcommand*{\af}{\glsxtrfull}%
426   \newcommand*{\afp}{\glsxtrfullpl}%
427   \newcommand*{\Ab}{\cGls}%
428   \newcommand*{\Abp}{\cGL spl}%
429   \newcommand*{\As}{\Glsxtrshort}%
430   \newcommand*{\Asp}{\Glsxtrshortpl}%
431   \newcommand*{\Al}{\Glsxtrlong}%
432   \newcommand*{\Alp}{\Glsxtrlongpl}%
433   \newcommand*{\Af}{\Glsxtrfull}%
434   \newcommand*{\Afp}{\Glsxtrfullpl}%
435   \newcommand*{\AB}{\cGLS}%
436   \newcommand*{\ABP}{\cGLS pl}%
437   \newcommand*{\AS}{\GLSxtrshort}%
438   \newcommand*{\ASP}{\GLSxtrshortpl}%
439   \newcommand*{\AL}{\GLSxtrlong}%
440   \newcommand*{\ALP}{\GLSxtrlongpl}%
441   \newcommand*{\AF}{\GLSxtrfull}%
442   \newcommand*{\AFP}{\GLSxtrfullpl}%

```

```

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

```

Disable this command after it's been used.

```

444   \let\GlsXtrDefineAbbreviationShortcuts\relax
445 }

```

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

```

446 \newcommand*{\GlsXtrDefineAcShortcuts}{%

```

```

447 \newcommand*{\ac}{\cgl{s}}%
448 \newcommand*{\acp}{\cgl{s}pl}%
449 \newcommand*{\acs}{\gl{xtr}short}%
450 \newcommand*{\acsp}{\gl{xtr}shortpl}%
451 \newcommand*{\acl}{\gl{xtr}long}%
452 \newcommand*{\aclp}{\gl{xtr}longpl}%
453 \newcommand*{\acf}{\gl{xtr}full}%
454 \newcommand*{\acfp}{\gl{xtr}fullpl}%
455 \newcommand*{\Ac}{\cGl{s}}%
456 \newcommand*{\Acp}{\cGl{s}pl}%
457 \newcommand*{\Acs}{\Gl{xtr}short}%
458 \newcommand*{\Acsp}{\Gl{xtr}shortpl}%
459 \newcommand*{\Acl}{\Gl{xtr}long}%
460 \newcommand*{\Aclp}{\Gl{xtr}longpl}%
461 \newcommand*{\Acf}{\Gl{xtr}full}%
462 \newcommand*{\Acfp}{\Gl{xtr}fullpl}%
463 \newcommand*{\AC}{\cGL{S}}%
464 \newcommand*{\ACP}{\cGL{S}pl}%
465 \newcommand*{\ACS}{\GL{Xtr}short}%
466 \newcommand*{\ACSP}{\GL{Xtr}shortpl}%
467 \newcommand*{\ACL}{\GL{Xtr}long}%
468 \newcommand*{\ACLP}{\GL{Xtr}longpl}%
469 \newcommand*{\ACF}{\GL{Xtr}full}%
470 \newcommand*{\ACFP}{\GL{Xtr}fullpl}%

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

```

Disable this command after it's been used.

```

472 \let\GlsXtrDefineAcShortcuts\relax
473 }

```

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

```

474 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
475 \newcommand*{\newentry}{\newglossaryentry}%
476 \ifdef\printsymbols
477 {%
478 \newcommand*{\newsym}{\gl{xtr}newsymbol}%
479 }{}%
480 \ifdef\printnumbers
481 {%
482 \newcommand*{\newnum}{\gl{xtr}newnumber}%
483 }{}%
484 \let\GlsXtrDefineOtherShortcuts\relax
485 }

```

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.

```

486 \newcommand*{\@glxtr@setupshortcuts}{%
tr@shortcutsval Store the value of the shortcuts option. (Needed by bib2gls.)
487 \newcommand*{\@glxtr@shortcutsval}{\ifglacrshortcutsacro\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 short-
cuts=all and shortcuts=none. Multiple use of this option in the same option list will over-
ride each other. New to v1.17: shortcuts=ac which implements \GlsXtrDefineAcShortcuts
(not included in shortcuts=all as it conflicts with other shortcuts).

488 \define@choicekey{glossaries-extra.sty}{shortcuts}%
489 [\@glxtr@shortcutsval\@glxtr@shortcutsnr]%
490 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
491 \ifcase\@glxtr@shortcutsnr\relax % acronyms
492 \renewcommand*{\@glxtr@setupshortcuts}{%
493 \glacrshortcutstrue
494 \DefineAcronymSynonyms
495 }%
496 \or % acro
497 \renewcommand*{\@glxtr@setupshortcuts}{%
498 \glacrshortcutstrue
499 \DefineAcronymSynonyms
500 }%
501 \or % abbreviations
502 \renewcommand*{\@glxtr@setupshortcuts}{%
503 \GlsXtrDefineAbbreviationShortcuts
504 }%
505 \or % abbr
506 \renewcommand*{\@glxtr@setupshortcuts}{%
507 \GlsXtrDefineAbbreviationShortcuts
508 }%
509 \or % other
510 \renewcommand*{\@glxtr@setupshortcuts}{%
511 \GlsXtrDefineOtherShortcuts
512 }%
513 \or % all
514 \renewcommand*{\@glxtr@setupshortcuts}{%
515 \glacrshortcutstrue

516 \GlsXtrDefineAcShortcuts
517 \GlsXtrDefineAbbreviationShortcuts
518 \GlsXtrDefineOtherShortcuts
519 }%
520 \or % true
521 \renewcommand*{\@glxtr@setupshortcuts}{%
522 \glacrshortcutstrue

523 \GlsXtrDefineAcShortcuts
524 \GlsXtrDefineAbbreviationShortcuts

```

```

525     \GlsXtrDefineOtherShortcuts
526 }%

527 \or % ac
528   \renewcommand*{\@glxtr@setupshortcuts}{%
529     \glsacrshortcutstrue
530     \GlsXtrDefineAcShortcuts
531   }%

  Leave none and false as last option.
532   \else % none, false
533     \renewcommand*{\@glxtr@setupshortcuts}{}%
534   \fi
535 }

lsxtr@doaccsupp
536 \newcommand*{\@glxtr@doaccsupp}{}

accsupp  If accsupp, load glossaries-accsupp package.
537 \@glxtr@declareoption{accsupp}{%
538   \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

GlossaryWarning  Warning text displayed in document if the external glossary file given by the argument is miss-
ing.
539 \newcommand{\glxtrNoGlossaryWarning}[1]{%
540   \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
541   \@glxtr@defaultnoglossarywarning{#1}%
542 }

omissingglstext  If true, suppress the text and warning produced if the external glossary file is missing.
543 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
544   [\@glxtr@nomissingglstextval\@glxtr@nomissingglstextnr]%
545   {true,false}[true]{%
546     \ifcase\@glxtr@nomissingglstextnr\relax % true
547       \renewcommand{\glxtrNoGlossaryWarning}[1]{\null}%
548     \else % false
549       \renewcommand{\glxtrNoGlossaryWarning}[1]{%
550         \@glxtr@defaultnoglossarywarning{#1}%
551       }%
552     \fi
553   }

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

xtr@redefstyles
554 \newcommand*{\@glxtr@redefstyles}{}

```

stylemods

```
555 \define@key{glossaries-extra.sty}{stylemods}[default]{%
556   \ifstrequal{#1}{default}%
557   {%
558     \renewcommand*{\@glxtr@redefstyles}{%
559       \RequirePackage{glossaries-extra-stylemods}}%
560   }%
561   {%
562     \ifstrequal{#1}{all}%
563     {%
564       \renewcommand*{\@glxtr@redefstyles}{%
565         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
566         \RequirePackage{glossaries-extra-stylemods}%
567       }%
568     }%
569     {%
570       \renewcommand*{\@glxtr@redefstyles}{}%
571       \@for\@glxtr@tmp:=#1\do{%
572         \IfFileExists{glossary-\@glxtr@tmp.sty}%
573         {%
574           \eappto\@glxtr@redefstyles{%
575             \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
576         }%
577         {%
578           \PackageError{glossaries-extra}%
579             {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
580              doesn’t exist (did you mean to use the ‘style’ key?)}%
581             {The list of values (#1) in the ‘stylemods’ key should
582              match the glossary-xxx.sty files provided with
583              glossaries.sty}%
584         }%
585       }%
586       \appto\@glxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
587     }
588   }%
589 }
```

glxtr@do@style

```
590 \newcommand*{\@glxtr@do@style}{}
```

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

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

Defer actual style change:

```
592 \renewcommand*{\@glxtr@do@style}{%
```

Set this as the default style:

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

Set this style:

```
594 \setglossarystyle{#1}%
595 }%
596 }
```

`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.

```
597 \newcommand*{\glxtr@inc@wrglossaryctr}[1]{}
```

`ocationHyperlink`

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

The first two arguments are always control sequences.

```
598 \newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
599 \glxtrhyperlink{#1#2#3}{#3}%
600 }
```

`cationhyperlink`

```
601 \newcommand*{\@glxtr@wrglossary@locationhyperlink}[3]{%
602 \pageref{wrglossary.#3}%
603 }
```

`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.

```
604 \@glxtr@declareoption{indexcounter}{%
605 \glxtr@doption{counter=wrglossary}%
606 \ifundef\c@wrglossary
607 {%
608 \newcounter{wrglossary}%
609 \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
610 }%
611 }%
612 \renewcommand*{\glxtr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is `wrglossary`.

```
613 \ifdefstring\@gl@counter{wrglossary}%
614 {%
615 \refstepcounter{wrglossary}%
616 \label{wrglossary.\thewrglossary}%
617 }%
```

```

618     {}%
619   }%
620   \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
621     \ifdefstring\glsentrycounter{wrglossary}%
622     {%
623       \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
624     }%
625     {\glsxtrhyperlink{##1##2##3}{##3}}%
626   }%
627 }

```

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

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

```

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

```

629 \newcommand*{\@glsxtrwrglossmark}{}
630 \AtBeginDocument{\renewcommand*{\@glsxtrwrglossmark}{\@glsxtrwrglossmark}}

```

`sxtrwrglossmark` Does nothing by default.

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

```

`debug` Provide extra debug options.

```

632 \define@choicekey{glossaries-extra.sty}{debug}
633 [ \@glsxtr@debugval \@glsxtr@debugnr ]%
634 {true,false,showtargets,showwrgloss,all}[true]{%
635   \ifcase\@glsxtr@debugnr\relax % true
636     \glsxtr@doption{debug=true}%
637     \renewcommand*{\@glsxtrwrglossmark}{}%
638   \or % false
639     \glsxtr@doption{debug=false}%
640     \renewcommand*{\@glsxtrwrglossmark}{}%
641   \or % showtargets
642     \glsxtr@doption{debug=showtargets}%
643   \or % showwrgloss
644     \glsxtr@doption{debug=true}%
645     \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
646   \or % all
647     \glsxtr@doption{debug=showtargets}%
648     \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
649   \fi
650 }

```

Pass all other options to glossaries.

```

651 \DeclareOptionX*{%
652   \expandafter\glsxtr@doption\expandafter{\CurrentOption}}

```

Process options.

```
653 \ProcessOptionsX

```


Load glossaries if not already loaded.

```
654 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
655 \@glsxtr@doaccsupp
```

Redefine \glspostdescription if required.

```
656 \@glsxtr@defpostpunc
```

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

```
657 \def\glsshowtarget#1{%
658   \glsxtrtitleorpdforheading
659   {%
660     \ifmmode
661       \texttt{\small [#1]}%
662     \else
663       \ifinner
664         \texttt{\small [#1]}%
665       \else
666         \marginpar{\texttt{\small #1}}%
667       \fi
668     \fi
669   }%
670   {[#1]}%
671   {\texttt{\small [#1]}}%
672 }
```

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

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

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

```
674 \newcommand*{\@glsxtr@doseeglossary}[2]{%
675   \glsdoifexists{#1}%
676   {%
677     \@glsxtrwrglossmark
678     \@glsxtr@org@doseeglossary{#1}{#2}%
679   }%
680 }
```

oindex@glossary

```
681 \newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
682   \@glsxtr@recordsee{#1}{#2}%
683   \@glsxtr@doseeglossary{#1}{#2}%
684 }
```

@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.)

```
685 \let\@glsxtr@org@gloautosee\@glo@autosee
```

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

```
686 \if@glxtr@autoseeindex
687 \else
688   \ifdef\@glxtr@org@gloautosee
689     {}%
690     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
691       option requires at least v4.30 of glossaries.sty}%
692       {You need to update the glossaries.sty package}%
693     }
694 \fi
```

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

```
695 \ifdef\@glo@autosee
696 {%
697   \renewcommand*{\@glo@autosee}{%
698     \if@glxtr@autoseeindex\@glxtr@org@gloautosee\fi}%
699 }%
700 {}
```

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

```
701 \renewcommand*{\gls@checkseeallowed}{%
702   \if@glxtr@autoseeindex\@gls@see@noindex\fi
703 }
```

Define abbreviations glossaries if required.

```
704 \@glxtr@abbreviationsdef
705 \let\@glxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
706 \@glxtr@setupshortcuts
```

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

```
707 \@glxtr@redef@for@gl@sentries
```

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

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

Now define the user command:

```
709 \newcommand*{\glossariesextrasetup}[1]{%
710   \let\glxtr@setup@record\relax
711   \let\glxtr@setupshortcuts\relax
712   \let\glxtr@redef@for@gl@sentries\relax
713   \setkeys{glossaries-extra.sty}{#1}%
714   \@glxtr@abbreviationsdef
715   \let\@glxtr@abbreviationsdef\relax
716   \@glxtr@setupshortcuts
```

```

717 \glxtr@setup@record
718 \@glxtr@redef@for@gl@sentries
719 }

```

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

```

720 \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.

```

721 \newcommand*{\glxtr@@do@wrglossary}[1]{%
722 \@glxtrwrglossmark
723 \glxtr@inc@wrglossaryctr{#1}%
724 \glxtr@org@@do@wrglossary{#1}%
725 }

```

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

```

726 \let\glxtr@saveentrycounter\@gls@saveentrycounter

```

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

```

727 \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter

```

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

`sxtrdialecthook`

```

728 \newcommand*{\@glxtrdialecthook}{}

```

Set up record option if required.

```

729 \glxtr@setup@record

```

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

```

730 \AtBeginDocument{%
731 \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
732 \def\@glxtrundeftag{\glxtrundeftag}%
733 }

```

1.2 Extra Utilities

`unusedOrUndefined`

```

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

```

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

```

734 \newcommand*{\GlsXtrIfUnusedOrUndefined}[3]{%
735 \ifgl@entryexists{#1}%

```

```

736 {\ifbool{glo@\glsdetoklabel{#1}@flag}{#3}{#2}}%
737 {#2}}%
738 }

```

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

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

```

739 \newcommand{\glstrifemptyglossary}[3]{%
740   \ifcsdef{glolist@#1}%
741   {%
742     \ifcsstring{glolist@#1}{,}{#2}{#3}%
743   }%
744   {%
745     \glstrundefaction{Glossary type '#1' doesn't exist}{}%
746     #2%
747   }%
748 }

```

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

```

749 \newcommand*\glstrifkeydefined[3]{%
750   \key@ifundefined{glossentry}{#1}{#3}{#2}%
751 }

```

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

```

752 \newcommand*\glstrprovidestoragekey{%
753   \@ifstar\@sglsxtr@provide@storagekey\@glxtr@provide@storagekey
754 }

```

`\glxtrprovide@storagekey` Unstarred version.

```

755 \newcommand*\@glxtr@provide@storagekey[3]{%
756   \key@ifundefined{glossentry}{#1}%
757   {%
758     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
759     \appto\@gls@keymap{,{#1}{#1}}%
760     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
761     \appto\@newglossaryentryposthook{%
762       \letcs{@glo@tmp}{@glo@#1}%
763       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
764     }%

```

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

```

765 \ifblank{#3}
766 {}%
767 {%
768 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
769 }%
770 }%
771 {%

```

Provide the no-link command if not already defined.

```

772 \ifblank{#3}
773 {}%
774 {%
775 \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
776 }%
777 }%
778 }

```

`\provide@storagekey` Starred version.

```

779 \newcommand*\s@glxtr@provide@storagekey[1]{%
780 \key@ifundefined{glossentry}{#1}%
781 {%
782 \expandafter\newcommand\expandafter*\expandafter
783 {\csname gls@assign@#1@field\endcsname}[2]{%
784 \@gls@expand@field{##1}{#1}{##2}%
785 }%
786 }%
787 }%
788 \@glxtr@provide@addstoragekey{#1}%
789 }

```

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

```

790 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

791 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

792 \newrobustcmd*\glxtrfmt{\@ifstar\s@glxtrfmt\@glxtrfmt}

```

`\@glxtrfmt` Unstarred form.

```

793 \newcommand*\@glxtrfmt[3][\@glxtrfmt]{#1}{#2}{#3}{}}

```

`\s@glstrfmt` Starred form.

```

794 \newcommand*{\s@glstrfmt}[3][]{%
795   \new@ifnextchar[{\s@glstrfmt{#1}{#2}{#3}}%
796   {\s@glstrfmt{#1}{#2}{#3}{}}%
797 }

```

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

```

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

```

`\@glstrfmt` Actual inner working.

```

799 \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).

```

800 \begingroup
801   \def\glslabel{#2}%
802   \glsdofexistsordo{#2}%
803   {%
804     \ifglshasfield{\GlsXtrFmtField}{#2}%
805     {%
806       \let\do@glslink@checkfirsthyper\relax
807       \expandafter\@glslink\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
808       {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
809     }%
810     {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
811   }%
812   {%

```

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

```

813     \begingroup
814       \@glslsetdefault@glslink@opts
815       \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
816       \ifKV@glslink@noindex\else\glssadd{#2}\fi
817     \endgroup
818     \glstrfmtdisplay{@firstofone}{#3}{#4}%
819   }%
820 \endgroup
821 }

```

`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).

```

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

```

`\glxtrentryfmt` No link or indexing.

```

823 \ifdef\texorpdfstring
824 {
825   \newcommand*{\glxtrentryfmt}[2]{%

```

```

826 \texorpdfstring{\@glstrententryfmt{#1}{#2}}{#2}%
827 }
828 }
829 {
830 \newcommand*{\glstrententryfmt}{\@glstrententryfmt}
831 }

```

@glstrententryfmt

```

832 \newrobustcmd*{\@glstrententryfmt}[2]{%
833 \glsoifexistsordo{#1}%
834 {%
835 \ifglshasfield{\GlsXtrFmtField}{#1}%
836 {%
837 \csuse{\glscurrentfieldvalue}{#2}%
838 }%
839 {#2}%
840 }%
841 {#2}%
842 }

```

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.

```

843 \newcommand*{\glxtrfieldlistadd}[3]{%
844 \listcsadd{glo@\glsetoklabel{#1}@#2}{#3}%
845 }

```

trfieldlistgadd Similarly but uses \listcsgadd.

```

846 \newcommand*{\glxtrfieldlistgadd}[3]{%
847 \listcsgadd{glo@\glsetoklabel{#1}@#2}{#3}%
848 }

```

trfieldlisteadadd Similarly but uses \listcseadd.

```

849 \newcommand*{\glxtrfieldlisteadadd}[3]{%
850 \listcseadd{glo@\glsetoklabel{#1}@#2}{#3}%
851 }

```

trfieldlistxadd Similarly but uses \listcsxadd.

```

852 \newcommand*{\glxtrfieldlistxadd}[3]{%
853 \listcsxadd{glo@\glsetoklabel{#1}@#2}{#3}%
854 }

```

Now provide commands to iterate over these lists.

fielddolistloop

```

855 \newcommand*{\glxtrfielddolistloop}[2]{%
856 \dolistcsloop{glo@\glsetoklabel{#1}@#2}%
857 }

```

ieldforlistloop

```
858 \newcommand*\glstrfieldforlistloop}[3]{%
859   \forlistcsloop{#3}{glo@glstetoklabel{#1}@#2}%
860 }
```

List element tests:

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

```
861 \newcommand*\glstrfieldifinlist}[5]{%
862   \ifinlistcs{#3}{glo@glstetoklabel{#1}@#2}{#4}{#5}%
863 }
```

rfieldxifinlist Expands item.

```
864 \newcommand*\glstrfieldxifinlist}[5]{%
865   \xifinlistcs{#3}{glo@glstetoklabel{#1}@#2}{#4}{#5}%
866 }
```

lsxtrforcsvfield

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

```
867 \newcommand*\glstrforcsvfield}[3]{%
868   \@glstrifhasfield{#2}{#1}%
869   {%
870     \let\glstrendfor\endfortrue
871     \@for\@glstr@label:=\glscurrentfieldvalue\do
872     {\expandafter#3\expandafter{\@glstr@label}}}%
873   }%
874 }
```

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.

```
875 \newrobustcmd{\glstrifhasfield}{%
876   \@ifstar{\s@glstrifhasfield}{\@glstrifhasfield}%
877 }
```

lsxtrifhasfield Unstarred version adds grouping.

```
878 \newcommand{\@glstrifhasfield}[4]{%
879   {\s@glstrifhasfield{#1}{#2}{#3}{#4}}%
880 }
```

lsxtrifhasfield Starred version omits grouping.

```
881 \newcommand{\s@glstrifhasfield}[4]{%
882   \letcs\glscurrentfieldvalue{glo@glstetoklabel{#2}@#1}%
883   \ifundef\glscurrentfieldvalue
```



```

884 {#4}%
885 {%
886 \ifdefempty\glscurrentfieldvalue{#4}{#3}%
887 }%
888 }

```

`rIfFieldNonZero` Designed for numeric fields.

```

889 \newcommand{\GlsXtrIfFieldNonZero}[4]{%
890 \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
891 }

```

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

Designed for numeric fields.

```

892 \newcommand{\GlsXtrIfFieldEqNum}[5]{%
893 \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
894 }

```

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

Designed for numeric fields.

```

895 \newcommand{\GlsXtrIfFieldCmpNum}[6]{%
896 {%
897 \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
898 \ifundef\glscurrentfieldvalue
899 {\def\glscurrentfieldvalue{0}}%
900 {%
901 \ifdefempty\glscurrentfieldvalue
902 {\def\glscurrentfieldvalue{0}}%
903 }%
904 }%
905 \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
906 }%
907 }

```

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

Just uses `\ifcsundef`.

```

908 \newcommand{\GlsXtrIfFieldUndef}[2]{%
909 \ifcsundef{glo@\glsetoklabel{#2}@#1}%
910 }

```

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

```

911 \newcommand*{\glxtrusefield}[2]{%
912   \@gls@entry@field{#1}{#2}%
913 }

```

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

```

914 \newcommand*{\Glsxtrusefield}[2]{%
915   \@gls@entry@field{#1}{#2}%
916 }

```

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

```

917 \newcommand*{\glxtrdeffield}[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}

```

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

```

918 \newcommand*{\glxtrdeffield}[2]{\protected@csedef{glo@\glsdetoklabel{#1}@#2}}

```

`etfieldifexists`

```

919 \newcommand*{\glxtrsetfieldifexists}[3]{\glsdoidexists{#1}{#3}}

```

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

```

920 \newrobustcmd*{\GlsXtrSetField}[3]{%
921   \glxtrsetfieldifexists{#1}{#2}%
922   {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
923 }

```

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

```

924 \newrobustcmd*{\GlsXtrLetField}[3]{%
925   \glxtrsetfieldifexists{#1}{#2}%
926   {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
927 }

```

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

```

928 \newrobustcmd*{\csGlsXtrLetField}[3]{%
929   \glxtrsetfieldifexists{#1}{#2}%
930   {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
931 }

```

`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.

```

932 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
933   \glxtrsetfieldifexists{#1}{#2}%
934   {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
935 }

```

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

```
936 \newrobustcmd*{\gGlsXtrSetField}[3]{%
937   \glsxtrsetfieldifexists{#1}{#2}%
938   {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
939 }
```

xGlsXtrSetField

```
940 \newrobustcmd*{\xGlsXtrSetField}[3]{%
941   \glsxtrsetfieldifexists{#1}{#2}%
942   {\protected@csxdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
943 }
```

eGlsXtrSetField

```
944 \newrobustcmd*{\eGlsXtrSetField}[3]{%
945   \glsxtrsetfieldifexists{#1}{#2}%
946   {\protected@csedef{glo@\glsdetoklabel{#1}@#2}{#3}}%
947 }
```

XtrIfFieldEqStr

```
948 \newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
949   \glsxtrifhasfield{#1}{#2}%
950   {%
951     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
952   }%
953   {#5}%
954 }
```

rIfFieldEqXpStr Like the above but first expands the string.

```
955 \newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
956   \glsxtrifhasfield{#1}{#2}%
957   {%
958     \protected@edef\@gls@tmp{#3}%
959     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
960   }%
961   {#5}%
962 }
```

fXpFieldEqXpStr Like the above but also expands the field value.

```
963 \newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
964   \glsxtrifhasfield{#1}{#2}%
965   {%
966     \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
967     \let\glscurrentfieldvalue\@gls@tmp
968     \protected@edef\@gls@tmp{#3}%
969     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
970   }%
971   {#5}%
972 }
```

lsXtrForeignText

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

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

```
973 \ifdef\foreignlanguage
974 {
975   \ifdef\GetTrackedDialectFromLanguageTag
976   {
977     \newcommand{\GlsXtrForeignText}[2]{%
```

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

```
978     \let\@glsxtr@org@currentfieldvalue\glscurrentfieldvalue
979     \glsxtrifhasfield{\GlsXtrForeignTextField}{#1}%
980     {%
981       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
982       {\glscurrentfieldvalue}{\@glsxtr@dialect}%
983       \let\@glsxtr@locale\glscurrentfieldvalue
984       \let\glscurrentfieldvalue\@glsxtr@org@currentfieldvalue
985       \ifdefempty\@glsxtr@dialect
986       {%
```

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

```
987     \ifundef\TrackedDialectClosestSubMatch
988     {%
989       \GlossariesExtraWarning{Can't obtain dialect label
990       (tracklang v1.3.6+ required)}%
991     }%
992     {\let\@glsxtr@dialect\TrackedDialectClosestSubMatch}%
993   }%
994   {}%
995   \ifdefempty\@glsxtr@dialect
996   {%
```

No tracked dialect found for the root language.

```
997   }%
998   {%
```

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

```
999     \ifcsundef{captions\@glsxtr@dialect}{}%
1000     {%
```

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

```
1001     \IfTrackedDialectHasMapping{\@glsxtr@dialect}%
1002     {%
1003       \edef\@glsxtr@dialect{%
1004         \GetTrackedDialectToMapping{\@glsxtr@dialect}}%
```

Does a caption hook exist for this?

```
1005         \ifcsundef{captions\@glsxtr@dialect}{}%  
1006         {%
```

No mapping. Try root language label instead.

```
1007         \ifcsundef{captions\@tracklang@lang}{}%  
1008         {%  
1009         \let\@glsxtr@dialect\@tracklang@lang  
1010         }%  
1011         }%  
1012         }%  
1013         {%
```

No mapping. Try root language label instead.

```
1014         \ifcsundef{captions\@tracklang@lang}{}%  
1015         {%  
1016         \let\@glsxtr@dialect\@tracklang@lang  
1017         }%  
1018         }%  
1019         }%  
1020         }%  
1021         \ifdefempty\@glsxtr@dialect  
1022         {%  
1023         \GlsXtrUnknownDialectWarning{\@glsxtr@locale}{\@tracklang@lang}%  
1024         #2%  
1025         }%  
1026         {\foreignlanguage{\@glsxtr@dialect}{#2}}%  
1027         }%  
1028         {#2}% key not set  
1029     }  
1030 }  
1031 {  
1032     \newcommand{\GlsXtrForeignText}[2]{%  
1033         \GlossariesExtraWarning{Can't encapsulate foreign text:  
1034             tracklang v1.3.6+ required}%  
1035         #2%  
1036     }  
1037 }  
1038 }  
1039 {  
    \foreignlanguage isn't defined so just do <text>.  
1040     \newcommand{\GlsXtrForeignText}[2]{#2}  
1041 }
```

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

```
1042 \newcommand*{\GlsXtrForeignTextField}{userii}
```

nDialectWarning

```
1043 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
```

```

1044 \GlossariesExtraWarning{Can't determine valid dialect label
1045   for locale '#1' (root language: #2)}%
1046 }

```

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

```

1047 \ifdef\GlsEntryCounterLabelPrefix
1048 {%
1049   \newcommand*\glstrpageref}[1]{%
1050     \ifglentrycounter
1051       \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1052     \else
1053       \ifglssubentrycounter
1054         \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1055       \else
1056         \gls{#1}%
1057       \fi
1058     \fi
1059   }
1060 }%
1061 {%
1062   \newcommand*\glstrpageref}[1]{%
1063     \ifglentrycounter
1064       \pageref{glsentry-\glsdetoklabel{#1}}%
1065     \else
1066       \ifglssubentrycounter
1067         \pageref{glsentry-\glsdetoklabel{#1}}%
1068       \else
1069         \gls{#1}%
1070       \fi
1071     \fi
1072   }
1073 }%

```

`lossarypreamble`

```

1074 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1075   \ifcsdef{glolist@#1}%
1076   {%
1077     \ifcsundef{@glossarypreamble@#1}%
1078     {\csdef{@glossarypreamble@#1}{}}%
1079     {}%
1080     \csappto{@glossarypreamble@#1}{#2}%
1081   }%
1082   {%
1083     \GlossariesExtraWarning{Glossary '#1' is not defined}%
1084   }%
1085 }

```

lossarypreamble

```

1086 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1087   \ifcsdef{glolist@#1}%
1088   {%
1089     \ifcsundef{@glossarypreamble@#1}%
1090     {\csdef{@glossarypreamble@#1}{}}%
1091     {}%
1092     \cspretoto{@glossarypreamble@#1}{#2}%
1093   }%
1094   {%
1095     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1096   }%
1097 }

```

1.3 Modifications to Commands Provided by glossaries

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

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

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

```

1098 \renewcommand*{\ifglsused}[3]{%
1099   \glsdoifexists{#1}{\ifbool{glo@glstoklabel{#1}@flag}{#2}{#3}}%
1100 }

```

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

ewglossaryentry

```

1101 \renewcommand*{\longnewglossaryentry}{%
1102   \@ifstar{\glstr@s@longnewglossaryentry}{\glstr@longnewglossaryentry}
1103 }

```

ewglossaryentry Starred version.

```

1104 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1105   \glsdoifnoexists{#1}%
1106   {%
1107     \bgroup
1108     \let\@org@newglossaryentryprehook\@newglossaryentryprehook

```

```

1109     \long\def\@newglossaryentryprehook{%
1110         \long\def\@glo@desc{#3}%
1111         \@org@newglossaryentryprehook
1112     }%
1113     \renewcommand*{\gls@assign@desc}[1]{%
1114         \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1115         \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1116     }
1117     \gls@defglossaryentry{#1}{#2}%
1118 \egroup
1119 }%
1120 }

```

`\newglossaryentry` Unstarred version.

```

1121 \newcommand{\@glsxtr@longnewglossaryentry}[3]{%
1122     \glsdoifnoexists{#1}%
1123     {%
1124         \bgroup
1125         \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1126         \long\def\@newglossaryentryprehook{%
1127             \long\def\@glo@desc{#3\glsxtrpostlongdescription}%
1128             \@org@newglossaryentryprehook
1129         }%
1130         \renewcommand*{\gls@assign@desc}[1]{%
1131             \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

The following is different from the base glossaries.sty:

```

1132         \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1133     }
1134     \gls@defglossaryentry{#1}{#2}%
1135 \egroup
1136 }%
1137 }

```

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

```

1138 \newcommand*{\glsxtrpostlongdescription}{\leavevmode\unskip\nopostdesc}

```

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

`\ignoredglossary` Redefine to check for star.

```

1139 \renewcommand{\newignoredglossary}{%
1140     \@ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
1141 }

```

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

```

1142 \newcommand*{\glsxtr@org@newignoredglossary}[1]{%
1143     \ifcsdef{glolist@#1}
1144     {%

```



```

1145 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1146 }%
1147 {%
1148 \ifdefempty\@ignored@glossaries
1149 {%
1150 \edef\@ignored@glossaries{#1}%
1151 }%
1152 {%
1153 \eappto\@ignored@glossaries{,#1}%
1154 }%
1155 \csgdef{glolist@#1}{,}%
1156 \ifcsundef{gls@#1@entryfmt}%
1157 {%
1158 \defglentryfmt[#1]{\glentryfmt}%
1159 }%
1160 {}%
1161 \ifdefempty\@gls@nohyperlist
1162 {%
1163 \renewcommand*{\@gls@nohyperlist}{#1}%
1164 }%
1165 {%
1166 \eappto\@gls@nohyperlist{,#1}%
1167 }%
1168 }%
1169 }

```

ignoredglossary Starred form.

```

1170 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
1171 \ifcsdef{glolist@#1}
1172 {%
1173 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1174 }%
1175 {%
1176 \ifdefempty\@ignored@glossaries
1177 {%
1178 \edef\@ignored@glossaries{#1}%
1179 }%
1180 {%
1181 \eappto\@ignored@glossaries{,#1}%
1182 }%
1183 \csgdef{glolist@#1}{,}%
1184 \ifcsundef{gls@#1@entryfmt}%
1185 {%
1186 \defglentryfmt[#1]{\glentryfmt}%
1187 }%
1188 {}%
1189 }%
1190 }

```

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

```

1191 \glsifusetranslator
1192 {%
1193   \renewcommand*{\glsettoctitle}[1]{%
1194     \ifcsdef{gls@tr@set@#1@toctitle}%
1195       {%
1196         \csuse{gls@tr@set@#1@toctitle}%
1197       }%
1198     {%
1199       \ifcsdef{@glotype@#1@title}%
1200         {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1201         {\def\glossarytoctitle{\glossarytitle}}%
1202       }%
1203     }%
1204 }
1205 {
1206   \renewcommand*{\glsettoctitle}[1]{%
1207     \ifcsdef{@glotype@#1@title}%
1208       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1209       {\def\glossarytoctitle{\glossarytitle}}%
1210   }
1211 }

```

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

```

1212 \newcommand{\provideignoredglossary}{%
1213   \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1214 }

```

`ignoredglossary` Unstarred version.

```

1215 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1216   \ifcsdef{glolist@#1}
1217   {}%
1218   {%
1219     \ifdefempty\@ignored@glossaries
1220     {%
1221       \edef\@ignored@glossaries{#1}%
1222     }%
1223     {%
1224       \eappto\@ignored@glossaries{,#1}%
1225     }%
1226     \csgdef{glolist@#1}{,}%
1227     \ifcsundef{gls@#1@entryfmt}%
1228     {%
1229       \defglentryfmt[#1]{\glentryfmt}%
1230     }%
1231     {}%
1232     \ifdefempty\@gls@nohyperlist
1233     {%

```

```

1234      \renewcommand*{\@gls@nohyperlist}{#1}%
1235  }%
1236  {%
1237      \eappto\@gls@nohyperlist{, #1}%
1238  }%
1239  }%
1240 }

```

`ignoredglossary` Starred form.

```

1241 \newcommand*{\glsxtr@s@provideignoredglossary}[1]{%
1242   \ifcsdef{glolist@#1}
1243   {%
1244   {%
1245     \ifdefempty\@ignored@glossaries
1246     {%
1247       \edef\@ignored@glossaries{#1}%
1248     }%
1249     {%
1250       \eappto\@ignored@glossaries{, #1}%
1251     }%
1252     \csgdef{glolist@#1}{,}%
1253     \ifcsundef{gls@#1@entryfmt}%
1254     {%
1255       \defglsentryfmt[#1]{\glsentryfmt}%
1256     }%
1257     {}%
1258   }%
1259 }

```

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

```

1260 \newcommand*{\glsxtrcopytoglossary}[2]{%
1261   \glsdoifexists{#1}%
1262   {%
1263     \ifcsdef{glolist@#2}
1264     {%
1265       \cseappto{glolist@#2}{#1,}%
1266     }%
1267     {%
1268       \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1269     }%
1270   }%
1271 }

```

1.3.1 Existence Checks

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

```

1272 \renewcommand{\glsdoifexists}[2]{%
1273   \ifglsentryexists{#1}{#2}%

```

1274 {%
 Define \glslabel in case it's needed after this command (for example in the post-link hook).

```
1275 \edef\glslabel{\glsdetoklabel{#1}}%
1276 \glstrundefaction{Glossary entry '\glslabel'
1277 has not been defined}{You need to define a glossary entry before
1278 you can reference it.}%
1279 }%
1280 }
```

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

```
1281 \renewcommand{\glsdoifnoexists}[2]{%
1282 \ifglstryexists{#1}{%
1283 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1284 has already been defined}{}}{#2}%
1285 }
```

sdoifexistsordo 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.

```
1286 \ifdef\glsdoifexistsordo
1287 {%
1288 \renewcommand{\glsdoifexistsordo}[3]{%
1289 \ifglstryexists{#1}{#2}%
1290 {%
1291 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1292 has not been defined}{You need to define a glossary entry
1293 before you can use it.}%
1294 #3%
1295 }%
1296 }%
1297 }
1298 {%
1299 \glstr@warnonexistsordo\glsdoifexistsordo
1300 \newcommand{\glsdoifexistsordo}[3]{%
1301 \ifglstryexists{#1}{#2}%
1302 {%
1303 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1304 has not been defined}{You need to define a glossary entry
1305 before you can use it.}%
1306 #3%
1307 }%
1308 }%
1309 }
```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```
1310 \ifdef\doifglossarynoexistsordo
1311 {%
1312 \renewcommand{\doifglossarynoexistsordo}[3]{%

```

```

1313 \ifglossaryexists{#1}%
1314 {%
1315 \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1316 #3%
1317 }%
1318 {#2}%
1319 }%
1320 }
1321 {%
1322 \glstr@warnonexistsordo\doifglossarynoexistsordo
1323 \newcommand{\doifglossarynoexistsordo}[3]{%
1324 \ifglossaryexists{#1}%
1325 {%
1326 \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1327 #3%
1328 }%
1329 {#2}%
1330 }%
1331 }
1332

```

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

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

```

1333 \appto\@newglossaryentryposthook{%
1334 \ifdefvoid\@glo@see
1335 {\csxdef{glo@\@glo@label @see}{}}%
1336 {%
1337 \csxdef{glo@\@glo@label @see}{\@glo@see}%
1338 \if@glstr@autoseeindex
1339 \@glstr@autoindexcrossrefs
1340 \fi
1341 }%
1342 }
1343 \appto\@gls@keymap{,{see}{see}}

```

`\glstrusesee` Apply `\glseeformat` to the see key if not empty.

```

1344 \newcommand*{\glstrusesee}[1]{%
1345 \glsdexists{#1}%
1346 {%
1347 \letcs{\@glo@see}{glo\glsetoklabel{#1}@see}%
1348 \ifdefempty\@glo@see
1349 {}%
1350 {%
1351 \expandafter\glstr@usesee\@glo@see\@end@glstr@usesee
1352 }%
1353 }%

```

```

1354 }

\glxtr@usesee
1355 \newcommand*{\glxtr@usesee}[1][\seename]{%
1356   \@glxtr@usesee[#1]%
1357 }

\@glxtr@usesee
1358 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1359   \glxtruseseeformat{#1}{#2}%
1360 }

xtruseseeformat  The format used by \glxtrusesee. The first argument is the tag (such as \seename). The
                  second argument is the comma-separated list of cross-referenced labels.
1361 \newcommand*{\glxtruseseeformat}[2]{%
1362   \glssseeformat{#1}{#2}{}%
1363 }

lsseeitemformat  glossaries originally defined \glssseeitemformat to use \glstryname but in v3.0 this was
                  switched to use \glstrytext due to problems occurring with the name field being sani-
                  tized. Since this is no longer a problem, glossaries-extra restores the original definition as it
                  makes more sense to use the name in the cross-reference list. This still uses \glssaccesstext
                  for abbreviations.
1364 \renewcommand*{\glssseeitemformat}[1]{%
1365   \ifglshashshort{\glslabel}{\glssaccesstext{#1}}{\glssaccessname{#1}}%
1366 }

lsxtruseseealso  Apply \glssseeformat to the seealso key if not empty. There's no optional tag to worry about
                  here.
1367 \newcommand*{\glxtruseseealso}[1]{%
1368   \glsdexists{#1}%
1369   {%
1370     \letcs{\@glo@see}{glo@glstdetoklabel{#1}@seealso}%
1371     \ifdefempty\@glo@see
1372     {%
1373       {%
1374         \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
1375       }%
1376     }%
1377 }

sesealsoformat  The format used by \glxtruseseealso. The argument is the comma-separated list of
                  cross-referenced labels.
1378 \newcommand*{\glxtruseseealsoformat}[1]{%
1379   \glssseeformat[\seealso]{#1}{}%
1380 }

```

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

```
1381 \newrobustcmd{\glxtrseeelist}[1]{%
1382   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1383 }
```

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

```
1384 \providecommand{\seealso}{see also}
```

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

```
1385 \ifdef\@xdycrossrefhook
1386 {
```

Add the cross-reference class definition to the hook.

```
1387   \appto\@xdycrossrefhook{%
1388     \write\glswrite{(define-crossref-class \string"seealso\string"
1389       :unverified )}%
1390     \write\glswrite{(markup-crossref-list
1391       :class \string"seealso\string"^^J\space\space\space
1392       :open \string"\string\glxtruseealsoformat\glsoopenbrace\string"
1393       :close \string"\glsclosebrace\string")}%
1394   }
```

Append to class list.

```
1395   \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

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

```
1396   \newrobustcmd*{\glxtrindexseealso}[2]{%
1397     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1398       \@glxtr@recordsee{#1}{#2}%
1399     \fi
1400     \glsoifexists{#1}%
1401     {%
1402       \@@glxtrwrglossmark
1403       \def\@gls@xref{#2}%
1404       \@onelevel@sanitize\@gls@xref
1405       \@gls@checkmkidxchars\@gls@xref
1406       \gls@glossary{\csname glo@#1@type\endcsname}{%
1407         (indexentry
1408           :tkey (\csname glo@#1@index\endcsname)
1409           :xref (\string"\@gls@xref\string")
1410           :attr \string"seealso\string"
1411         )
1412       }%
1413     }%
1414   }
```

```

1415 }
1416 {
    xindy not in use or glossaries version too old to support this.
1417 \newrobustcmd*{\glxtrindexseealso}{\glssee[\seealsoname]}
1418 }

```

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`.

```

1419 \ifdef\gls@set@xr@key
1420 {

```

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

```

1421 \define@key{glossentry}{alias}{%
1422 \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1423 }
1424 \define@key{glossentry}{seealso}{%
1425 \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1426 }

```

Add to the key mappings.

```

1427 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1428 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1429 \appto\@newglossaryentryposthook{%
1430 \ifdefvoid\@glo@seealso
1431 {\csxdef{glo@\@glo@label @seealso}{}}%
1432 {%
1433 \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1434 \if@glxtr@autoseeindex
1435 \@glxtr@autoindexcrossrefs
1436 \fi
1437 }%

```

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

```

1438 \ifdefvoid\@glo@alias
1439 {\csxdef{glo@\@glo@label @alias}{}}%
1440 {%
1441 \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1442 }%
1443 }

```

Provide user-level commands to access the values.


```

\glxtralias
1444 \newcommand*\glxtralias}[1]{\@gls@entry@field{#1}{alias}}

trseealsolabels
1445 \newcommand*\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

Add to the \@glo@autosee hook.
1446 \appto\@glo@autoseehook{%
1447 \ifdefvoid\@glo@alias
1448 {%
1449 \ifdefvoid\@glo@seealso
1450 }%
1451 {%
1452 \edef\@do@glsssee{\noexpand\glxtrindexseealso
1453 {\@glo@label}{\@glo@seealso}}%
1454 \@do@glsssee
1455 }%
1456 }%
1457 {%

Add cross-reference if see key hasn't been used.
1458 \ifdefvoid\@glo@see
1459 {%
1460 \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1461 \@do@glsssee
1462 }%
1463 }%
1464 }%
1465 }%
1466 }
1467 {

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

\glxtralias
1468 \glsaddstoragekey*{alias}{\glxtralias}

trseealsolabels
1469 \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.
1470 \appto\@newglossaryentryposthook{%
1471 \ifcsvoid{glo@\@glo@label @alias}%
1472 {%
1473 \ifcsvoid{glo@\@glo@label @seealso}%
1474 }%

```

```

1475     {%
1476     \edef\@do@glsssee{\noexpand\glxtrindexseealso
1477     {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1478     \@do@glsssee
1479     }%
1480     }%
1481     {%

```

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

```

1482     \ifdefvoid\@glo@see
1483     {%
1484     \edef\@do@glsssee{\noexpand\glsssee
1485     {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1486     \@do@glsssee
1487     }%
1488     }%
1489     }%
1490 }
1491 }

```

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

```

1492 \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.

```

1493 \newcommand*\glxtraddallcrossrefs{%
1494   \forallglossaries{\@glo@type}%
1495   {%
1496     \for@gl@sentries[\@glo@type]{\@glo@label}%
1497     {%
1498       \if@gl@sused{\@glo@label}%
1499       {\expandafter\glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1500     }%
1501   }%
1502 }

```

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

```

1503 \newcommand*\@glxtr@addunusedxrefs[1]{%
1504   \letcs{\@glo@see}{glo@\gl@detoklabel{#1}@see}%
1505   \ifdefvoid\@glo@see
1506   {%
1507   {%
1508     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1509   }%
1510   \letcs{\@glo@see}{glo@\gl@detoklabel{#1}@seealso}%
1511   \ifdefvoid\@glo@see
1512   {%
1513   {%

```

```

1514 \expandafter\glsxtr@addunused\@glo@see\@end\glsxtr@addunused
1515 }%
1516 }

```

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

```

1517 \newcommand*{\glsxtr@addunused}[1][]{%
1518 \@glsxtr@addunused
1519 }

```

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

```

1520 \def\@glsxtr@addunused#1\@end\glsxtr@addunused{%
1521 \@for\@glsxtr@label:=#1\do
1522 {%
1523 \ifglsused{\@glsxtr@label}{}%
1524 {%
1525 \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1526 \glsunset{\@glsxtr@label}%
1527 \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1528 }%
1529 }%
1530 }

```

`\glsxtrunusedformat`

```

1531 \newcommand*{\glsxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

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

```

1532 \ifdef\gls@begindocdefs
1533 {%
1534 \renewcommand*{\gls@begindocdefs}{%
1535 \ifnum\@glsxtr@docdefval=1\relax
1536 \@gls@enablesavenonumberlist
1537 \edef\@gls@restoreat{%
1538 \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1539 \makeatletter
1540 \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1541 \@gls@restoreat
1542 \undef\@gls@restoreat
1543 \gls@defdocnewglossaryentry
1544 \else
1545 \ifnum\@glsxtr@docdefval=3\relax

```

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

```

1546 \@gls@enablesavenonumberlist

```

```

1547      \let\gls@checkseeallowed\relax
1548      \let\newglossaryentry\new@atom@glossaryentry
1549      \global\newwrite\@gls@deffile
1550      \immediate\openout\@gls@deffile=\jobname.glsdefs

```

Write all currently defined entries.

```

1551      \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
1552      \fi
1553      \fi
1554  }
1555 }
1556 {%
1557   \ifnum\@glsxtr@docdefval=3\relax
1558     \PackageError{glossaries-extra}{Package option
1559       'docdef=\@glsxtr@docdefsetting' requires at least version 4.37
1560       of the base glossaries.sty package}{}
1561     \fi
1562 }

```

m@glossaryentry

```

1563 \newrobustcmd{\new@atom@glossaryentry}[2]{%
1564   \gls@defglossaryentry{#1}{#2}%
1565   \@gls@writedef{#1}%
1566 }

```

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.

```

1567 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1568 \renewcommand{\makenoidxglossaries}{%
1569   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1570   {%
1571     \glsxtr@orgmakenoidxglossaries

```

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

```

1572   \renewcommand{\@do@seeglossary}[2]{%
1573     \@glsxtrwrglossmark
1574     \edef\@gls@label{\glsdetoklabel{##1}}%
1575     \protected@write\@auxout{}{%
1576       \string\@gls@reference
1577       {\csname glo@\@gls@label @type\endcsname}%
1578       {\@gls@label}%
1579       {%
1580         \string\glsseeformat##2{}%
1581       }%
1582     }%
1583   }%

```

Check for docdefs=restricted:

```

1584   \if@glsxtrdocdefrestricted

```

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

```

1585 \renewcommand*{\@gls@reference}[3]{%
1586 \ifcsundef{\glsref@##1}{\csgdef{\glsref@##1}{}}{}%
1587 \ifinlistcs{##2}{\glsref@##1}%
1588 {}%
1589 {\listcsgadd{\glsref@##1}{##2}}%
1590 \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1591 {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1592 {}%
1593 \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1594 }%
1595 \else

```

Disable document definitions.

```

1596 \@glsxtrdocdeffalse
1597 \fi
1598 \disable@keys{glossaries-extra}{docdef}%
1599 }%
1600 {%
1601 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1602 not permitted\MessageBreak
1603 with record=\@glsxtr@record@setting\space package option}%
1604 {You may only use \string\makenoidxglossaries\space with the
1605 record=off option}%
1606 }%
1607 }

```

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

```

1608 \renewcommand*{\gls@defdocnewglossaryentry}{%
1609 \ifcase\@glsxtr@docdefval
1610 docdef=false:
1611 \renewcommand*{\newglossaryentry}[2]{%
1612 \PackageError{glossaries-extra}{Glossary entries must
1613 be \MessageBreak defined in the preamble with \MessageBreak
1614 package option 'docdef=false'\MessageBreak(consider using
1615 'docdef=restricted')}{Move your glossary definitions to
1616 the preamble. You can also put them in a \MessageBreak separate file
1617 and load them with \string\loadglsentries.}%
1618 }%
1619 \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.

```

1619 \let\gls@checkseeallowed\relax
1620 \let\newglossaryentry\newglossaryentry
1621 \else

```

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

```

1622 \let\gls@checkseeallowed\relax
1623 \fi
1624 }%

```

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

`rEnableOnTheFly`

```

1625 \newcommand*{\GlsXtrEnableOnTheFly}{%
1626 \ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1627 }

```

`rEnableOnTheFly`

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

```

1628 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1629 \renewcommand*{\glsdetoklabel}[1]{%
1630 \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1631 {%
1632 \expandafter\detokenize\expandafter{##1}%
1633 }%
1634 {\detokenize{##1}}}%
1635 }%
1636 \@GlsXtrEnableOnTheFly
1637 }
1638 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1639 \expandafter\if\glsbackslash#1%
1640 #3%
1641 \else
1642 #4%
1643 \fi
1644 }

```

`sxtrstarflywarn`

```

1645 \newcommand*{\glsxtrstarflywarn}{%
1646 \GlossariesExtraWarning{Experimental starred version of
1647 \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1648 read the warnings in the glossaries-extra user manual)}}%
1649 }

```

`rEnableOnTheFly`

```

1650 \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:

```

\glstrcat
1651 \newcommand*\glstrcat{general}

\glstr
1652 \newcommand*\glstr[1] [] {%
1653 \def\glstr@keylist{##1}%
1654 \@glstr
1655 }

\@glstr
1656 \newcommand*\@glstr[2] [] {%
1657 \ifglstryexists{##2}%
1658 {%
1659 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1660 }%
1661 {%
1662 \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1663 description={\nopostdesc},##1}%
1664 }%
1665 \expandafter\gls\expandafter[\glstr@keylist]{##2}%
1666 }

\Glsxtr
1667 \newcommand*\Glsxtr[1] [] {%
1668 \def\glstr@keylist{##1}%
1669 \@Glsxtr
1670 }

\@Glsxtr
1671 \newcommand*\@Glsxtr[2] [] {%
1672 \ifglstryexists{##2}%
1673 {%
1674 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1675 }%
1676 {%
1677 \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1678 description={\nopostdesc},##1}%
1679 }%
1680 \expandafter\Gls\expandafter[\glstr@keylist]{##2}%
1681 }

\glstrpl
1682 \newcommand*\glstrpl[1] [] {%
1683 \def\glstr@keylist{##1}%
1684 \@glstrpl
1685 }

\@glstrpl

```

```

1686 \newcommand*{\@glxstrpl}[2][ ]{%
1687   \ifglentryexists{##2}%
1688   {%
1689     \ifblank{##1}{ }\GlsXtrWarning{##1}{##2}}%
1690   }%
1691   {%
1692     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
1693       description={\nopostdesc},##1}%
1694   }%
1695   \expandafter\glsp\expandafter[\glxtr@keylist]{##2}%
1696 }

```

\Glsxtrpl

```

1697 \newcommand*{\Glsxtrpl}[1][ ]{%
1698   \def\glxtr@keylist{##1}%
1699   \@Glsxtrpl
1700 }

```

\@Glsxtrpl

```

1701 \newcommand*{\@Glsxtrpl}[2][ ]{%
1702   \ifglentryexists{##2}
1703   {%
1704     \ifblank{##1}{ }\GlsXtrWarning{##1}{##2}}%
1705   }%
1706   {%
1707     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
1708       description={\nopostdesc},##1}%
1709   }%
1710   \expandafter\glsp\expandafter[\glxtr@keylist]{##2}%
1711 }

```

\GlsXtrWarning

```

1712 \newcommand*{\GlsXtrWarning}[2]{%
1713   \def\glxtr@optlist{##1}%
1714   \@onelevel@sanitize\glxtr@optlist
1715   \GlossariesExtraWarning{The options ‘\glxtr@optlist’ have
1716     been ignored for entry ‘##2’ as it has already been defined}%
1717 }

```

Disable commands after the glossary:

```

1718 \renewcommand\@printglossary[2]{%
1719   \def\glxtr@printglossopts{##1}%
1720   \@glxtr@orgprintglossary{##1}{##2}%
1721   \def\glxtr{\@glxtr@disabledflycommand\glxtr}%
1722   \def\glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1723   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1724   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1725 }

```


abledflycommand

```

1726 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1727   \PackageError{glossaries-extra}%
1728   {\string##1\space can't be used after any of the \MessageBreak
1729    glossaries have been displayed}%
1730   {The on-the-fly commands enabled by
1731    \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1732    before the glossaries. If you want to use any entries \MessageBreak
1733    after any of the glossaries, you must use the standard \MessageBreak
1734    method of first defining the entry and then using the \MessageBreak
1735    entry with commands like \string\gls}%
1736   \@glxtr@disabledflycommand
1737 }%
1738 \newcommand*{\@glxtr@disabledflycommand}[2][\{##2}

```

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

```

1739 \let\GlsXtrEnableOnTheFly\relax
1740 }
1741 \@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.

```

1742 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

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

etglossarystyle

```

1743 \renewcommand*{\setglossarystyle}[1]{%
1744   \ifcsundef{@glsstyle@#1}%
1745   {%
1746     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1747   }%
1748   {%
1749     \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

1750   \protected@edef\@glxtr@current@style{#1}%
1751 }%
1752 \ifx\@glossary@default@style\relax
1753   \protected@edef\@glossary@default@style{#1}%
1754 \fi
1755 }

```

In case we have an old version of glossaries:

```

1756 \ifdef\@glossary@default@style
1757 {}

```

```

1758 {%
1759   \let\@glossary@default@style\relax
1760 }

```

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

```

1761 \ifdef\glslistdottedwidth
1762 {%
1763   \ifdim\glslistdottedwidth=.5\hsize
1764     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1765     \AtBeginDocument{%
1766       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1767         \setlength{\glslistdottedwidth}{.5\columnwidth}%
1768       \fi
1769     }%
1770   \fi
1771 }
1772 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

1773 \ifdef\glsdescwidth
1774 {%
1775   \ifdim\glsdescwidth=.6\hsize
1776     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1777     \AtBeginDocument{%
1778       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1779         \setlength{\glsdescwidth}{.6\columnwidth}%
1780       \fi
1781     }%
1782   \fi
1783 }
1784 {}%

```

and for \glspagelistwidth:

lspagelistwidth

```

1785 \ifdef\glspagelistwidth
1786 {%
1787   \ifdim\glspagelistwidth=.1\hsize
1788     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1789     \AtBeginDocument{%
1790       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1791         \setlength{\glspagelistwidth}{.1\columnwidth}%
1792       \fi
1793     }%
1794   \fi
1795 }
1796 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```
1797 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1798 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1799   \glsnonumberlistfalse
1800   \renewcommand*{\glossaryentrynumbers}[1]{%
1801     \ifglsentryexists{\glscurrententrylabel}%
1802     {%
1803       \@glsxtrpreloctag
1804       \GlsXtrFormatLocationList{#1}%
1805       \@glsxtrpostloctag
1806       \gls@save@numberlist{#1}%
1807     }{}%
1808   }%
1809 \else
1810   \glsnonumberlisttrue
1811   \renewcommand*{\glossaryentrynumbers}[1]{%
1812     \ifglsentryexists{\glscurrententrylabel}%
1813     {%
1814       \gls@save@numberlist{#1}%
1815     }{}%
1816   }%
1817 \fi
```

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

```
1818 \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

```
1819 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1820   \let\@glsxtrpreloctag\@glsxtrpreloctag
1821   \let\@glsxtrpostloctag\@glsxtrpostloctag
1822   \renewcommand*{\@glsxtr@pagetag}{#1}%
1823   \renewcommand*{\@glsxtr@pagestag}{#2}%
1824   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
1825     \csgdef{\@glsxtr@preloctag@##1}{##2}%
1826   }%
1827   \renewcommand*{\@glsxtr@doloctag}{%
1828     \ifcsundef{\@glsxtr@preloctag\@glscurrententrylabel}%
1829     {%
1830       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1831         Rerun required}%
1832     }%
1833   }%
```

```

1834      \csuse{@glsxtr@preloctag@glscurrententrylabel}%
1835    }%
1836  }%
1837 }
1838 \onlypreamble\GlsXtrEnablePreLocationTag

```

glsxtrpreloctag

```

1839 \newcommand*{\@@glsxtrpreloctag}{%
1840   \let\@glsxtr@org@delimN\delimN
1841   \let\@glsxtr@org@delimR\delimR
1842   \let\@glsxtr@org@glsignore\glsignore
      \gdef is required as the delimiters may occur inside a scope.
1843   \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1844   \renewcommand*{\delimN}{%
1845     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1846     \@glsxtr@org@delimN}%
1847   \renewcommand*{\delimR}{%
1848     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1849     \@glsxtr@org@delimR}%
1850   \renewcommand*{\glsignore}[1]{%
1851     \gdef\@glsxtr@thisloctag{\relax}%
1852     \@glsxtr@org@glsignore{##1}}%
1853   \@glsxtr@doloctag
1854 }

```

glsxtrpreloctag

```

1855 \newcommand*{\@glsxtrpreloctag}{}

```

@glsxtr@pagetag

```

1856 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

1857 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

1858 \newcommand*{\@@glsxtrpostloctag}{%
1859   \let\delimN\@glsxtr@org@delimN
1860   \let\delimR\@glsxtr@org@delimR
1861   \let\glsignore\@glsxtr@org@glsignore
1862   \protected@write\@auxout{}%
1863     {\string\@glsxtr@savepreloctag{\glscurrententrylabel}{\@glsxtr@thisloctag}}%
1864 }

```

lsxtrpostloctag

```

1865 \newcommand*{\@glsxtrpostloctag}{}

```

lsxtr@preloctag

```
1866 \newcommand*{\@glxtr@savepreloctag}[2]{%
1867 \protected@write\@auxout{}{%
1868   \string\providecommand\string\@glxtr@savepreloctag[2]{}}
```

glxtr@doloctag

```
1869 \newcommand*{\@glxtr@doloctag}{}
```

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

```
1870 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
1871   \XKV@plfalse
1872   \XKV@sttrue
1873   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1874   {%
1875     \csname glsnonumberlist\XKV@resa\endcsname
1876     \ifglsnonumberlist
1877       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1878     \else
1879       \def\glossaryentrynumbers##1{%
1880         \@glxtr@preloctag
1881         \GlsXtrFormatLocationList{##1}%
1882         \@glxtr@postloctag
1883         \gls@save@numberlist{##1}}%
1884       \fi
1885   }%
1886 }
```

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.

```
1887 \renewcommand*{\glsentryfmt}{%
1888   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}{}}%
1889   \glsifregular{\glslabel}%
1890   {\glxtrregularfont{\glsentryfmt}}%
1891   {%
1892     \ifglshasshort{\glslabel}%
1893     {\glxtrabbreviationfont{\glxtrgenabbrvfmt}}%
1894     {\glxtrregularfont{\glsentryfmt}}%
1895   }%
1896 }
```

`sxtrregularfont` Font used for regular entries.

```
1897 \newcommand*{\glxsxtrregularfont}[1]{#1}
```

`bbreviationfont` Font used for abbreviation entries.

```
1898 \newcommand*{\glxstrabbreviationfont}[1]{#1}
```

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

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

```
1899 \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).

```
1900 \@glxsxtr@record{#2}{#3}{glslink}%
1901 \glsdoifexists{#3}%
1902 {%
```

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).

```
1903 \let\glxsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1904 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1905 \def\glscustomtext{#4}%
1906 \@glxsxtr@field@linkdefs
1907 #1%
1908 \@gls@link[#2]{#3}{#4}%
1909 \let\ifKV@glslink@hyper\glxsxtrorg@ifKV@glslink@hyper
1910 }%
1911 \glspostlinkhook
1912 }
```

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

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

```
1913 \let\@glxsxtr@org@gls@\@gls@
1914 \def\@gls@#1#2{%
1915 \@glxsxtr@record{#1}{#2}{glslink}%
1916 \@glxsxtr@org@gls@{#1}{#2}%
1917 }%
```

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

```
1918 \let\@glxsxtr@org@glspl@\@glspl@
1919 \def\@glspl@#1#2{%
```

```

1920 \@glstr@record{#1}{#2}{glslink}%
1921 \@glstr@org@glsp1@{#1}{#2}%
1922 }%

```

\@Gls@ Save the original definition and redefine.

```

1923 \let\@glstr@org@Gls@\@Gls@
1924 \def\@Gls@#1#2{%
1925 \@glstr@record{#1}{#2}{glslink}%
1926 \@glstr@org@Gls@{#1}{#2}%
1927 }%

```

\@Glspl@ Save the original definition and redefine.

```

1928 \let\@glstr@org@Glspl@\@Glspl@
1929 \def\@Glspl@#1#2{%
1930 \@glstr@record{#1}{#2}{glslink}%
1931 \@glstr@org@Glspl@{#1}{#2}%
1932 }%

```

\@GLS@ Save the original definition and redefine.

```

1933 \let\@glstr@org@GLS@\@GLS@
1934 \def\@GLS@#1#2{%
1935 \@glstr@record{#1}{#2}{glslink}%
1936 \@glstr@org@GLS@{#1}{#2}%
1937 }%

```

\@GLSpl@ Save the original definition and redefine.

```

1938 \let\@glstr@org@GLSpl@\@GLSpl@
1939 \def\@GLSpl@#1#2{%
1940 \@glstr@record{#1}{#2}{glslink}%
1941 \@glstr@org@GLSpl@{#1}{#2}%
1942 }%

```

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

```

1943 \renewcommand*{\@glsdisp}[3][{}]{%
1944 \@glstr@record{#1}{#2}{glslink}%
1945 \glsdoifexists{#2}{%
1946 \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1947 \let\glsifplural\@secondoftwo
1948 \let\glscapscase\@firstofthree
1949 \def\glscustomtext{#3}%
1950 \def\glsinsert{}%
1951 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1952 \@gl@link[#1]{#2}{\@glo@text}%
1953 \ifKV@glslink@local
1954 \glslocalunset{#2}%
1955 \else
1956 \glsunset{#2}%

```

```

1957 \fi
1958 }%
1959 \glspostlinkhook
1960 }

```

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

```

1961 \renewcommand*{\@gls@link}[3][\%
1962 \@glxtr@record{#1}{#2}{glslink}%
1963 \glsdoifexistsordo{#2}%
1964 {%
1965 \let\do@gls@link@checkfirsthyper\relax

```

Post-link hook commands need initialising.

```

1966 \def\glscustomtext{#3}%
1967 \@glxtr@field@linkdefs
1968 \@gls@link[#1]{#2}{#3}%
1969 }%
1970 {%
1971 \glstextformat{#3}%
1972 }%
1973 \glspostlinkhook
1974 }

```

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

```

1975 \newcommand*{\glxtrinitwrgloss}{%
1976 \glsifattribute{glslabel}{wrgloss}{after}%
1977 {%
1978 \glxtrinitwrglossbeforefalse
1979 }%
1980 {%
1981 \glxtrinitwrglossbeforetrue
1982 }%
1983 }

```

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

```

1984 \newif\ifglxtrinitwrglossbefore
1985 \glxtrinitwrglossbeforetrue

```

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

```

1986 \define@choicekey{glslink}{wrgloss}%
1987 [\@glxtr@wrglossval\@glxtr@wrglossnr]%
1988 {before,after}%
1989 {%
1990 \ifcase\@glxtr@wrglossnr\relax
1991 \glxtrinitwrglossbeforetrue
1992 \or
1993 \glxtrinitwrglossbeforefalse
1994 \fi
1995 }

```



```

1996 \define@key{glslink}{thevalue}{\def\@glstr@thevalue{#1}}

1997 \define@key{glslink}{theHvalue}{\def\@glstr@theHvalue{#1}}

tr@hyperoutside  Define a hyperoutside key to determine whether \hyperlink should be outside \glstextformat.
1998 \define@boolkey{glslink}[glstr@]{hyperoutside}[true]{}
1999 \glstr@hyperoutsidettrue

ocal@textformat  Provide a key to locally change the text format.
2000 \define@key{glslink}{textformat}{%
2001   \ifcsdef{#1}
2002   {%
2003     \letcs{\@glstr@local@textformat}{#1}%
2004   }%
2005   {%
2006     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2007   }%
2008 }

2009 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

nithyperoutside  Set the default if the hyperoutside is omitted.
2010 \newcommand*{\glstrnithyperoutside}{%
2011   \glsifattribute{\glslabel}{hyperoutside}{false}%
2012   {%
2013     \glstr@hyperoutsidetfalse
2014   }%
2015   {%
2016     \glstr@hyperoutsidettrue
2017   }%
2018 }

r@inc@linkcount  Does nothing by default.
2019 \newcommand*{\glstr@inc@linkcount}{}

slinkpresetkeys  User hook performed immediately before options are set. Does nothing by default.
2020 \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.
2021 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2022   \protected@edef\@glstr@tmp{#2}%
2023   \expandafter#1\expandafter{\@glstr@tmp}%
2024 }

\@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.

```

```

2025 \def\@gls@link[#1]#2#3{%
2026   \leavevmode
2027   \edef\glslabel{\glsdetoklabel{#2}}%
2028   \def\@gls@link@opts{#1}%
2029   \let\@gls@link@label\glslabel
2030   \let\@gls@numberformat\@glsxtr@defaultnumberformat
2031   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2032   \edef\gls@type{\csname glo@\glslabel @type\endcsname}%
2033   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

    Save current value of \glslinkprefix:
2034   \let\@glsxtr@org@glslinkprefix\glslinkprefix

    Initialise \@glsxtr@local@textformat
2035   \let\@glsxtr@local@textformat\relax

    Initialise thevalue and theHvalue (v1.19).
2036   \def\@glsxtr@thevalue{%
2037     \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%

    Initialise when indexing should occur (new to v1.14).
2038   \glsxtrinitwrgloss

    Initialise whether \hyperlink should be outside \gls@textformat (new to v1.21).
2039   \glsxtrinithyperoutside

    Note that the default link options may override \glsxtrinitwrgloss.
2040   \@gls@setdefault@glslink@opts

    Increment link counter if enabled (new to v1.26).
2041   \glsxtr@inc@linkcount

    As the original definition.
2042   \do@gl:disablehyperinlist
2043   \do@gls@link@checkfirsthyper

    User hook before options are set (new to v1.26):
2044   \glslinkpresetkeys

    Set options.
2045   \setkeys{glslink}{#1}%

    User hook after options are set:
2046   \glslinkpostsetkeys

    Check thevalue and theHvalue before saving (v1.19).
2047   \ifdefempty{\@glsxtr@thevalue}%
2048   {%
2049     \@gls@saveentrycounter
2050   }%
2051   {%
2052     \let\theglsentrycounter\@glsxtr@thevalue
2053     \def\theHglentrycounter{\@glsxtr@theHvalue}%
2054   }%
2055   \@gls@setsort{\glslabel}%

```

Check if the textformat key has been used.

```
2056 \ifx\@glxtr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
2057 \glshasattribute{\glslabel}{textformat}%
2058 {%
2059 \edef\@glxtr@attrval{\glsetattribute{\glslabel}{textformat}}%
2060 \ifcsdef{\@glxtr@attrval}%
2061 {%
2062 \letcs{\@glxtr@textformat}{\@glxtr@attrval}%
2063 }%
2064 {%
2065 \GlossariesExtraWarning{Unknown control sequence name
2066 '\@glxtr@attrval' supplied in textformat attribute
2067 for entry '\glslabel'. Reverting to default \string\glstextformat}%
2068 \let\@glxtr@textformat\glstextformat
2069 }%
2070 }%
2071 {%
2072 \let\@glxtr@textformat\glstextformat
2073 }%
2074 \else
2075 \let\@glxtr@textformat\@glxtr@local@textformat
2076 \fi
```

Do write if it should occur before the link text:

```
2077 \ifglxtrinitwrglossbefore
2078 \do@wrglossary{#2}%
2079 \fi
```

Do the link text:

```
2080 \ifKV@glslink@hyper
2081 \ifglxtr@hyperoutside
2082 \@glslink{\glolinkprefix\glslabel}{\@glxtr@textformat{#3}}%
2083 \else
2084 \@glxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2085 \fi
2086 \else
2087 \ifglxtr@hyperoutside
2088 \glndonohyperlink{\glolinkprefix\glslabel}{\@glxtr@textformat{#3}}%
2089 \else
2090 \@glxtr@textformat{\glndonohyperlink{\glolinkprefix\glslabel}{#3}}%
2091 \fi
2092 \fi
```

Do write if it should occur after the link text:

```
2093 \ifglxtrinitwrglossbefore
2094 \else
2095 \do@wrglossary{#2}%
2096 \fi
```

Restore original value of \glolinkprefix:

```
2097 \let\glolinkprefix\@glstr@org@glolinkprefix
```

As the original definition:

```
2098 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2099 }
```

```
2100 \define@key{glossadd}{thevalue}{\def\@glstr@thevalue{#1}}
```

```
2101 \define@key{glossadd}{theHvalue}{\def\@glstr@theHvalue{#1}}
```

lsaddpresetkeys

```
2102 \newcommand*{\gladdpresetkeys}{}
```

saddpostsetkeys

```
2103 \newcommand*{\gladdpostsetkeys}{}
```

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

```
2104 \renewrobustcmd*{\gladd}[2][\%
2105 \glstrifinmark
2106 {}%
2107 {%
2108 \@gl@adjustmode
2109 \@glstr@record{#1}{#2}{glossadd}%
2110 \glsoifexists{#2}%
2111 {%
2112 \let\@gl@numberformat\@glstr@defaultnumberformat
2113 \edef\@gl@counter{\csname glo@\gl@detoklabel{#2}@counter\endcsname}%
2114 \def\@glstr@thevalue{}%
2115 \def\@glstr@theHvalue{\@glstr@thevalue}%

```

Implement any default settings (before options are set)

```
2116 \gladdpresetkeys
2117 \setkeys{glossadd}{#1}%

```

Implement any default settings (after options are set)

```
2118 \gladdpostsetkeys
2119 \ifdefempty{\@glstr@thevalue}%
2120 {%
2121 \@gl@saveentrycounter
2122 }%
2123 {%
2124 \let\thegl@entrycounter\@glstr@thevalue
2125 \def\theHgl@entrycounter{\@glstr@theHvalue}%
2126 }%

```

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

```
2127 \@gl@setsort{#2}%
2128 \@do@wrglossary{#2}%
2129 }%

```

```

2130 }%
2131 }

```

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

```

2132 \newrobustcmd{\glsaddeach}[2][]{%
2133   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2134 }

```

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

```

2135 \newcommand*{\@glsxtr@field@linkdefs}{%
2136   \let\glsxtrifwasfirstuse\@secondoftwo
2137   \let\glsifplural\@secondoftwo
2138   \let\glschapscase\@firstofthree
2139   \let\glsinsert\@empty
2140 }

```

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

`assignfieldfont`

```

2141 \newcommand*{\glsxtrassignfieldfont}[1]{%
2142   \ifglsentryexists{#1}%
2143   {%
2144     \ifgls hasshort{#1}%
2145     {%
2146       \glssetabbrvfmt{\gls category{#1}}%
2147       \glsifregular{#1}%
2148       {\let\@gls@field@font\glsxtrregularfont}%
2149       {\let\@gls@field@font\@firstofone}%
2150     }%
2151     {%
2152       \glsifnotregular{#1}%
2153       {\let\@gls@field@font\@firstofone}%
2154       {\let\@gls@field@font\glsxtrregularfont}%
2155     }%
2156   }%
2157   {%
2158     \let\@gls@field@font\@gobble
2159   }%
2160 }

```

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

```

2161 \def\@glstext@#1#2[#3]{%
2162   \glsxtrassignfieldfont{#2}%
2163   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2164 }

```

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

```

2165 \def\@GLStext@#1#2[#3]{%

```

```

2166 \glstrassignfieldfont{#2}%
2167 \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2168 {\@gls@field@font{\Glsaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2169 }

```

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

```

2170 \def\@Glstext@#1#2[#3]{%
2171 \glstrassignfieldfont{#2}%
2172 \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2173 {\@gls@field@font{\Glsaccesstext{#2}#3}}%
2174 }

```

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`

```

2175 \newcommand*\glstrchecknohyperfirst[1]{%
2176 \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2177 }

```

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

```

2178 \def\@glsfirst@#1#2[#3]{%
2179 \glstrassignfieldfont{#2}%
2180 \gls@field@link
2181 [\let\glstrifwasfirstuse\@firstoftwo
2182 \glstrchecknohyperfirst{#2}%
2183 ]{#1}{#2}%
2184 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2185 }

```

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

```

2186 \def\@Glsfirst@#1#2[#3]{%
2187 \glstrassignfieldfont{#2}%
2188 \gls@field@link
2189 [\let\glstrifwasfirstuse\@firstoftwo
2190 \let\glscapscase\@secondofthree
2191 \glstrchecknohyperfirst{#2}%
2192 ]%
2193 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2194 }

```

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

```

2195 \def\@GLSfirst@#1#2[#3]{%
2196 \glstrassignfieldfont{#2}%

```

Ensure that \GLSfirst honours the nohyperfirst attribute.

```
2197 \@gls@field@link
2198 [\let\glstrifwasfirstuse\@firstoftwo
2199 \let\glscapscase\@thirdofthree
2200 \glstrchecknohyperfirst{#2}%
2201 ]%
2202 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2203 }
```

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

```
2204 \def\@glsplural@#1#2[#3]{%
2205 \glstrassignfieldfont{#2}%
2206 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2207 {\@gls@field@font{\glsaccessplural{#2}#3}}%
2208 }
```

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

```
2209 \def\@GLsplural@#1#2[#3]{%
2210 \glstrassignfieldfont{#2}%
2211 \@gls@field@link
2212 [\let\glsifplural\@firstoftwo
2213 \let\glscapscase\@secondofthree
2214 ]%
2215 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}#3}}%
2216 }
```

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

```
2217 \def\@GLSplural@#1#2[#3]{%
2218 \glstrassignfieldfont{#2}%
2219 \@gls@field@link
2220 [\let\glsifplural\@firstoftwo
2221 \let\glscapscase\@thirdofthree
2222 ]%
2223 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2224 }
```

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

```
2225 \def\glsfirstplural@#1#2[#3]{%
2226 \glstrassignfieldfont{#2}%
2227 \@gls@field@link
2228 [\let\glstrifwasfirstuse\@firstoftwo
2229 \let\glsifplural\@firstoftwo
2230 \glstrchecknohyperfirst{#2}%
2231 ]%
2232 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2233 }
```

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

```
2234 \def\@Glsfirstplural@#1#2[#3]{%
2235   \glstrassignfieldfont{#2}%
      Ensure that \glfirstplural honours the nohyperfirst attribute.
2236   \@gls@field@link
2237   [\let\glstrifwasfirstuse\@firstoftwo
2238   \let\glstifplural\@firstoftwo
2239   \let\glscapscase\@secondofthree
2240   \glstrchecknohyperfirst{#2}%
2241   ]%
2242   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2243 }
```

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

```
2244 \def\@GLSfirstplural@#1#2[#3]{%
2245   \glstrassignfieldfont{#2}%
      Ensure that \glfirstplural honours the nohyperfirst attribute.
2246   \@gls@field@link
2247   [\let\glstrifwasfirstuse\@firstoftwo
2248   \let\glstifplural\@firstoftwo
2249   \let\glscapscase\@thirdofthree
2250   \glstrchecknohyperfirst{#2}%
2251   ]%
2252   {#1}{#2}%
2253   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2254 }
```

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

```
2255 \def\@glsname@#1#2[#3]{%
2256   \glstrassignfieldfont{#2}%
2257   \@gls@field@link{#1}{#2}{\@gls@field@font{\Glsaccessname{#2}#3}}%
2258 }
```

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

```
2259 \def\@Glsname@#1#2[#3]{%
2260   \glstrassignfieldfont{#2}%
2261   \@gls@field@link
2262   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2263   {\@gls@field@font{\Glsaccessname{#2}#3}}%
2264 }
```

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

```
2265 \def\@GLSname@#1#2[#3]{%
2266   \glstrassignfieldfont{#2}%
2267   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2268   {#1}{#2}%
2269   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2270 }
```



```

\@Glsdesc@
2271 \def\@Glsdesc@#1#2[#3]{%
2272   \glstrassignfieldfont{#2}%
2273   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2274 }

\@GLSdesc@   First letter uppercase version.
2275 \def\@GLSdesc@#1#2[#3]{%
2276   \glstrassignfieldfont{#2}%
2277   \@gls@field@link
2278   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2279   {\@gls@field@font{\@GLSaccessdesc{#2}#3}}%
2280 }

\@GLSdesc@   All uppercase version.
2281 \def\@GLSdesc@#1#2[#3]{%
2282   \glstrassignfieldfont{#2}%
2283   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2284   {#1}{#2}{\@gls@field@font{\@GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2285 }

@Glsdescplural@   No case-changing version.
2286 \def\@Glsdescplural@#1#2[#3]{%
2287   \glstrassignfieldfont{#2}%
2288   \@gls@field@link
2289   [\let\glscapscase\@secondoftwo
2290    \let\glsifplural\@firstoftwo
2291    ]{#1}{#2}{\@gls@field@font{\@glsaccessdescplural{#2}#3}}%
2292 }

@GLSdescplural@   First letter uppercase version.
2293 \def\@GLSdescplural@#1#2[#3]{%
2294   \glstrassignfieldfont{#2}%
2295   \@gls@field@link
2296   [\let\glscapscase\@secondoftwo
2297    \let\glsifplural\@firstoftwo
2298    ]{#1}{#2}{\@gls@field@font{\@GLSaccessdescplural{#2}#3}}%
2299 }

@GLSdescplural@   All uppercase version.
2300 \def\@GLSdesc@#1#2[#3]{%
2301   \glstrassignfieldfont{#2}%
2302   \@gls@field@link
2303   [\let\glscapscase\@thirdoftwo
2304    \let\glsifplural\@firstoftwo
2305    ]%
2306    {#1}{#2}%
2307    {\@gls@field@font{\@GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2308 }

```

```

\@glssymbol@
2309 \def\@glssymbol@#1#2[#3]{%
2310   \glstrassignfieldfont{#2}%
2311   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2312 }

\@Glsymbol@ First letter uppercase version.
2313 \def\@Glsymbol@#1#2[#3]{%
2314   \glstrassignfieldfont{#2}%
2315   \@gls@field@link
2316   [\let\glscapscase\@secondoftwo]%
2317   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2318 }

\@GLSymbol@ All uppercase version.
2319 \def\@GLSymbol@#1#2[#3]{%
2320   \glstrassignfieldfont{#2}%
2321   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2322   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2323 }

lssymbolplural@ No case-changing version.
2324 \def\@lssymbolplural@#1#2[#3]{%
2325   \glstrassignfieldfont{#2}%
2326   \@gls@field@link
2327   [\let\glscapscase\@secondoftwo
2328   \let\glsifplural\@firstoftwo
2329   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2330 }

lssymbolplural@ First letter uppercase version.
2331 \def\@lssymbolplural@#1#2[#3]{%
2332   \glstrassignfieldfont{#2}%
2333   \@gls@field@link
2334   [\let\glscapscase\@secondoftwo
2335   \let\glsifplural\@firstoftwo
2336   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2337 }

LSsymbolplural@ All uppercase version.
2338 \def\@LSsymbol@#1#2[#3]{%
2339   \glstrassignfieldfont{#2}%
2340   \@gls@field@link
2341   [\let\glscapscase\@thirdoftwo
2342   \let\glsifplural\@firstoftwo
2343   ]%
2344   {#1}{#2}%
2345   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2346 }

```

\@Glsuseri@ First letter uppercase version.

```
2347 \def\@Glsuseri@#1#2[#3]{%
2348   \glstrassignfieldfont{#2}%
2349   \@gls@field@link
2350   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2351   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2352 }
```

\@GLSuseri@ All uppercase version.

```
2353 \def\@GLSuseri@#1#2[#3]{%
2354   \glstrassignfieldfont{#2}%
2355   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2356   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
2357 }
```

\@Glsuserii@ First letter uppercase version.

```
2358 \def\@Glsuserii@#1#2[#3]{%
2359   \glstrassignfieldfont{#2}%
2360   \@gls@field@link
2361   [\let\glscapscase\@secondoftwo]%
2362   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2363 }
```

\@GLSuserii@ All uppercase version.

```
2364 \def\@GLSuserii@#1#2[#3]{%
2365   \glstrassignfieldfont{#2}%
2366   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2367   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2368 }
```

\@Glsuseriii@ First letter uppercase version.

```
2369 \def\@Glsuseriii@#1#2[#3]{%
2370   \glstrassignfieldfont{#2}%
2371   \@gls@field@link
2372   [\let\glscapscase\@secondoftwo]%
2373   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
2374 }
```

\@GLSuseriii@ All uppercase version.

```
2375 \def\@GLSuseriii@#1#2[#3]{%
2376   \glstrassignfieldfont{#2}%
2377   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2378   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2379 }
```

\@Glsuseriv@ First letter uppercase version.

```
2380 \def\@Glsuseriv@#1#2[#3]{%
2381   \glstrassignfieldfont{#2}%
2382 }
```

```

2382 \@gls@field@link
2383 [\let\glscapscase\@secondoftwo]%
2384 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
2385 }

```

\@GLSuseriv@ All uppercase version.

```

2386 \def\@GLSuseriv@#1#2[#3]{%
2387 \glstrassignfieldfont{#2}%
2388 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2389 {#1}{#2}%
2390 {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2391 }

```

\@Glsuserv@ First letter uppercase version.

```

2392 \def\@Glsuserv@#1#2[#3]{%
2393 \glstrassignfieldfont{#2}%
2394 \@gls@field@link
2395 [\let\glscapscase\@secondoftwo]%
2396 {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
2397 }

```

\@GLSuserv@ All uppercase version.

```

2398 \def\@GLSuserv@#1#2[#3]{%
2399 \glstrassignfieldfont{#2}%
2400 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2401 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2402 }

```

\@Glsuservi@ First letter uppercase version.

```

2403 \def\@Glsuservi@#1#2[#3]{%
2404 \glstrassignfieldfont{#2}%
2405 \@gls@field@link
2406 [\let\glscapscase\@secondoftwo]%
2407 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2408 }

```

\@GLSuservi@ All uppercase version.

```

2409 \def\@GLSuservi@#1#2[#3]{%
2410 \glstrassignfieldfont{#2}%
2411 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2412 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2413 }

```

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

\@acrshort No case change.

```

2414 \def\@acrshort#1#2[#3]{%

```

```

2415 \glsdoifexists{#2}%
2416 {%
2417   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2418   \let\glxtrifwasfirstuse\@secondoftwo
2419   \let\gl@sifplural\@secondoftwo
2420   \let\glscapscase\@firstofthree
2421   \let\gl$insert\@empty
2422   \def\glscustomtext{%
2423     \acronymfont{\gl@saccessshort{#2}}#3%
2424   }%
2425   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2426 }%
2427 \glspostlinkhook
2428 }

```

\@Acrshort First letter uppercase.

```

2429 \def\@Acrshort#1#2[#3]{%
2430   \glsdoifexists{#2}%
2431   {%
2432     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2433     \let\glxtrifwasfirstuse\@secondoftwo
2434     \let\gl@sifplural\@secondoftwo
2435     \let\glscapscase\@secondofthree
2436     \let\gl$insert\@empty
2437     \def\glscustomtext{%
2438       \acronymfont{\Glsaccessshort{#2}}#3%
2439     }%
2440     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2441   }%
2442   \glspostlinkhook
2443 }

```

\@ACRshort All uppercase.

```

2444 \def\@ACRshort#1#2[#3]{%
2445   \glsdoifexists{#2}%
2446   {%
2447     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2448     \let\glxtrifwasfirstuse\@secondoftwo
2449     \let\gl@sifplural\@secondoftwo
2450     \let\glscapscase\@thirdofthree
2451     \let\gl$insert\@empty
2452     \def\glscustomtext{%
2453       \mfirstucMakeUppercase{\acronymfont{\gl@saccessshort{#2}}#3}%
2454     }%
2455     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2456   }%
2457   \glspostlinkhook
2458 }

```

\@acrshortpl No case change.

```
2459 \def\@acrshortpl#1#2[#3]{%
2460   \glsdoifexists{#2}%
2461   {%
2462     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2463     \let\glxtrifwasfirstuse\@secondoftwo
2464     \let\gl@sifplural\@firstoftwo
2465     \let\glscapscase\@firstofthree
2466     \let\gl$insert\@empty
2467     \def\glscustomtext{%
2468       \acronymfont{\gl@saccesssshortpl{#2}}#3%
2469     }%
2470     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2471   }%
2472   \glspostlinkhook
2473 }
```

\@Acrshortpl First letter uppercase.

```
2474 \def\@Acrshortpl#1#2[#3]{%
2475   \glsdoifexists{#2}%
2476   {%
2477     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2478     \let\glxtrifwasfirstuse\@secondoftwo
2479     \let\gl@sifplural\@firstoftwo
2480     \let\glscapscase\@secondofthree
2481     \let\gl$insert\@empty
2482     \def\glscustomtext{%
2483       \acronymfont{\Glsaccesssshortpl{#2}}#3%
2484     }%
2485     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2486   }%
2487   \glspostlinkhook
2488 }
```

\@ACRshortpl All uppercase.

```
2489 \def\@ACRshortpl#1#2[#3]{%
2490   \glsdoifexists{#2}%
2491   {%
2492     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2493     \let\glxtrifwasfirstuse\@secondoftwo
2494     \let\gl@sifplural\@firstoftwo
2495     \let\glscapscase\@thirdofthree
2496     \let\gl$insert\@empty
2497     \def\glscustomtext{%
2498       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshortpl{#2}}#3}%
2499     }%
2500     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2501   }%
2502   \glspostlinkhook
```

2503 }

\@acrlong No case change.

```
2504 \def\@acrlong#1#2[#3]{%
2505   \glsdoifexists{#2}%
2506   {%
2507     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2508     \let\glxtrifwasfirstuse\@secondoftwo
2509     \let\gl@sifplural\@secondoftwo
2510     \let\glscapscase\@firstofthree
2511     \let\gl$insert\@empty
2512     \def\glscustomtext{%
2513       \acronymfont{\gl@saccesslong{#2}}#3%
2514     }%
2515     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2516   }%
2517   \glspostlinkhook
2518 }
```

\@Acrlong First letter uppercase.

```
2519 \def\@Acrlong#1#2[#3]{%
2520   \glsdoifexists{#2}%
2521   {%
2522     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2523     \let\glxtrifwasfirstuse\@secondoftwo
2524     \let\gl@sifplural\@secondoftwo
2525     \let\glscapscase\@secondofthree
2526     \let\gl$insert\@empty
2527     \def\glscustomtext{%
2528       \acronymfont{\Glsaccesslong{#2}}#3%
2529     }%
2530     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2531   }%
2532   \glspostlinkhook
2533 }
```

\@ACRlong All uppercase.

```
2534 \def\@ACRlong#1#2[#3]{%
2535   \glsdoifexists{#2}%
2536   {%
2537     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2538     \let\glxtrifwasfirstuse\@secondoftwo
2539     \let\gl@sifplural\@secondoftwo
2540     \let\glscapscase\@thirdofthree
2541     \let\gl$insert\@empty
2542     \def\glscustomtext{%
2543       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslong{#2}}#3}%
2544     }%
2545     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2546   }
```

```

2546 }%
2547 \glspostlinkhook
2548 }

```

\@acrlongpl No case change.

```

2549 \def\@acrlongpl#1#2[#3]{%
2550 \glsdoidexists{#2}%
2551 {%
2552 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2553 \let\glxtrifwasfirstuse\@secondoftwo
2554 \let\gl@sifplural\@firstoftwo
2555 \let\glscapscase\@firstofthree
2556 \let\gl$insert\@empty
2557 \def\glscustomtext{%
2558 \acronymfont{\glaccesslongpl{#2}}#3%
2559 }%
2560 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2561 }%
2562 \glspostlinkhook
2563 }

```

\@Acrlongpl First letter uppercase.

```

2564 \def\@Acrlongpl#1#2[#3]{%
2565 \glsdoidexists{#2}%
2566 {%
2567 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2568 \let\glxtrifwasfirstuse\@secondoftwo
2569 \let\gl@sifplural\@firstoftwo
2570 \let\glscapscase\@secondofthree
2571 \let\gl$insert\@empty
2572 \def\glscustomtext{%
2573 \acronymfont{\Glsaccesslongpl{#2}}#3%
2574 }%
2575 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2576 }%
2577 \glspostlinkhook
2578 }

```

\@ACRlongpl All uppercase.

```

2579 \def\@ACRlongpl#1#2[#3]{%
2580 \glsdoidexists{#2}%
2581 {%
2582 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2583 \let\glxtrifwasfirstuse\@secondoftwo
2584 \let\gl@sifplural\@firstoftwo
2585 \let\glscapscase\@thirdofthree
2586 \let\gl$insert\@empty
2587 \def\glscustomtext{%
2588 \mfirstucMakeUppercase{\acronymfont{\glaccesslongpl{#2}}#3}%

```



```

2589 }%
2590 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2591 }%
2592 \glspostlinkhook
2593 }

```

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

\@glsaddkey

```

2594 \renewcommand*{\@glsaddkey}[7]{%
2595   \key@ifundefined{glossentry}{#1}%
2596   {%
2597     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2598     \appto\@gls@keymap{,{#1}{#1}}%
2599     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2600     \appto\@newglossaryentryposthook{%
2601       \letcs{\@glo@tmp}{@glo@#1}%
2602       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2603     }%
2604     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2605     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2606   \ifcsdef{@gls@user@#1@}%
2607   {%
2608     \PackageError{glossaries}%
2609     {Can't define '\string#5' as helper command
2610     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2611     exists}%
2612     {}%
2613   }%
2614   {%
2615     \expandafter\newcommand\expandafter*\expandafter
2616     {\csname @gls@user@#1\endcsname}[2][ ]{%
2617       \new@ifnextchar[%
2618         {\csuse{@gls@user@#1@}{##1}{##2}}%
2619         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2620     \csdef{@gls@user@#1@}##1##2[##3]{%
2621       \@gls@field@link{##1}{##2}{#3{##2}##3}%
2622     }%
2623     \newrobustcmd*{#5}{%
2624       \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2625   }%

```

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

```

2626   \ifcsdef{@Gls@user@#1@}%
2627   {%
2628     \PackageError{glossaries}%
2629     {Can't define '\string#6' as helper command

```

```

2630      '\expandafter\string\csname @Gls@user@#1@endcsname' already
2631      exists}%
2632  {}%
2633 }%
2634 {%
2635   \expandafter\newcommand\expandafter*\expandafter
2636   {\csname @Gls@user@#1@endcsname}[2][\%
2637     \new@ifnextchar[%
2638       {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2639     {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
2640   \csdef{@Gls@user@#1@}##1##2[##3]{%
2641     \@gls@field@link[\let\glscapscase\@secondofthree]%
2642     {##1}{##2}{#4{##2}##3}%
2643   }%
2644   \newrobustcmd*{#6}{%
2645     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@endcsname}%
2646   }%

```

Finally the all caps version (modified):

```

2647   \ifcsdef{@GLS@user@#1@}%
2648   {%
2649     \PackageError{glossaries}%
2650     {Can't define '\string#7' as helper command
2651     '\expandafter\string\csname @GLS@user@#1@endcsname' already
2652     exists}%
2653   {}%
2654 }%
2655 {%
2656   \expandafter\newcommand\expandafter*\expandafter
2657   {\csname @GLS@user@#1@endcsname}[2][\%
2658     \new@ifnextchar[%
2659       {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2660       {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2661   \csdef{@GLS@user@#1@}##1##2[##3]{%
2662     \@gls@field@link[\let\glscapscase\@thirdofthree]%
2663     {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2664   }%
2665   \newrobustcmd*{#7}{%
2666     \expandafter\@gls@hyp@opt\csname @GLS@user@#1@endcsname}%
2667   }%
2668 }%
2669 {%
2670   \PackageError{glossaries-extra}{Key '#1' already exists}{}%
2671 }%
2672 }

```

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

```

2673 \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.

```
2674 \let\@glstr@org@checkfirsthyper\@gls@link@checkfirsthyper
2675 \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 \glstrifwasfirstuse to \@secondoftwo (which is done in \@glstr@field@linkdefs).

```
2676 \ifglsused{\glslabel}%
2677 {\let\glstrifwasfirstuse\@secondoftwo}
2678 {\let\glstrifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
2679 \edef\glscategorylabel{\glscategory{\glslabel}}%
2680 \ifglsused{\glslabel}%
2681 {%
2682   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
2683   {\KV@glslink@hyperfalse}{}}%
2684 }%
2685 {%
2686   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
2687   {\KV@glslink@hyperfalse}{}}%
2688 }%
2689 \glslinkcheckfirsthyperhook
2690 }
```

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.

```
2691 \ifdef\do@glstdisablehyperinlist
2692 {%
2693   \let\@glstr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
2694   \renewcommand*{\do@glstdisablehyperinlist}{%
2695     \@glstr@do@glstdisablehyperinlist
2696     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}}%
2697   }
2698 }
2699 }
```

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

```
2700 \define@boolkey{glslink}{noindex}[true]{}
2701 \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

```
2702 \ifdef\@gls@setdefault@glslink@opts
```

```

2703 {
2704   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2705     \KV@glslink@noindexfalse
2706     \@glxtrsetaliasnoindex
2707   }
2708 }
2709 {
    Not defined so prepend it to \do@glstdisablehyperinlist to achieve the same effect.
2710   \newcommand*{\@gls@setdefault@glslink@opts}{%
2711     \KV@glslink@noindexfalse
2712     \@glxtrsetaliasnoindex
2713   }
2714   \preto\do@glstdisablehyperinlist{\@gls@setdefault@glslink@opts}
2715 }

```

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.)

```

2716 \providecommand*{\glxtrsetaliasnoindex}{%
2717   \KV@glslink@noindextrue
2718 }

```

setaliasnoindex

```

2719 \newcommand*{\@glxtrsetaliasnoindex}{%
2720   \glxtrifhasfield{alias}{\glslabel}%
2721   {%
2722     \let\glxtrindexaliased\@glxtrindexaliased
2723     \glxtrsetaliasnoindex
2724     \let\glxtrindexaliased\@no@glxtrindexaliased
2725   }%
2726   {}%
2727 }

```

xtrindexaliased

```

2728 \newcommand{\@glxtrindexaliased}{%
2729   \ifKV@glslink@noindex
2730   \else
2731     \begingroup
2732     \let\@glsnumberformat\@glxtr@defaultnumberformat
2733     \edef\@gls@counter{\csname glo@\glstdetoklabel{\glslabel}@counter\endcsname}%
2734     \glxtr@saveentrycounter
2735     \@@do@wrglossary{\glxtralias{\glslabel}}%
2736     \endgroup
2737   \fi
2738 }

```

xtrindexaliased

```

2739 \newcommand{\@no@glxtrindexaliased}{%

```

```

2740 \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2741 not permitted outside definition of \string\glxtrsetaliasnoindex}%
2742 {}%
2743 }

```

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

```

2744 \let\glxtrindexaliased\@no\glxtrindexaliased

```

`\setdefaultglsOpts` Set the default options for `\glslink` etc.

```

2745 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2746 \renewcommand*{\@gls@setdefault@glslink@opts}{%
2747 \setkeys{glslink}{#1}%
2748 \@glxtrsetaliasnoindex
2749 }%
2750 }

```

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

```

2751 \newcommand*{\glxtrifindexing}[2]{%
2752 \ifKV@glslink@noindex #2\else #1\fi
2753 }

```

`\glswriteentry` Redefine to test for `indexonlyfirst` category attribute.

```

2754 \renewcommand*{\glswriteentry}[2]{%
2755 \glxtrifindexing
2756 {%
2757 \ifglsindexonlyfirst
2758 \ifglsused{#1}
2759 {\glxtrdoautoindexname{#1}{dualindex}}%
2760 {#2}%
2761 \else
2762 \glsifattribute{#1}{indexonlyfirst}{true}%
2763 {\ifglsused{#1}
2764 {\glxtrdoautoindexname{#1}{dualindex}}%
2765 {#2}}%
2766 {#2}%
2767 \fi
2768 }%
2769 {}%
2770 }

```

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

```

2771 \appto\do@wrglossary{\@glxtr@do@wrindex
2772 \glxtrdowrglossaryhook{\@gls@label}%
2773 }

```

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

Similarly for the “noidx” version:

s@noidxglossary

```
2774 \appto\gls@noidxglossary{\@glstrdo@wrindex
2775 \glstrdowrglossaryhook{\@gls@label}}%
2776 }
```

xtrdo@wrindex

```
2777 \newcommand*{\@glstrdo@wrindex}{%
2778 \glstrdoautoindexname{\@gls@label}{dualindex}}%
2779 }
```

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.)

```
2780 \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.

```
2781 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2782 \let\glslinkvar\@firstofthree
2783 \let\@gls@hyp@opt@cs#1\relax
2784 \@ifstar{\s@gls@hyp@opt}%
2785 {\@ifnextchar+%
2786 {\@firstoftwo{\p@gls@hyp@opt}}%
2787 {%
2788 \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2789 {\@firstoftwo{\@alt@gls@hyp@opt}}%
2790 {\#1}%
2791 }%
2792 }%
2793 }
```

alt@gls@hyp@opt User version

```
2794 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
2795 \let\glslinkvar\@firstofthree
2796 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
2797 \newcommand*{\@gls@alt@hyp@opt@char}{}
```

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

```
2798 \newcommand*{\@gls@alt@hyp@opt@keys}{}
```

rSetAltModifier

```
2799 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2800 \let\@gls@hyp@opt\@gls@alt@hyp@opt
2801 \def\@gls@alt@hyp@opt@char{\#1}%
2802 \def\@gls@alt@hyp@opt@keys{\#2}%
2803 }
```

org@dohyperlink

```
2804 \let\glstr@org@dohyperlink\glsdohyperlink
```

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

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

```
2805 \ifdef\glsnavhyperlink
```

```
2806 {
```

```
2807   \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
```

```
2808     \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
2809   {%
```

```
2810     \let\glsdohyperlink\glstr@org@dohyperlink
```

```
2811     \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

```
2812   }%
```

```
2813 }%
```

```
2814 }
```

```
2815 {}
```

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

```
2816 \renewcommand*{\glsdohyperlink}[2]{%
```

```
2817   \glsattribute{\glslabel}{targeturl}%
```

```
2818   {%
```

```
2819     \glsattribute{\glslabel}{targetname}%
```

```
2820     {%
```

```
2821       \glsattribute{\glslabel}{targetcategory}%
```

```
2822     {%
```

```
2823       \hyperref{\glsattribute{\glslabel}{targeturl}}{%
```

```
2824         {\glsattribute{\glslabel}{targetcategory}}{%
```

```
2825         {\glsattribute{\glslabel}{targetname}}{%
```

```
2826         {\glstrprotectlinks#2}}{%
```

```
2827       }%
```

```
2828     {%
```

```
2829       \hyperref{\glsattribute{\glslabel}{targeturl}}{%
```

```
2830         }{%
```

```
2831         {\glsattribute{\glslabel}{targetname}}{%
```

```
2832         {\glstrprotectlinks#2}}{%
```

```
2833       }%
```

```
2834     }%
```

```
2835   {%
```

```

2836     \href{\glsgetattribute{\glslabel}{targeturl}}{%
2837         {\glsxtrprotectlinks#2}}%
2838     }%
2839 }%
2840 {%

```

Check for alias.

```

2841     \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2842     \ifvoid\gloaliaslabel
2843     {%
2844         \glsxtrhyperlink{#1}{\glsxtrprotectlinks#2}}%
2845     }%
2846     {%

```

Redirect link to the alias target.

```

2847     \glsxtrhyperlink
2848     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2849     {\glsxtrprotectlinks#2}}%
2850     }%
2851 }%
2852 }

```

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

```

2853 \ifdef\@glsshowtarget
2854 {
2855     \newcommand{\glsxtrhyperlink}[2]{%
2856         \@glsshowtarget{#1}%
2857         \hyperlink{#1}{#2}%
2858     }%
2859 }
2860 {
2861     \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
2862 }

```

`\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.

```

2863 \renewrobustcmd*{\glsdohyperlink}[2][\glsentrytext{\@glo@label}]{%
2864     \glsdoifexists{#2}%
2865     {%
2866         \def\@glo@label{#2}%
2867         {\edef\glslabel{#2}%
2868             \@glslink{\glolinkprefix\glslabel}{#1}}%
2869     }%
2870 }

```

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


```

2871 \renewcommand{\glsdisablehyper}{%
2872   \KV@glslink@hyperfalse
2873   \def\@glslink{\glsdonohyperlink}%
2874   \let\@glstarget\@secondoftwo
2875 }

```

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

```

2876 \renewcommand{\glsenablehyper}{%
2877   \KV@glslink@hypertrue
2878   \def\@glslink{\glsdohyperlink}%
2879   \def\@glstarget{\glsdohypertarget}%
2880 }

```

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

```

2881 \def\glsdonohyperlink#1#2{{\glstrprotectlinks #2}}

```

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

```

2882 \ifcsundef{hyperlink}%
2883 {%
2884   \def\@glslink{\glsdonohyperlink}
2885 }%
2886 {%
2887   \def\@glslink{\glsdohyperlink}
2888 }

```

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

```

2889 \newcommand*{\glstrprotectlinks}{%
2890   \KV@glslink@hyperfalse
2891   \KV@glslink@noindextrue
2892   \let\@gls\@glstr@p@text@
2893   \let\@Gls\@Gls@p@text@
2894   \let\@GLS\@GLS@p@text@
2895   \let\@glspl\@glstr@p@plural@
2896   \let\@Glspl\@Gls@p@plural@
2897   \let\@GLSpl\@GLS@p@plural@
2898   \let\@glstrshort\@glstr@p@short@
2899   \let\@Glsrshort\@Gls@p@short@
2900   \let\@GLSrshort\@GLS@p@short@
2901   \let\@glstrlong\@glstr@p@long@
2902   \let\@Glsrlong\@Gls@p@long@
2903   \let\@GLSrlong\@GLS@p@long@
2904   \let\@glstrshortpl\@glstr@p@shortpl@
2905   \let\@Glsrshortpl\@Gls@p@shortpl@

```

```

2906 \let\@GLSxtrshortpl\@GLSxtrp@shortpl@
2907 \let\@glxtrlongpl\@glxtrp@longpl@
2908 \let\@Glsxtrlongpl\@Glsxtrp@longpl@
2909 \let\@GLSxtrlongpl\@GLSxtrp@longpl@
2910 \let\@acrshort\@glxtrp@acrshort@
2911 \let\@Acrshort\@Glsxtrp@acrshort@
2912 \let\@ACRshort\@GLSxtrp@acrshort@
2913 \let\@acrshortpl\@glxtrp@acrshortpl@
2914 \let\@Acrshortpl\@Glsxtrp@acrshortpl@
2915 \let\@ACRshortpl\@GLSxtrp@acrshortpl@
2916 \let\@acrlong\@glxtrp@acrlong@
2917 \let\@Acrlong\@Glsxtrp@acrlong@
2918 \let\@ACRlong\@GLSxtrp@acrlong@
2919 \let\@acrlongpl\@glxtrp@acrlongpl@
2920 \let\@Acrlongpl\@Glsxtrp@acrlongpl@
2921 \let\@ACRlongpl\@GLSxtrp@acrlongpl@
2922 }

```

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

@glxtrp@text@

```
2923 \def\@glxtrp@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtrp@text@

```
2924 \def\@Glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtrp@text@

```
2925 \def\@GLSxtrp@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtrp@plural@

```
2926 \def\@lsxtrp@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lSxtrp@plural@

```
2927 \def\@lSxtrp@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtrp@plural@

```
2928 \def\@LSxtrp@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtrp@short@

```

2929 \def\@glxtrp@short@#1#2[#3]{%
2930 {%
2931   \glssetabbrvfmt{\glscategory{#2}}%
2932   \glsabbrvfont{\glsentryshort{#2}}#3%
2933 }%
2934 }

```

Glsxtrp@short@

```
2935 \def\@Glsxtrp@short@#1#2[#3]{%
```

```

2936 {%
2937   \glssetabbrvfmt{\glscategory{#2}}%
2938   \glsabbrvfont{\Glsentryshort{#2}}#3%
2939 }%
2940 }

```

GLSxtr@p@short@

```

2941 \def\@GLSxtr@p@short@#1#2[#3]{%
2942   {%
2943     \glssetabbrvfmt{\glscategory{#2}}%
2944     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
2945   }%
2946 }

```

sxtr@p@shortpl@

```

2947 \def\@glxsxtr@p@shortpl@#1#2[#3]{%
2948   {%
2949     \glssetabbrvfmt{\glscategory{#2}}%
2950     \glsabbrvfont{\glsentryshortpl{#2}}#3%
2951   }%
2952 }

```

sxtr@p@shortpl@

```

2953 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2954   {%
2955     \glssetabbrvfmt{\glscategory{#2}}%
2956     \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2957   }%
2958 }

```

Sxtr@p@shortpl@

```

2959 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2960   {%
2961     \glssetabbrvfmt{\glscategory{#2}}%
2962     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
2963   }%
2964 }

```

@glxsxtr@p@long@

```

2965 \def\@glxsxtr@p@long@#1#2[#3]{\{\glsentrylong{#2}#3}\}

```

@Glsxtr@p@long@

```

2966 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}\}

```

@GLSxtr@p@long@

```

2967 \def\@GLSxtr@p@long@#1#2[#3]{%
2968   {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}

```

lsxtr@p@longpl@
2969 \def\@glxtr@p@longpl@#1#2[#3]{\glentrylongpl{#2}#3}}

lsxtr@p@longpl@
2970 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glentrylongpl{#2}#3}}

LSxtr@p@longpl@
2971 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2972 {\mfirstucMakeUppercase{\glslongfont{\glentrylongpl{#2}#3}}}

xtr@p@acrshort@
2973 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\glentryshort{#2}#3}}

xtr@p@acrshort@
2974 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glentryshort{#2}#3}}

xtr@p@acrshort@
2975 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2976 {\mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}#3}}}

r@p@acrshortpl@
2977 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glentryshortpl{#2}#3}}

r@p@acrshortpl@
2978 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glentryshortpl{#2}#3}}

r@p@acrshortpl@
2979 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2980 {\mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}#3}}}

sxtr@p@acrlong@
2981 \def\@glxtr@p@acrlong@#1#2[#3]{\glentrylong{#2}#3}}

sxtr@p@acrlong@
2982 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glentrylong{#2}#3}}

Sxtr@p@acrlong@
2983 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2984 {\mfirstucMakeUppercase{\glentrylong{#2}#3}}}

tr@p@acrlongpl@
2985 \def\@glxtr@p@acrlongpl@#1#2[#3]{\glentrylongpl{#2}#3}}

tr@p@acrlongpl@
2986 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glentrylongpl{#2}#3}}

tr@p@acrlongpl@

```
2987 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2988 {\mfirstucMakeUppercase{\glstrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glxtrp@opt

```
2989 \newcommand*{\@glxtrp@opt}{hyper=false,noindex}
```

\glxtrsetpopts Used in glossary to switch hyperlinks on for the \@glxtrp type of commands.

```
2990 \newcommand*{\glxtrsetpopts}[1]{%
2991 \renewcommand*{\@glxtrp@opt}{#1}%
2992 }
```

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

```
2993 \newcommand*{\glossxtrsetpopts}{%
2994 \glxtrsetpopts{noindex}%
2995 }
```

\@@glxtrp

```
2996 \newrobustcmd*{\@@glxtrp}[2]{%
```

Add scope.

```
2997 {%
2998 \let\glspostlinkhook\relax
2999 \csname#1\expandafter\endcsname\expandafter[\@glxtrp@opt]{#2}[]%
3000 }%
3001 }
```

\@glxtrp

```
3002 \newrobustcmd*{\@glxtrp}[2]{%
3003 \ifcsdef{gls#1}%
3004 {%
3005 \@glxtrp{gls#1}{#2}%
3006 }%
3007 {%
3008 \ifcsdef{glsxtr#1}%
3009 {%
3010 \@glxtrp{glsxtr#1}{#2}%
3011 }%
3012 {%
3013 \PackageError{glossaries-extra}{‘#1’ not recognised by
3014 \string\glxtrp}{}%
3015 }%
3016 }%
3017 }
```

\@Glsxtrp

```
3018 \newrobustcmd*{\@Glsxtrp}[2]{%
```

```

3019 \ifcsdef{Gls#1}%
3020 {%
3021   \@@glsxtrp{Gls#1}{#2}%
3022 }%
3023 {%
3024   \ifcsdef{Glsxtr#1}%
3025   {%
3026     \@@glsxtrp{Glsxtr#1}{#2}%
3027   }%
3028   {%
3029     \PackageError{glossaries-extra}{‘#1’ not recognised by
3030     \string\Glsxtrp}{}%
3031   }%
3032 }%
3033 }

```

\@GLSxtrp

```

3034 \newrobustcmd*{\@GLSxtrp}[2]{%
3035   \ifcsdef{GLS#1}%
3036   {%
3037     \@@glsxtrp{GLS#1}{#2}%
3038   }%
3039   {%
3040     \ifcsdef{GLSxtr#1}%
3041     {%
3042       \@@glsxtrp{GLSxtr#1}{#2}%
3043     }%
3044     {%
3045       \PackageError{glossaries-extra}{‘#1’ not recognised by
3046       \string\GLSxtrp}{}%
3047     }%
3048   }%
3049 }

```

\glsxtr@entry@p

```

3050 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3051   \glsifattribute{#1}{headuc}{true}%
3052   {%
3053     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3054   }%
3055   {%
3056     \@gls@entry@field{#1}{#2}%
3057   }%
3058 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3059 \ifdef\texorpdfstring
3060 {
3061   \newcommand{\glsxtrp}[2]{%

```

```

3062 \protect\NoCaseChange
3063 {%
3064 \protect\texorpdfstring
3065 {%
3066 \protect\glstrifinmark
3067 {%
3068 \ifcsdef{glstrhead#1}%
3069 {%
3070 {\protect\csuse{glstrhead#1}{#2}}%
3071 }%
3072 {%
3073 \glstr@headentry@p{#2}{#1}%
3074 }%
3075 }%
3076 {%
3077 \@glstrp{#1}{#2}%
3078 }%
3079 }%
3080 {%
3081 \protect\@glstr@entry@field{#2}{#1}%
3082 }%
3083 }%
3084 }
3085 }
3086 {
3087 \newcommand{\glstrp}[2]{%
3088 \protect\NoCaseChange
3089 {%
3090 \protect\glstrifinmark
3091 {%
3092 \ifcsdef{glstrhead#1}%
3093 {%
3094 {\protect\csuse{glstrhead#1}}%
3095 }%
3096 {%
3097 \glstr@headentry@p{#2}{#1}%
3098 }%
3099 }%
3100 {%
3101 \@glstrp{#1}{#2}%
3102 }%
3103 }%
3104 }
3105 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3106 \newcommand*{\glsp}{\glstrp{short}}

```

```

\glspt
3107 \newcommand*{\glspt}{\glsxtrp{text}}

\Glsxtrp As above but use first letter upper case (but not for the bookmarks, which can't process
\uppercase).
3108 \ifdef\texorpdfstring
3109 {
3110   \newcommand{\Glsxtrp}[2]{%
3111     \protect\NoCaseChange
3112     {%
3113       \protect\texorpdfstring
3114       {%
3115         \protect\glsxtrifinmark
3116         {%
3117           \ifcsdef{Glsxtrhead#1}%
3118           {%
3119             {\protect\csuse{Glsxtrhead#1}{#2}}%
3120           }%
3121           {%
3122             \protect\@Gls@entry@field{#2}{#1}%
3123           }%
3124         }%
3125         {%
3126           \@Glsxtrp{#1}{#2}%
3127         }%
3128       }%
3129       {%
3130         \protect\@gls@entry@field{#2}{#1}%
3131       }%
3132     }%
3133   }
3134 }
3135 {
3136   \newcommand{\Glsxtrp}[2]{%
3137     \protect\NoCaseChange
3138     {%
3139       \protect\glsxtrifinmark
3140       {%
3141         \ifcsdef{Glsxtrhead#1}%
3142         {%
3143           {\protect\csuse{Glsxtrhead#1}}%
3144         }%
3145         {%
3146           \protect\@Gls@entry@field{#2}{#1}%
3147         }%
3148       }%
3149       {%
3150         \@Glsxtrp{#1}{#2}%
3151       }%

```



```

3152     }%
3153   }
3154 }

```

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

```

3155 \ifdef\teorpdfstring
3156 {
3157   \newcommand{\GLSxtrp}[2]{%
3158     \protect\NoCaseChange
3159     {%
3160       \protect\teorpdfstring
3161       {%
3162         \protect\glxtrifinmark
3163         {%
3164           \ifcsdef{GLSxtr#1}%
3165           {%
3166             {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3167           }%
3168           {%
3169             \protect\mfirstucMakeUppercase
3170             {%
3171               \protect\@gls@entry@field{#2}{#1}%
3172             }%
3173           }%
3174         }%
3175       }%
3176       \@GLSxtrp{#1}{#2}%
3177     }%
3178   }%
3179   {%
3180     \protect\@gls@entry@field{#2}{#1}%
3181   }%
3182 }%
3183 }
3184 }
3185 {
3186   \newcommand{\GLSxtrp}[2]{%
3187     \protect\NoCaseChange
3188     {%
3189       \protect\glxtrifinmark
3190       {%
3191         \ifcsdef{GLSxtr#1}%
3192         {%
3193           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3194         }%
3195         {%
3196           \protect\mfirstucMakeUppercase
3197           {%
3198             \protect\@gls@entry@field{#2}{#1}%

```

```

3199         }%
3200     }%
3201 }%
3202 {%
3203     \@GLSxtrp{#1}{#2}%
3204 }%
3205 }%
3206 }
3207 }

```

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 switched off by redefining \@glsunset and then switched back on again by changing the definition back to \@glsxtr@unset.

```

\@glsxtr@unset  Global unset.
3208 \newcommand*{\@glsxtr@unset}[1]{%
3209     \@glsunset{#1}%
3210     \glsxtrpostunset{#1}%
3211 }%

\@glsunset  Global unset.
3212 \let\@glsunset\@glsxtr@unset

glsxtrpostunset
3213 \newcommand*{\glsxtrpostunset}[1]{%

```

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

```

tUnsetBuffering
3214 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3215     \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3216 }

tUnsetBuffering  Unstarred version doesn't check for duplicates.
3217 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3218     \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer

```

```

3219 \def\@glxtr@unset@buffer{}%
3220 \let\@glsunset\@glxtrbuffer@unset
3221 }

```

tUnsetBuffering Starred version checks for duplicates.

```

3222 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3223 \let\@glxtr@org@unset@buffer\@glxtr@unset@buffer
3224 \def\@glxtr@unset@buffer{}%
3225 \let\@glsunset\@glxtrbuffer@nodup@unset
3226 }

```

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

```

3227 \newcommand*{\@glxtrbuffer@unset}[1]{%
3228 \listxadd\@glxtr@unset@buffer{#1}%
3229 }

```

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.)

```

3230 \newcommand*{\@glxtrbuffer@nodup@unset}[1]{%
3231 \expandafter\ifinlist\expandafter{#1}{\@glxtr@unset@buffer}{}%
3232 {\listxadd\@glxtr@unset@buffer{#1}}%
3233 }

```

pUnsetBuffering

```

3234 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3235 \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3236 }

```

pUnsetBuffering Unstarred form (global unset).

```

3237 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3238 \let\@glsunset\@glxtr@unset
3239 \forlistloop\@glsunset\@glxtr@unset@buffer
3240 \let\@glxtr@unset@buffer\@glxtr@org@unset@buffer
3241 }

```

pUnsetBuffering Starred form (local unset).

```

3242 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3243 \forlistloop\@glsllocalunset\@glxtr@unset@buffer
3244 \let\@glsunset\@glxtr@unset
3245 }

```

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

```

3246 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3247 \forlistloop#1\@glxtr@unset@buffer
3248 }

```

```

\@glslocalunset  Local unset.
3249 \renewcommand*{\@glslocalunset}[1]{%
3250   \@glslocalunset{#1}%
3251   \glsxtrpostlocalunset{#1}%
3252 }%

rpostlocalunset
3253 \newcommand*{\glsxtrpostlocalunset}[1]{%

\@glsreset  Global reset.
3254 \renewcommand*{\@glsreset}[1]{%
3255   \@glsreset{#1}%
3256   \glsxtrpostreset{#1}%
3257 }%

glsxtrpostreset
3258 \newcommand*{\glsxtrpostreset}[1]{%

\@glslocalreset  Local reset.
3259 \renewcommand*{\@glslocalreset}[1]{%
3260   \@glslocalreset{#1}%
3261   \glsxtrpostlocalreset{#1}%
3262 }%

rpostlocalreset
3263 \newcommand*{\glsxtrpostlocalreset}[1]{%

slocalreseteach  Locally reset a list of entries.
3264 \newcommand*{\glslocalreseteach}[1]{%
3265   \gls@ifnotmeasuring
3266   {%
3267     \@for\@gls@thislabel:=#1\do{%
3268       \glsdoifexists{\@gls@thislabel}%
3269       {%
3270         \@glslocalreset{\@gls@thislabel}%
3271       }%
3272     }%
3273   }%
3274 }

slocalunseteach  Locally unset a list of entries.
3275 \newcommand*{\glslocalunseteach}[1]{%
3276   \gls@ifnotmeasuring
3277   {%
3278     \@for\@gls@thislabel:=#1\do{%
3279       \glsdoifexists{\@gls@thislabel}%
3280       {%
3281         \@glslocalunset{\@gls@thislabel}%

```

```

3282     }%
3283     }%
3284 }%
3285 }

```

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

```

3286 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```

3287   \glsenableentrycount

```

Redefine `\gls` etc:

```

3288   \renewcommand*{\gls}{\cglsl}%
3289   \renewcommand*{\Gls}{\cGls}%
3290   \renewcommand*{\glspl}{\cglspl}%
3291   \renewcommand*{\Glspl}{\cGlspl}%
3292   \renewcommand*{\GLS}{\cGLS}%
3293   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the `entrycount` attribute:

```

3294   \@glsxtr@setentrycountunsetattr{#1}{#2}%

```

In case this command is used again:

```

3295   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3296   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3297     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3298       can't be used with \string\GlsXtrEnableEntryCounting}%
3299     {Use one or other but not both commands}}%
3300 }

```

`ycountunsetattr`

```

3301 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
3302   \@for\@glsxtr@cat:=#1\do
3303   {%
3304     \ifdefempty{\@glsxtr@cat}{}%
3305     {%
3306       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3307     }%
3308   }%
3309 }

```

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

`nableentrycount`

```

3310 \renewcommand*{\glsenableentrycount}{%

```

Enable new fields:

```

3311   \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%

```

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

```

3312 \renewcommand*{\gls@defdocnewglossaryentry}{%
3313 \renewcommand*\newglossaryentry[2]{%
3314 \PackageError{glossaries}{\string\newglossaryentry\space
3315 may only be used in the preamble when entry counting has
3316 been activated}{If you use \string\glsenableentrycount\space
3317 you must place all entry definitions in the preamble not in
3318 the document environment}%
3319 }%
3320 }%

```

New commands to access new fields:

```

3321 \newcommand*{\glsentrycurrcount}[1]{%
3322 \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3323 {0}{\@gls@entry@field{##1}{currcount}}}%
3324 }%
3325 \newcommand*{\glsentryprevcount}[1]{%
3326 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3327 {0}{\@gls@entry@field{##1}{prevcount}}}%
3328 }%

```

Adjust post unset and reset:

```

3329 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
3330 \renewcommand*{\glsxtrpostunset}[1]{%
3331 \@glsxtr@entrycount@org@unset{##1}%
3332 \@gls@increment@currcount{##1}%
3333 }%
3334 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
3335 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3336 \@glsxtr@entrycount@org@localunset{##1}%
3337 \@gls@local@increment@currcount{##1}%
3338 }%
3339 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
3340 \renewcommand*{\glsxtrpostreset}[1]{%
3341 \@glsxtr@entrycount@org@reset{##1}%
3342 \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3343 }%
3344 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
3345 \renewcommand*{\glsxtrpostlocalreset}[1]{%
3346 \@glsxtr@entrycount@org@localreset{##1}%
3347 \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3348 }%

```

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

```

3349 \let\@cgl@@cgl@
3350 \let\@cgl@pl@@cgl@pl@

3351 \let\@cGl@@cGl@
3352 \let\@cGl@pl@@cGl@pl@
3353 \let\@cGL@@cGL@

```

```
3354 \let\@cGLSpl@ \@cGLSpl@
```

The rest is as the original definition.

```
3355 \AtEndDocument{\@gls@write@entrycounts}%
3356 \renewcommand*{\@gls@entry@count}[2]{%
3357   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
3358 }%
3359 \let\glsenableentrycount\relax
3360 \renewcommand*{\glsenableentryunitcount}{%
3361   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3362     can't be used with \string\glsenableentrycount}%
3363   {Use one or other but not both commands}%
3364 }%
3365 }
```

`ite@entrycounts` Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```
3366 \renewcommand*{\@gls@write@entrycounts}{%
3367   \immediate\write\@auxout
3368   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3369   \count@=0\relax
3370   \forallglsentries{\@glsentry}{%
3371     \glshasattribute{\@glsentry}{entrycount}%
3372     {%
3373       \ifglsused{\@glsentry}%
3374       {%
3375         \immediate\write\@auxout
3376         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3377       }%
3378     }%
3379     \advance\count@ by \@ne
3380   }%
3381 }%
3382 }%
3383 \ifnum\count@=0
3384   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3385     \MessageBreak with \string\glsenableentrycount\space but the
3386     \MessageBreak attribute 'entrycount' hasn't
3387     \MessageBreak been assigned to any of the defined
3388     \MessageBreak entries}%
3389 \fi
3390 }
```

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

```
3391 \newcommand*{\glxtrifcounttrigger}[3]{%
```

```

3392 \glshasattribute{#1}{entrycount}%
3393 {%
3394   \ifnum\glstryprevcount{#1}>\glsggetattribute{#1}{entrycount}\relax
3395     #3%
3396   \else
3397     #2%
3398   \fi
3399 }%
3400 {#3}%
3401 }

```

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

\@@cgl@

```

3402 \def\@@cgl@#1#2[#3]{%
3403   \glxtrifcounttrigger{#2}%
3404   {%
3405     \cglformat{#2}{#3}%
3406     \glunset{#2}%
3407   }%
3408   {%
3409     \@gls@{#1}{#2}[#3]%
3410   }%
3411 }%

```

\@@cglsp@

```

3412 \def\@@cglsp@#1#2[#3]{%
3413   \glxtrifcounttrigger{#2}%
3414   {%
3415     \cglspformat{#2}{#3}%
3416     \glunset{#2}%
3417   }%
3418   {%
3419     \@glspl@{#1}{#2}[#3]%
3420   }%
3421 }%

```

\@@cGls@

```

3422 \def\@@cGls@#1#2[#3]{%
3423   \glxtrifcounttrigger{#2}%
3424   {%
3425     \cGlsformat{#2}{#3}%
3426     \glunset{#2}%
3427   }%
3428   {%
3429     \@Gls@{#1}{#2}[#3]%
3430   }%
3431 }%

```


\@@cGlsp1@

```
3432 \def\@@cGlsp1@#1#2[#3]{%
3433   \glxtrifcounttrigger{#2}%
3434   {%
3435     \cGlsp1format{#2}{#3}%
3436     \glset{#2}%
3437   }%
3438   {%
3439     \@Glsp1@{#1}{#2}[#3]%
3440   }%
3441 }%
```

\@@cGLS@

```
3442 \def\@@cGLS@#1#2[#3]{%
3443   \glxtrifcounttrigger{#2}%
3444   {%
3445     \cGLSformat{#2}{#3}%
3446     \glset{#2}%
3447   }%
3448   {%
3449     \@GLS@{#1}{#2}[#3]%
3450   }%
3451 }%
```

\@@cGLSp1@

```
3452 \def\@@cGLSp1@#1#2[#3]{%
3453   \glxtrifcounttrigger{#2}%
3454   {%
3455     \cGLSp1format{#2}{#3}%
3456     \glset{#2}%
3457   }%
3458   {%
3459     \@GLSp1@{#1}{#2}[#3]%
3460   }%
3461 }%
```

Remove default warnings from \cgl's etc so that it can be used interchangeable with \gl's etc.

\@cgl's@

```
3462 \def\@cgl's@#1#2[#3]{\@gl's@{#1}{#2}[#3]}
```

\@cGL's@

```
3463 \def\@cGL's@#1#2[#3]{\@GL's@{#1}{#2}[#3]}
```

\@cglsp1@

```
3464 \def\@cglsp1@#1#2[#3]{\@glsp1@{#1}{#2}[#3]}
```

\@cGLspl@

```
3465 \def\@cGLspl@#1#2[#3]{\@GLspl@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

\cGLS

```
3466 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

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

```
3467 \newcommand*{\@cGLS}[2][\%  
3468 \new@ifnextchar[\@cGLS@{#1}{#2}]{\@cGLS@{#1}{#2}[ ]}%  
3469 }
```

\@cGLS@

```
3470 \def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}
```

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

```
3471 \newcommand*{\cGLSformat}[2]{%  
3472 \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%  
3473 }
```

\cGLSpl

```
3474 \newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

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

```
3475 \newcommand*{\@cGLSpl}[2][\%  
3476 \new@ifnextchar[\@cGLSpl@{#1}{#2}]{\@cGLSpl@{#1}{#2}[ ]}%  
3477 }
```

\@cGLSpl@

```
3478 \def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}
```

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

```
3479 \newcommand*{\cGLSplformat}[2]{%  
3480 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%  
3481 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3482 \renewcommand*{\cglformat}[2]{%  
3483 \glsifregular{#1}  
3484 {\glsentryfirst{#1}}%  
3485 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%  
3486 }
```

\cGlsformat

```
3487 \renewcommand*{\cGlsformat}[2]{%
3488   \glsifregular{#1}
3489   {\Glsentryfirst{#1}}%
3490   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
3491 }
```

\cGlsplformat

```
3492 \renewcommand*{\cGlsplformat}[2]{%
3493   \glsifregular{#1}
3494   {\glsentryfirstplural{#1}}%
3495   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}#2%
3496 }
```

\cGlsplformat

```
3497 \renewcommand*{\cGlsplformat}[2]{%
3498   \glsifregular{#1}
3499   {\Glsentryfirstplural{#1}}%
3500   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
3501 }
```

New code similar to above for unit counting.

defunitcounters

```
3502 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3503   \edef\@glo@countunit{\csuse{@glxtr@categoryattr@@\@glo@category @unitcount}}%
3504   \ifdefvoid\@glo@countunit
3505     {}%
3506     {%
3507       \@glxtr@ifunitcounter{\@glo@countunit}%
3508       {}%
3509       {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
3510     }%
3511 }
```

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

```
3512 \newcommand*{\@glxtr@unitcountlist}{}
```

@addunitcounter

```
3513 \newcommand*{\@glxtr@addunitcounter}[1]{%
3514   \listadd{\@glxtr@unitcountlist}{#1}%
3515   \ifcsundef{glxtr@theunit@#1}
3516     {%
3517       \ifcsdef{theH#1}%
3518       {\csdef{glxtr@theunit@#1}{\csuse{theH#1}}}%
3519       {\csdef{glxtr@theunit@#1}{\csuse{the#1}}}%
3520     }%
3521   {}%
3522 }
```

r@ifunitcounter

```
3523 \newcommand*{\@glxtr@ifunitcounter}[3]{%
3524   \xifinlist{#1}{\@glxtr@unitcountlist}{#2}{#3}%
3525 }
```

urrentunitcount

```
3526 \newcommand*\@glxtr@currentunitcount[1]{%
3527   glo@\glstdetoklabel{#1}@currunit@\glsggetattribute{#1}{unitcount}.%
3528   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
3529 }
```

eviousunitcount

```
3530 \newcommand*\@glxtr@previousunitcount[1]{%
3531   glo@\glstdetoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
3532   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
3533 }
```

t@currunitcount

```
3534 \newcommand*{\@glx@increment@currunitcount}[1]{%
3535   \glshasattribute{#1}{unitcount}%
3536   {%
3537     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3538     \ifcsundef{\@glxtr@csname}%
3539     {%
3540       \csgdef{\@glxtr@csname}{1}%
3541       \listcsxadd
3542       {glo@\glstdetoklabel{#1}@unitlist}%
3543       {\glsggetattribute{#1}{unitcount}.%
3544        \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
3545     }%
3546   }%
3547   {%
3548     \csxdef{\@glxtr@csname}%
3549     {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3550   }%
3551 }%
3552 {}%
3553 }
```

t@currunitcount

```
3554 \newcommand*{\@glx@local@increment@currunitcount}[1]{%
3555   \glshasattribute{#1}{unitcount}%
3556   {%
3557     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3558     \ifcsundef{\@glxtr@csname}%
3559     {%
3560       \csdef{\@glxtr@csname}{1}%
3561       \listcseadd
3562       {glo@\glstdetoklabel{#1}@unitlist}%

```

```

3563     {\glsgetattribute{#1}{unitcount}}.%
3564     \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3565   }%
3566 }%
3567 {%
3568   \csedef{\@glsxtr@csname}%
3569   {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3570   }%
3571 }%
3572 {}%
3573 }

```

r@currunitcount

```

3574 \newcommand*{\@glsxtr@currunitcount}[2]{%
3575   \ifcsundef
3576   {glo@\glsdetoklabel{#1}@currunit@#2}%
3577   {0}%
3578   {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}}%
3579 }%

```

r@prevunitcount

```

3580 \newcommand*{\@glsxtr@prevunitcount}[2]{%
3581   \ifcsundef
3582   {glo@\glsdetoklabel{#1}@prevunit@#2}%
3583   {0}%
3584   {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}}%
3585 }%

```

eentryunitcount

```

3586 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3587   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3588   \renewcommand*{\gls@defdocnewglossaryentry}{%
3589     \renewcommand*\newglossaryentry[2]{%
3590       \PackageError{glossaries}{\string\newglossaryentry\space
3591       may only be used in the preamble when entry counting has
3592       been activated}{If you use \string\glsenableentryunitcount\space
3593       you must place all entry definitions in the preamble not in
3594       the document environment}}%
3595     }%
3596   }%
  New commands to access new fields:
3597   \newcommand*{\glsentrycurrcount}[1]{%
3598     \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3599     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3600   }%

```

```

3601 \newcommand*{\glstentryprevcount}[1]{%
3602   \@glstxtr@prevunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
3603   \csuse{glstxtr@theunit@\glstgetattribute{##1}{unitcount}}}%
3604 }%

```

Access total count:

```

3605 \newcommand*{\glstentryprevtotalcount}[1]{%
3606   \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
3607   {0}%
3608   {%
3609     \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
3610   }%
3611 }%

```

Access max value:

```

3612 \newcommand*{\glstentryprevmaxcount}[1]{%
3613   \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
3614   {0}%
3615   {%
3616     \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
3617   }%
3618 }%

```

Adjust post unset and reset:

```

3619 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
3620 \renewcommand*{\glstxtrpostunset}[1]{%
3621   \@glstxtr@entryunitcount@org@unset{##1}%
3622   \@glst@increment@currunitcount{##1}%
3623 }%
3624 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
3625 \renewcommand*{\glstxtrpostlocalunset}[1]{%
3626   \@glstxtr@entryunitcount@org@localunset{##1}%
3627   \@glst@local@increment@currunitcount{##1}%
3628 }%
3629 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
3630 \renewcommand*{\glstxtrpostreset}[1]{%
3631   \glsthasattribute{##1}{unitcount}%
3632   {%
3633     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
3634     \ifcsundef{\@glstxtr@csname}%
3635     {}%
3636     {\csgdef{\@glstxtr@csname}{0}}%
3637   }%
3638   {}%
3639 }%
3640 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
3641 \renewcommand*{\glstxtrpostlocalreset}[1]{%
3642   \@glstxtr@entryunitcount@org@localreset{##1}%
3643   \glsthasattribute{##1}{unitcount}%
3644   {%
3645     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%

```

```

3646 \ifcsundef{\@glsxtr@csname}%
3647 {}%
3648 {\csdef{\@glsxtr@csname}{0}}%
3649 }%
3650 {}%
3651 }%

```

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

```

3652 \let\@cgls@\@cgls@
3653 \let\@cglspl@\@cglspl@

3654 \let\@cGls@\@cGls@
3655 \let\@cGlspl@\@cGlspl@
3656 \let\@cGLS@\@cGLS@
3657 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3658 \AtEndDocument{\@gls@write@entryunitcounts}%
3659 \renewcommand*{\@gls@entry@unitcount}[3]{%
3660 \csgdef{glo@glsdetoklabel{##1}@prevunit@##3}{##2}%
3661 \ifcsundef{glo@glsdetoklabel{##1}@prevunittotal}%
3662 {\csgdef{glo@glsdetoklabel{##1}@prevunittotal}{##2}}%
3663 {%
3664 \csxdef{glo@glsdetoklabel{##1}@prevunittotal}{
3665 \number\numexpr\csuse{glo@glsdetoklabel{##1}@prevunittotal}+##2}%
3666 }%
3667 \ifcsundef{glo@glsdetoklabel{##1}@prevunitmax}%
3668 {\csgdef{glo@glsdetoklabel{##1}@prevunitmax}{##2}}%
3669 {%
3670 \ifnum\csuse{glo@glsdetoklabel{##1}@prevunitmax}<##2
3671 \csgdef{glo@glsdetoklabel{##1}@prevunitmax}{##2}%
3672 \fi
3673 }%
3674 }%
3675 \let\glsenableentryunitcount\relax
3676 \renewcommand*{\glsenableentrycount}{%
3677 \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3678 can't be used with \string\glsenableentryunitcount}%
3679 {Use one or other but not both commands}%
3680 }%
3681 }
3682 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3683 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3684 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3685 \immediate\write\@auxout

```

```

3686   {\string\@gls@entry@unitcount
3687     {\@glsentry}%
3688     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3689     }%
3690     {#1}}%
3691 }

```

entryunitcounts

```

3692 \newcommand*{\@gls@write@entryunitcounts}{%
3693   \immediate\write\@auxout
3694   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3695   \count@=0\relax
3696   \forallglsentries{\@glsentry}{%
3697     \glshasattribute{\@glsentry}{unitcount}%
3698     {%
3699       \ifglsused{\@glsentry}%
3700       {%
3701         \forlistcsloop
3702           {\@gls@write@entryunitcounts@do}%
3703           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3704       }%
3705     }%
3706     \advance\count@ by \@ne
3707   }%
3708 }%
3709 }%
3710 \ifnum\count@=0
3711   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3712     \MessageBreak with \string\glsenableentryunitcount\space but the
3713     \MessageBreak attribute ‘unitcount’ hasn’t
3714     \MessageBreak been assigned to any of the defined
3715     \MessageBreak entries}%
3716 \fi
3717 }

```

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.

```

3718 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3719   \glsenableentryunitcount

```

Redefine \gls etc:

```

3720   \renewcommand*{\gls}{\cgl}%
3721   \renewcommand*{\Gls}{\cGls}%
3722   \renewcommand*{\glspl}{\cglspl}%
3723   \renewcommand*{\Glspl}{\cGlspl}%
3724   \renewcommand*{\GLS}{\cGLS}%
3725   \renewcommand*{\GLSpl}{\cGLSpl}%

```


Set the entrycount attribute:

```
3726 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
3727 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
3728 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3729 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3730 can't be used with \string\GlsXtrEnableEntryUnitCounting}%
3731 {Use one or other but not both commands}}%
3732 }
```

countunsetattr

```
3733 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3734 \@for\@glsxtr@cat:=#1\do
3735 {%
3736 \ifdefempty{\@glsxtr@cat}{}%
3737 {%
3738 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3739 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3740 }%
3741 }%
3742 }
```

1.3.6 Acronym Modifications

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

genericNewAcronym

```
3743 \renewcommand*{\SetGenericNewAcronym}{%
3744 \let\@Gls@entryname\@Gls@acrenryname
3745 \renewcommand{\newacronym}[4][{}]{%
3746 \ifdefempty{\@glsacronymlists}%
3747 {%
3748 \def\@glo@type{\acronymtype}%
3749 \setkeys{glossentry}{##1}%
3750 \DeclareAcronymList{\@glo@type}%
3751 }%
3752 {}%
3753 \glskeylisttok{##1}%
3754 \glslabeltok{##2}%
3755 \glsshorttok{##3}%
3756 \glslongtok{##4}%
3757 \newacronymhook
```

```

3758 \protected@edef\@do@newglossaryentry{%
3759 \noexpand\newglossaryentry{\the\glslabeltok}%
3760 {%
3761 type=\acronymtype,%
3762 name={\expandonce{\acronymentry{##2}}},%
3763 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3764 text={\the\glsshorttok},%
3765 short={\the\glsshorttok},%
3766 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3767 long={\the\glslongtok},%
3768 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3769 category=acronym,
3770 \GenericAcronymFields,%
3771 \the\glskeylisttok
3772 }%
3773 }%
3774 \@do@newglossaryentry
3775 }%
3776 \renewcommand*{\acrfullfmt}[3]{%
3777 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3778 \renewcommand*{\Acrfullfmt}[3]{%
3779 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3780 \renewcommand*{\ACRfullfmt}[3]{%
3781 \glslink[##1]{##2}{%
3782 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
3783 \renewcommand*{\acrfullplfmt}[3]{%
3784 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3785 \renewcommand*{\Acrfullplfmt}[3]{%
3786 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3787 \renewcommand*{\ACRfullplfmt}[3]{%
3788 \glslink[##1]{##2}{%
3789 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
3790 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
3791 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
3792 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
3793 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
3794 }

```

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:

```

3795 \let\@glxstr@org@setacronymstyle\setacronymstyle
3796 \let\@glxstr@org@newacronymstyle\newacronymstyle

```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbreviationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```

3797 \newcommand*{\MakeAcronymsAbbreviations}{%
3798 \renewcommand*{\newacronym}[4][]{%

```

```

3799 \glstr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3800 }%
3801 \renewcommand*{\firstacronymfont}[1]{\glstrfirstabbrvfont{##1}}%
3802 \renewcommand*{\acronymfont}[1]{\glstrabbrvfont{##1}}%
3803 \renewcommand*{\setacronymstyle}[1]{%
3804 \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
3805 unavailable.
3806 Use \string\setabbreviationstyle\space instead.
3807 The original acronym interface can be restored with
3808 \string\RestoreAcronyms}{}}%
3809 }%
3810 \renewcommand*{\newacronymstyle}[1]{%
3811 \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3812 available unless you restore the original acronym interface with
3813 \string\RestoreAcronyms}}%
3814 \@glstr@org@newacronymstyle{##1}%
3815 }%
3816 }

```

Switch acronyms to abbreviations:

```
3817 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3818 \newcommand*{\RestoreAcronyms}{%
3819 \SetGenericNewAcronym
3820 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3821 \renewcommand{\acronymfont}[1]{##1}%
3822 \let\setacronymstyle\@glstr@org@setacronymstyle
3823 \let\newacronymstyle\@glstr@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.

```

3824 \renewcommand*{\@gls@link@checkfirsthyper}{%
3825 \ifglstrused{\glstrlabel}%
3826 {\let\glstrifwasfirstuse\@secondoftwo}
3827 {\let\glstrifwasfirstuse\@firstoftwo}%
3828 \@glstr@org@checkfirsthyper
3829 }
3830 \glssetcategoryattribute{acronym}{regular}{false}%
3831 \setacronymstyle{long-short}%
3832 }

```

\glsacspace Allow the user to customise the maximum value.

```

3833 \renewcommand*{\glsacspace}[1]{%
3834 \settowidth{\dimen@}{(\firstacronymfont{\glstrentryshort{#1}})}%
3835 \ifdim\dimen@<\glsacspacemax~\else\space\fi
3836 }

```

\glsacspacemax Value used in the above.

```
3837 \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

```
3838 \newcommand*{\@glxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
3839 \let\@glxtr@org@makeglossaries\makeglossaries
```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with record.

\makeglossaries

```
3840 \renewcommand*{\makeglossaries}[1][]{%
3841   \ifx\@glxtr@record@setting\@glxtr@record@setting@only
3842     \PackageError{glossaries-extra}{\string\makeglossaries\space
3843       not permitted\MessageBreak with record=only package option}%
3844     {You may only use \string\makeglossaries\space with
3845       record=off or record=alsoindex options}%
3846   \else
3847     \ifblank{#1}%
3848       {\@glxtr@org@makeglossaries}%
3849       {%
3850         \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3851           \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3852             not permitted\MessageBreak with record=alsoindex package option}%
3853           {You may only use the hybrid \string\makeglossaries[...]\space with
3854             record=off option}%
3855         \else
3856           \edef\@glxtr@reg@glosslist{#1}%
3857           \ifundef{\glswrite}{\newwrite\glswrite}{}%
3858           \protected@write\@auxout{}{\string\providecommand
3859             \string\@glxtr@order[1]}{}
3860           \protected@write\@auxout{}{\string\providecommand
3861             \string\@istfilename[1]}{}
3862           \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3863           \protected@write\@auxout{}{\string\@glxtr@order{\glxtr@order}}
3864           \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}
3865           \write\@auxout{\string\providecommand\string\@glxtr@reference[3]}{}%
3866         \for\@glo@type:=#1\do{%
3867           \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3868         }%
```

Iterate through each supplied glossary type and activate it.

```
3866   \for\@glo@type:=#1\do{%
3867     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3868   }%
```

New glossaries must be created before `\makeglossaries`:

```
3869 \renewcommand*\newglossary[4] [] {%
3870 \PackageError{glossaries}{New glossaries
3871 must be created before \string\makeglossaries}{You need
3872 to move \string\makeglossaries\space after all your
3873 \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
3874 \let\@makeglossary\relax
3875 \let\makeglossary\relax
3876 \renewcommand\makeglossaries[1] [] {}%
```

Disable all commands that have no effect after `\makeglossaries`

```
3877 \@disable@onlypremakeg
```

Allow see key:

```
3878 \let\gls@checkseeallowed\relax
```

Adjust `\@do@seeglossary`. This needs to check for the entry's existence but don't increment associated counter.

```
3879 \renewcommand*\@do@seeglossary}[2] {%
3880 \glsdoifexists{##1}%
3881 {%
3882 \edef\@gls@label{\glsdetoklabel{##1}}%
3883 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3884 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3885 {\@glsxtr@org@doseeglossary{##1}{##2}}%
3886 {%
3887 \@@glsxtrwrglossmark
3888 \protected@write\@auxout{}{%
3889 \string\@gls@reference
3890 {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
3891 }%
3892 }%
3893 }%
3894 }%
```

Adjust `\@do@@wrglossary`

```
3895 \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
3896 \def\@do@@wrglossary{%
3897 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3898 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3899 {\@glsxtr@@do@@wrglossary}%
3900 {\@gls@noidxglossary}%
3901 }%
```

Suppress warning about no `\makeglossaries`

```
3902 \let\warn@nomakeglossaries\relax
3903 \def\warn@noprintglossary{%
3904 \GlossariesWarningNoLine{No \string\printglossary\space
3905 or \string\printglossaries\space
```

```

3906         found.^^J(Remove \string\makeglossaries\space if you don't want
3907         any glossaries.)^^JThis document will not have a glossary}%
3908     }%

```

Only warn for glossaries not listed.

```

3909     \renewcommand{\@gls@noref@warn}[1]{%
3910         \edef\@gls@type{##1}%
3911         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3912         {%
3913             \GlossariesExtraWarning{Can't use
3914                 \string\printnoidxglossary[type={\@gls@type}]
3915                 when '\@gls@type' is listed in the optional argument of
3916                 \string\makeglossaries}%
3917         }%
3918     }%
3919     \GlossariesWarning{Empty glossary for
3920         \string\printnoidxglossary[type={##1}].
3921         Rerun may be required (or you may have forgotten to use
3922         commands like \string\gls)}%
3923 }%
3924 }%

```

Adjust display number list to check for type:

```

3925     \renewcommand*{\glsdisplaynumberlist}[1]{%
3926         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3927         {\@glsxtr@idx@displaynumberlist{##1}}%
3928         {\@glsxtr@noidx@displaynumberlist{##1}}%
3929     }%

```

Adjust entry list:

```

3930     \renewcommand*{\glsentrynumberlist}[1]{%
3931         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3932         {\@glsxtr@idx@entrynumberlist{##1}}%
3933         {\@glsxtr@noidx@entrynumberlist{##1}}%
3934     }%

```

Adjust number list loop

```

3935     \renewcommand*{\glsnumberlistloop}[2]{%
3936         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3937         {%
3938             \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3939                 not available for glossary '##1'}{%
3940             }%
3941             {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3942         }%

```

Only sanitize sort for normal indexing glossaries.

```

3943     \renewcommand*{\glsprestandardsort}[3]{%
3944         \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3945         {%
3946             \glsdosanitizesort

```

```

3947     }%
3948     {%
3949         \ifglssanitizesort
3950         \@gls@noidx@sanitizesort
3951         \else
3952         \@gls@noidx@nosanitizesort
3953         \fi
3954     }%
3955 }%

```

Unlike `\makenoidxglossaries` we can't automatically set `sanitizesort=false`. All entries must be defined in the preamble.

```

3956     \renewcommand*\new@glossaryentry[2]{%
3957         \PackageError{glossaries-extra}{Glossary entries must be defined
3958         in the preamble\MessageBreak when you use the optional argument
3959         of \string\makeglossaries}{Either move your definitions to the
3960         preamble or don't use the optional argument of
3961         \string\makeglossaries}%
3962     }%

```

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).

```

3963     \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
3964     \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3965         \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
3966         type=\glsdefaulttype,\@end@glsxtr@gettype
3967         \def\@glo@sorttype{\@glo@default@sorttype}%
3968     }%

```

Check automake setting:

```

3969     \ifglsautomake
3970     \renewcommand*\@gls@doautomake{%
3971         \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
3972             \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3973         }%
3974     }%
3975     \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3976     \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3977     \fi
3978 }%
3979 \fi
3980 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\rgprintglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes

\provideignoredglossary to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

3981 \newcommand{\@glstr@orgprintglossary}[2]{%
3982   \def\@glo@type{\glstexttype}%
3983   \def\glossarytitle{%
3984     \ifcsdef{\@glo@type\@glo@type @title}%
3985       {\csuse{\@glo@type\@glo@type @title}}%
3986       {\glossaryname}}%
3987   \def\glossarytoctitle{\glossarytitle}%
3988   \let\org@glossarytitle\glossarytitle
3989   \def\@glossarystyle{%
3990     \ifx\@glossary@default@style\relax
3991       \GlossariesWarning{No default glossary style provided \MessageBreak
3992         for the glossary '\@glo@type'. \MessageBreak
3993         Using deprecated fallback. \MessageBreak
3994         To fix this set the style with \MessageBreak
3995         \string\setglossarystyle\space or use the \MessageBreak
3996         style key=value option}%
3997     \fi
3998   }%
3999   \def\gls@dotoc@title{\glstexttoctitle{\@glo@type}}%
4000   \let\org@glossaryentrynumbers\glossaryentrynumbers
4001   \bgroup
4002     \@printgloss@setsort
4003     \setkeys{printgloss}{#1}%
4004     \ifx\glossarytitle\org@glossarytitle
4005     \else
4006       \cslet{\@glo@type\@glo@type @title}{\glossarytitle}%
4007     \fi
4008     \let\currentglossary\@glo@type
4009     \let\org@glossaryentrynumbers\glossaryentrynumbers
4010     \let\glsnonextpages\@glsnonextpages
4011     \let\glsnextpages\@glsnextpages
4012     \glstractivatenopost
4013     \gls@dotoc@title
4014     \@glossarystyle
4015     \let\gls@org@glossaryentryfield\glossentry
4016     \let\gls@org@glossarysubentryfield\subglossentry
4017     \renewcommand{\glossentry}[1]{%
4018       \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4019       \gls@org@glossaryentryfield{##1}%
4020     }%
4021     \renewcommand{\subglossentry}[2]{%
4022       \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4023       \gls@org@glossarysubentryfield{##1}{##2}%
4024     }%
4025     \@gls@preglossaryhook

```



```

4026      #2%
4027    \egroup
4028    \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4029    \global\let\warn@noprintglossary\relax
4030  }

```

ractivatenopost Change \nopostdesc and \glxtrnopostpunc to behave as they do in the glossary.

```

4031 \newcommand*{\glxtractivatenopost}{%
4032   \let\nopostdesc\@nopostdesc
4033   \let\glxtrnopostpunc\@glxtr@nopostpunc
4034 }

```

lsxtrnopostpunc

```

4035 \newrobustcmd*{\glxtrnopostpunc}{%

```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

4036 \newcommand{\@glxtr@nopostpunc}{%
4037   \let\@glxtr@org@postdescription\glspostdescription
4038   \ifglsnopostdot
4039     \renewcommand{\glspostdescription}{%
4040       \glsnopostdottrue
4041       \let\glspostdescription\@glxtr@org@postdescription
4042       \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4043       \glxtrpostdescription
4044       \@glxtr@nopostpunc@postdesc}%
4045   \else
4046     \renewcommand{\glspostdescription}{%
4047       \let\glspostdescription\@glxtr@org@postdescription
4048       \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4049       \glxtrpostdescription
4050       \@glxtr@nopostpunc@postdesc}%
4051   \fi
4052   \glsnopostdotfalse
4053 }

```

stpunc@postdesc

```

4054 \newcommand*{\@glxtr@nopostpunc@postdesc}{%

```

estore@postpunc

```

4055 \newcommand*{\@glxtr@restore@postpunc}{%
4056   \def\@glxtr@nopostpunc@postdesc{%
4057     \@glxtr@org@postdescription
4058     \let\@glxtr@nopostpunc@postdesc\@empty
4059     \let\glxtrrestorepostpunc\@empty
4060   }%
4061 }

```

restorepostpunc Does nothing outside of glossary.

```
4062 \newcommand*{\glxtrrestorepostpunc}{}

\@printglossary Redefine.
```

```
4063 \renewcommand{\@printglossary}[2]{%
4064   \def\@glxtr@printglossopts{#1}%
4065   \@glxtr@orgprintglossary{#1}{#2}%
4066 }
```

Add a key that switches off the entry targets:

```
4067 \define@choicekey{printgloss}{target}
4068 [\@glxtr@printglossval\@glxtr@printglossnr]%
4069 {true,false}[true]%
4070 {%
4071   \ifcase\@glxtr@printglossnr

4072     \def\@glstarget{\glsdohypertarget}%
4073   \else
4074     \let\@glstarget\@secondoftwo
4075   \fi
4076 }
```

hypernameprefix

```
4077 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:
```

```
4078 \define@key{printgloss}{targetnameprefix}{%
4079   \renewcommand{\@glxtrhypernameprefix}{#1}%
4080 }

4081 \define@key{printgloss}{prefix}{%
4082   \renewcommand{\glolinkprefix}{#1}%
4083 }
```

glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```
4084 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4085 \renewcommand{\glsdohypertarget}[2]{%
4086   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4087 }
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
4088 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4089   \def\@glstarget{\glsdohypertarget}%
4090 \fi
4091 \end{macro}
```

@makeglossaries For the benefit of makeglossaries

```
4092 \newcommand*{\glxtr@makeglossaries}[1]{}
```

@glxtr@gettype Get just the type.

```
4093 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4094   \def\@glo@type{#2}%
4095 }
```

@assign@sortkey Assign the sort key.

```
4096 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4097   \edef\@glo@type{\@glo@type}%
4098   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4099   {%
4100     \@glo@no@assign@sortkey{#1}%
4101   }%
4102   {%
4103     \@glo@assign@sortkey{#1}%
4104   }%
4105 }
```

Display number list for the regular version:

splaynumberlist

```
4106 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4107 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4108   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4109   \ifdef\@gls@loclist
4110   {%
4111     \def\@gls@noidxloclist@sep{%
4112       \def\@gls@noidxloclist@sep{%
4113         \def\@gls@noidxloclist@sep{%
4114           \glsnumlistsep
4115         }%
4116         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4117       }%
4118     }%
4119     \def\@gls@noidxloclist@finalsep{}%
4120     \def\@gls@noidxloclist@prev{}%
4121     \forlistloop{\@gls@noidxdisplayloclisthandler}{\@gls@loclist}%
4122     \@gls@noidxloclist@finalsep
4123     \@gls@noidxloclist@prev
4124   }%
4125   {%
4126     \glxtrundeftag
4127     \glsdoifexists{#1}%
4128     {%
4129       \GlossariesWarning{Missing location list for ‘#1’. Either
4130         a rerun is required or you haven’t referenced the entry.}%

```

```

4131 }%
4132 }%
4133 }%
4134

```

And for the number list loop:

@numberlistloop

```

4135 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
4136   \letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
4137   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4138   \let\@gls@org@glssseeformat\glssseeformat
4139   \let\glsnoidxdisplayloc#2\relax
4140   \let\glssseeformat#3\relax
4141   \ifdef\@gls@loclist
4142   {%
4143     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4144   }%
4145   {%

4146     \glsxtrundeftag
4147     \glsoifexists{#1}%
4148     {%
4149       \GlossariesWarning{Missing location list for ‘##1’. Either
4150         a rerun is required or you haven’t referenced the entry.}%
4151     }%
4152   }%
4153   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4154   \let\glssseeformat\@gls@org@glssseeformat
4155 }%

```

Same for entry number list.

entrynumberlist

```

4156 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4157   \letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
4158   \ifdef\@gls@loclist
4159   {%
4160     \glsnoidxloclist{\@gls@loclist}%
4161   }%
4162   {%

4163     \glsxtrundeftag
4164     \glsoifexists{#1}%
4165     {%
4166       \GlossariesWarning{Missing location list for ‘#1’. Either
4167         a rerun is required or you haven’t referenced the entry.}%
4168     }%
4169   }%
4170 }%

```

entrynumberlist

```
4171 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
4172 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4173   \protected@edef\@glsxtr@titlelabel{#1}%
4174   \ifdefvoid\@glsxtr@titlelabel
4175   {%
4176     {%
4177       \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4178     }%
4179     \ifdefvoid{\@glsxtr@titlelabel}%
4180     {%
4181       \DTLifint{#1}%
4182       {%
4183         \ifnum#1<256\relax
4184           \edef#2{\char#1\relax}%
4185         \else
4186           \edef#2{#1}%
4187         \fi
4188       }%
4189       {%
4190         \ifcsundef{#1groupname}%
4191         {\def#2{#1}}%
4192         {\letcs#2{#1groupname}}%
4193       }%
4194     }%
4195     {%
4196       \let#2\@glsxtr@titlelabel
4197     }%
4198 }
```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```
4199 \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.

```
4200 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4201   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4202   \@onelevel@sanitize\@glsxtr@titlelabel
4203   \ifcsdef{\@glsxtr@titlelabel}
4204   {\letcs{#2}{\@glsxtr@titlelabel}}%
4205   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4206 }
4207 \let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

trsetgrouptitle Sets the title for the given group label.

```
4208 \newcommand{\glsxtrsetgrouptitle}[2]{%
```

```

4209 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
4210 \onelevel@sanitize\@glstr@titlelabel
4211 \protected@csxdef{\@glstr@titlelabel}{#2}%
4212 }

```

`\alsetgrouptitle` As above put only locally defines the title.

```

4213 \newcommand{\glstrlocalsetgrouptitle}[2]{%
4214 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
4215 \onelevel@sanitize\@glstr@titlelabel
4216 \protected@csedef{\@glstr@titlelabel}{#2}%
4217 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4218 \renewcommand*{\glsnavigation}{%
4219 \def\@gls@between{%
4220 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
4221 {%
4222 \def\@gls@list{%
4223 }%
4224 {%
4225 \expandafter\let\expandafter\@gls@list
4226 \csname \@gls@hypergroup@list@\@glo@type\endcsname
4227 }%
4228 \@for\@gls@tmp:=\@gls@list\do{%
4229 \@gls@between
4230 \glstrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4231 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4232 \let\@gls@between\glshypernavsep
4233 }%
4234 }

```

`@noidx@glossary`

```

4235 \renewcommand*{\@print@noidx@glossary}{%
4236 \ifcsdef{\@glsref@\@glo@type}%
4237 {%
4238 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4239 {%
4240 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4241 }%
4242 {%
4243 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4244 }%
4245 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4246 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4247 \def\@gls@currentlettergroup{%
4248 \begin{theglossary}%

```

```

4249 \glossaryheader
4250 \glsresetentrylist
4251 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
4252 \end{theglossary}%
4253 \glossarypostamble
4254 }%
4255 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4256 \glstrifemptyglossary{\@glo@type}%
4257 }%
4258 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4259 \@gls@noref@warn{\@glo@type}%
4260 }%
4261 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4262 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4263 \setentrycounter[#1]{#2}%
4264 \@glstr@display@loc#3\empty\end@glstr@display@loc{#4}%
4265 }

```

`xtr@display@loc` Patch to check for range formations.

```

4266 \def\@glstr@display@loc#1#2\end@glstr@display@loc#3{%
4267 \ifx#1(\relax
4268 \glstrdisplaystartloc{#2}{#3}%
4269 \else
4270 \ifx#1)\relax
4271 \glstrdisplayendloc{#2}{#3}%
4272 \else
4273 \glstrdisplaysingleloc{#1#2}{#3}%
4274 \fi
4275 \fi
4276 }

```

`isplaysingleloc` Single location.

```

4277 \newcommand*{\glstrdisplaysingleloc}[2]{%
4278 \csuse{#1}{#2}%
4279 }

```

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 `\glstrlocrangefmt`.

`displaystartloc` Start of a location range.

```

4280 \newcommand*{\glstrdisplaystartloc}[2]{%
4281 \edef\glstrlocrangefmt{#1}%
4282 \ifx\glstrlocrangefmt\empty
4283 \def\glstrlocrangefmt{\glsnumberformat}%

```

```

4284 \fi
4285 \expandafter\glxtrdisplaysingleloc
4286 \expandafter{\glxtrlocrangefmt}{#2}%
4287 }

```

`trdisplayendloc` End of a location range.

```

4288 \newcommand*{\glxtrdisplayendloc}[2]{%
4289 \edef\@glxtr@tmp{#1}%
4290 \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{glnumberformat}}{}}%
4291 \ifx\glxtrlocrangefmt\@glxtr@tmp
4292 \else
4293 \GlossariesExtraWarning{Mismatched end location range
4294 (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
4295 \fi
4296 \expandafter\glxtrdisplayendloohook\expandafter{\@glxtr@tmp}{#2}%
4297 \expandafter\glxtrdisplaysingleloc
4298 \expandafter{\glxtrlocrangefmt}{#2}%
4299 \def\glxtrlocrangefmt{}%
4300 }

```

`splayendloohook` Allow the user to hook into the end of range command.

```

4301 \newcommand*{\glxtrdisplayendloohook}[2]{

```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```

4302 \newcommand*{\glxtrlocrangefmt}{}

```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4303 \renewcommand*{\setentrycounter}[2][{}]{%
4304 \def\glxtrcounterprefix{#1}%
4305 \ifx\glxtrcounterprefix\@empty
4306 \def\@glo@counterprefix{.}%
4307 \else
4308 \def\@glo@counterprefix{.#1.}%
4309 \fi
4310 \def\glsetentrycounter{#2}%
4311 }

```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4312 \def\@gls@removespaces#1 #2\@nil{%
4313 \toks@=\expandafter{\the\toks@#1}%
4314 \ifx\@#2\%
4315 \edef\x{\the\toks@}%
4316 \ifx\x\empty
4317 \else
4318 \expandafter\glxtrlocationhyperlink\expandafter
4319 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4320 \fi

```



```

4321 \else
4322   \@gls@ReturnAfterFi{%
4323     \@gls@removespaces#2\@nil
4324   }%
4325 \fi
4326 }

```

cationhyperlink

```

4327 \newcommand*{\glstrlocationhyperlink}[3]{%
4328   \ifdefvoid\glstrsupplocationurl
4329     {%
4330       \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4331     }%
4332   {%
4333     \hyperref{\glstrsupplocationurl}{#1#2#3}{#3}%
4334   }%
4335 }

```

supphypernumber

```

4336 \newcommand*{\glstrsupphypernumber}[1]{%
4337   {%
4338     \glshasattribute{\glscurrententrylabel}{externallocation}%
4339     {%
4340       \def\glstrsupplocationurl{%
4341         \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4342     }%
4343     {%
4344       \def\glstrsupplocationurl{}%
4345     }%
4346     \glshypernumber{#1}%
4347   }%
4348 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4349 \renewcommand{\@print@glossary}{%
4350   \makeatletter
4351   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4352   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4353   {%
4354     {\glstrNoGlossaryWarning{\@glo@type}}%
4355     \ifglsxindy
4356       \ifcsundef{@xdy@\@glo@type @language}%
4357       {%
4358         \edef\@do@auxoutstuff{%
4359           \noexpand\AtEndDocument{%
4360             \noexpand\immediate\noexpand\write\@auxout{%

```

```

4361         \string\providecommand\string\@xdylanguage[2]{}%
4362         \noexpand\immediate\noexpand\write\@auxout{%
4363         \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4364     }%
4365 }%
4366 }%
4367 {%
4368     \edef\@do@auxoutstuff{%
4369     \noexpand\AtEndDocument{%
4370     \noexpand\immediate\noexpand\write\@auxout{%
4371     \string\providecommand\string\@xdylanguage[2]{}%
4372     \noexpand\immediate\noexpand\write\@auxout{%
4373     \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4374     @language\endcsname}}%
4375     }%
4376     }%
4377     }%
4378     \@do@auxoutstuff
4379     \edef\@do@auxoutstuff{%
4380     \noexpand\AtEndDocument{%
4381     \noexpand\immediate\noexpand\write\@auxout{%
4382     \string\providecommand\string\@gls@codepage[2]{}%
4383     \noexpand\immediate\noexpand\write\@auxout{%
4384     \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4385     }%
4386     }%
4387     \@do@auxoutstuff
4388     \fi
4389     \renewcommand*{\@warn@nomakeglossaries}{%
4390     \GlossariesWarningNoLine{\string\makeglossaries\space
4391     hasn't been used,^^Jthe glossaries will not be updated}%
4392     }%
4393 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4394 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4395 This document is incomplete. The external file associated with
4396 the glossary '#1' (which should be called \texttt{#2})
4397 hasn't been created.%
4398 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4399 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4400 This has probably happened because there are no entries defined
4401 in this glossary.%
4402 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4403 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4404   If you don't want this glossary,
4405   add \texttt{nomain} to your package option list when you load
4406   \texttt{glossaries-extra.sty}. For example:%
4407 }

```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```

4408 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4409   Did you forget to use \texttt{type=#1} when you defined your
4410   entries? If you tried to load entries into this glossary with
4411   \texttt{\string\loadglsentries} did you remember to use
4412   \texttt{[#1]} as the optional argument? If you did, check that
4413   the definitions in the file you loaded all had the type set
4414   to \texttt{\string\glsdefaulttype}.%
4415 }

```

arningCheckFile Advisory message to check the file contents.

```

4416 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4417   Check the contents of the file \texttt{#1}. If
4418   it's empty, that means you haven't indexed any of your entries in this
4419   glossary (using commands like \texttt{\string\gls} or
4420   \texttt{\string\glsadd}) so this list can't be generated.
4421   If the file isn't empty, the document build process hasn't been
4422   completed.%
4423 }

```

WarningAutoMake Message when automake option has been used.

```

4424 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4425   You may need to rerun \LaTeX. If you already have, it may be that
4426   \TeX's shell escape doesn't allow you to run
4427   \ifglxindy xindy\else makeindex\fi. Check the
4428   transcript file \texttt{\jobname.log}. If the shell escape is
4429   disabled, try one of the following:
4430
4431   \begin{itemize}
4432     \item Run the external (Lua) application:
4433
4434       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4435
4436     \item Run the external (Perl) application:
4437
4438       \texttt{makeglossaries \string"\jobname\string"}
4439   \end{itemize}
4440
4441   Then rerun \LaTeX\ on this document.
4442   \GlossariesExtraWarning{Rerun required to build the
4443   glossary '#1' or check TeX's shell escape allows
4444   you to run \ifglxindy xindy\else makeindex\fi}%
4445 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4446 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4447   You need to either replace \texttt{\string\makenoidxglossaries}
4448   with \texttt{\string\makeglossaries} or replace
4449   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4450   \texttt{\string\printnoidxglossary}
4451   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4452   this document.%
4453 }

```

arningBuildInfo Build advice.

```

4454 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4455   Try one of the following:
4456   \begin{itemize}
4457     \item Add \texttt{automake} to your package option list when you load
4458       \texttt{glossaries-extra.sty}. For example:
4459
4460       \texttt{\string\usepackage[automake]%
4461         \glssopenbrace glossaries-extra\glsclosebrace}
4462
4463     \item Run the external (Lua) application:
4464
4465     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4466
4467     \item Run the external (Perl) application:
4468
4469     \texttt{makeglossaries \string"\jobname\string"}
4470   \end{itemize}
4471
4472   Then rerun \LaTeX\ on this document.%
4473 }

```

trRecordWarning Paragraph for record=only.

```

4474 \newcommand{\GlsXtrRecordWarning}[1]{%
4475   \texttt{\string\printglossary} doesn't work
4476   with the \texttt{record=only} package option
4477   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4478   instead (or change the package option).%
4479 }

```

oGlsWarningTail Final paragraph.

```

4480 \newcommand{\GlsXtrNoGlsWarningTail}{%
4481   This message will be removed once the problem has been fixed.%
4482 }

```

GlsWarningNoOut No out file created. Build advice.

```

4483 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4484   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4485   \texttt{\string\makeglossaries} or you have used

```

```

4486 \texttt{\string\nofiles}. If this is just a draft version of the
4487 document, you can suppress this message using the
4488 \texttt{nomissingglstext} package option.%
4489 }

```

glossarywarning

```

4490 \newcommand*{\@glstr@defaultnoglossarywarning}[1]{%
4491 \glossarysection[\glossarytoctitle]{\glossarytitle}
4492 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4493 \par
4494 \glstrifemptyglossary{#1}%
4495 {%
4496 \GlsXtrNoGlsWarningEmptyStart\space
4497 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4498 \medskip
4499 \noindent\texttt{\string\usepackage[nomain\ifglstacronym ,acronym\fi]%
4500 \glstopenbrace glossaries-extra\glstclosebrace}
4501 \medskip
4502 }%
4503 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4504 }%
4505 {%
4506 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4507 {%
4508 \GlsXtrNoGlsWarningCheckFile
4509 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4510
4511 \ifglstautomake
4512
4513 \GlsXtrNoGlsWarningAutoMake{#1}
4514
4515 \else
4516
4517 \ifthenelse{\equal{#1}{main}}%
4518 {%
4519 \GlsXtrNoGlsWarningEmptyMain\par
4520 \medskip
4521 \noindent\texttt{\string\usepackage[nomain]%
4522 \glstopenbrace glossaries-extra\glstclosebrace}
4523 \medskip
4524 }%
4525 {}%
4526
4527 \ifdequal\makeglossaries\@no@makeglossaries
4528 {%
4529 \GlsXtrNoGlsWarningMisMatch
4530 }%
4531 {}%
4532
4533 \GlsXtrNoGlsWarningBuildInfo

```

```

4533     }%
4534   \fi
4535 }%
4536 {%
4537   \GlsXtrNoGlsWarningNoOut
4538   {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4539 }%
4540 }%
4541 \par
4542 \GlsXtrNoGlsWarningTail
4543 }

```

`glossarywarning` Warn about using `\printglossary` with `record`

```

4544 \newcommand*{\@glxtr@record@noglossarywarning}[1]{%
4545   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4546   with record=only package option\MessageBreak(use
4547   \string\printunsrtglossary[type=#1])\MessageBreak
4548   instead (or change the package option)}%
4549   \glossarysection[\glossarytoctitle]{\glossarytitle}
4550   \GlsXtrRecordWarning{#1}
4551   \GlsXtrNoGlsWarningTail
4552 }

```

Provide some commands to accompany the `record` option for use with **bib2gls**.

`xltrresourcefile` 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.

```

4553 \newcommand*{\glxtrresourcefile}[2] []{%
  The record option can't be set after this command.
4554   \disable@keys{glossaries-extra.sty}{record}%
4555   \glxtr@writefields
4556   \protected@write\@auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%
4557   \let\@glxtr@org@see@noindex\@gl@see@noindex
4558   \let\@gl@see@noindex\relax
4559   \IfFileExists{#2.glstex}%
4560   {%

```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```

4561     \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4562     \makeatletter
4563     \@input{#2.glstex}%
4564     \@bibgls@restoreat
4565   }%
4566   {%
4567     \GlossariesExtraWarning{No file '#2.glstex'}%
4568   }%
4569   \let\@gl@see@noindex\@glxtr@org@see@noindex
4570 }

```

```

4571 \@onlypreamble\glxtrresourcefile

xtrresourceinit  Code used during the protected write operation.
4572 \newcommand*{\glxtrresourceinit}{%

trresourcecount
4573 \newcount\glxtrresourcecount

trLoadResources  Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.
4574 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
4575   \ifnum\glxtrresourcecount=0\relax
4576     \glxtrresourcefile[#1]{\jobname}%
4577   \else
4578     \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4579   \fi
4580   \advance\glxtrresourcecount by 1\relax
4581 }

glxtr@resource
4582 \newcommand*{\glxtr@resource}[2]{%

\glxtr@fields
4583 \newcommand*{\glxtr@fields}[1]{%

xtr@texencoding
4584 \newcommand*{\glxtr@texencoding}[1]{%

\glxtr@langtag
4585 \newcommand*{\glxtr@langtag}[1]{%

@pluralsuffixes
4586 \newcommand*{\glxtr@pluralsuffixes}[4]{%

tr@shortcutsval
4587 \newcommand*{\glxtr@shortcutsval}[1]{%

sxtr@linkprefix
4588 \newcommand*{\glxtr@linkprefix}[1]{%

xtr@writefields  This information only needs to be written once, so disable it after it's been used.
4589 \newcommand*{\glxtr@writefields}{%

4590   \protected@write\@auxout{%
4591     {\string\providecommand*{\string\glxtr@fields}[1]{}}%
4592   \protected@write\@auxout{%
4593     {\string\providecommand*{\string\glxtr@resource}[2]{}}%
4594   \protected@write\@auxout{%
4595     {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}%

```

```

4596 \protected@write\@auxout{}%
4597   {\string\providecommand*\string\glxtr@shortcutsval}[1]{}%
4598 \protected@write\@auxout{}%
4599   {\string\providecommand*\string\glxtr@linkprefix}[1]{}%
4600 \protected@write\@auxout{}\string\glxtr@fields{\@glx@keymap}}%

```

```

4601 \protected@write\@auxout{}%
4602   {\string\providecommand*\string\glxtr@record}[5]{}%

```

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`.

```

4603 \ifdef\CurrentTrackedLanguageTag
4604 {%
4605   \protected@write\@auxout{}%
4606     \string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
4607 }%
4608 {}%
4609 \protected@write\@auxout{}\string\glxtr@pluralsuffixes
4610   {\glxtr@pluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4611   {\glxtr@abbrvpluralsuffix}}%
4612 \ifdef\inputencodingname
4613 {%
4614   \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
4615 }%
4616 {%

```

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.)

```

4617   \@ifpackageloaded{fontspec}%
4618   {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}%
4619   {}%
4620 }%
4621 \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.

```

4622 \AtBeginDocument
4623   {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}%
4624 \let\glxtr@writefields\relax

```

If the `automake` option is on, try running `bib2gls` if the aux file exists. The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```

4625 \ifglxautomake
4626   \IfFileExists{\jobname.aux}%
4627   {\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`.


```

4628 \ifx\@gls@doautomake\@gls@doautomake@err
4629 \let\@gls@doautomake\relax
4630 \fi
4631 \fi
4632 }

```

do@automake@err

```

4633 \newcommand*{\@gls@doautomake@err}{%
4634 \PackageError{glossaries}{You must use
4635 \string\makeglossaries\space with automake=true}
4636 {%
4637 Either remove the automake=true setting or
4638 add \string\makeglossaries\space to your document preamble.%
4639 }%
4640 }

```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```

4641 \newcommand*{\glsxtr@record}[5]{%

```

r@counterrecord Aux file command.

```

4642 \newcommand*{\glsxtr@counterrecord}[3]{%
4643 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4644 }

```

unterrecordhook Hook used by \@glsxtr@dorecord.

```

4645 \newcommand*{\@glsxtr@counterrecordhook}{%

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

4646 \newcommand*{\GlsXtrRecordCounter}[1]{%
4647 \@glsxtr@recordcounter{#1}%
4648 }
4649 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

4650 \newcommand*{\@glsxtr@docounterrecord}[1]{%
4651 \protected@write\@auxout{}{\string\glsxtr@counterrecord
4652 {\@gls@label}{#1}{\csuse{the#1}}}%
4653 }

```

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.

```

4654 \newcommand*{\glsxtrglossentry}[1]{%

```

```

4655 \glxstrtitleorpdforheading
4656 {\@glxstrglossentry{#1}}%
4657 {\glstryname{#1}}%
4658 {\glxstrheadname{#1}}%
4659 }

```

`\glxstrglossentry` Another test is needed in case `\@glxstrglossentry` has been written to the table of contents.

```

4660 \newrobustcmd*{\@glxstrglossentry}[1]{%
4661   \glxstrtitleorpdforheading
4662   {%
4663     \glsdodefexists{#1}%
4664     {%
4665       \begingroup
4666         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4667         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4668         \ifglshasparent{#1}%
4669           {\GlsXtrStandaloneSubEntryItem{#1}}%
4670           {\glstryitem{#1}}%
4671           \glstarget{#1}{\glstryname{#1}}%
4672         \endgroup
4673       }%
4674     }%
4675     {\glstryname{#1}}%
4676     {\glxstrheadname{#1}}%
4677 }

```

`\oneGlossaryType` To make it easier to adjust the definition of `\currentglossary` within `\glxstrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4678 \newcommand{\GlsXtrStandaloneGlossaryType}{\glstrytype{\glscurrententrylabel}}

```

`\oneSubEntryItem` Used for sub-entries in standalone format. The argument is the entry's label.

```

4679 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4680   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
4681 }

```

`\glossentryother` As `\glxstrglossentry` 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.

```

4682 \newcommand*{\glxstrglossentryother}[3]{%
4683   \ifstrempy{#1}%
4684   {%
4685     \ifcsdef{glxstrhead#3}%
4686     {%
4687       \glxstrtitleorpdforheading
4688       {\@glxstrglossentryother{#2}{#3}{#1}}%

```

```

4689     {\@gls@entry@field{#2}{#3}}%
4690     {\csuse{glsxtrhead#3}{#2}}%
4691 }%
4692 {%
4693     \glsxtrtitleorpdforheading
4694     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4695     {\@gls@entry@field{#2}{#3}}%
4696     {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4697 }%
4698 }%
4699 {%
4700     \glsxtrtitleorpdforheading
4701     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4702     {\@gls@entry@field{#2}{#3}}%
4703     {#1}%
4704 }%
4705 }

```

`glossentryother` As `\@glsxtrglossentry` but uses a different field.

```

4706 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
4707     \glsxtrtitleorpdforheading
4708     {%
4709         \glsdoifexists{#1}%
4710         {%
4711             \begingroup
4712                 \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4713                 \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4714                 \ifglshasparent{#1}%
4715                     {\GlsXtrStandaloneSubEntryItem{#1}}%
4716                     {\glsentryitem{#1}}%
4717                     \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4718                 \endgroup
4719             }%
4720         }%
4721         {\@gls@entry@field{#1}{#2}}%
4722         {#3}%
4723 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4724 \newcommand*{\printunsrtglossary}{%
4725     \@ifstar\s@printunsrtglossary\@printunsrtglossary
4726 }

```

`ntunsrtglossary` Unstarred version.

```

4727 \newcommand*{\@printunsrtglossary}[1][\]{%
4728     \@printglossary{type=\glsdefaultttype,#1}{\@print@unsrt@glossary}%
4729 }

```

ntunsrtglossary Starred version.

```
4730 \newcommand*{\s@printunsrtglossary}[2][\{%  
4731   \begin{group}  
4732     #2%  
4733     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%  
4734   \end{group}  
4735 }
```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```
4736 \newcommand*{\printunsrtglossaries}{%  
4737   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%  
4738 }
```

@unsrt@glossary

```
4739 \newcommand*{\@print@unsrt@glossary}{%  
4740   \glossarysection[\glossarytoctitle]{\glossarytitle}%  
4741   \glossarypreamble  
  
   check for empty list  
4742   \glxtrifemptyglossary{\@glo@type}%  
4743   {%  
4744     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%  
4745   }%  
4746   {%  
  
4747     \key@ifundefined{glossentry}{group}%  
4748     {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%  
4749     {\let\@gls@getgrouptitle\@glxtr@unsrt@getgrouptitle}%  
4750     \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
4751   \def\@glxtr@doglossary{%  
4752     \begin{theglossary}%  
4753     \glossaryheader  
4754     \glsresetentrylist  
4755   }%  
4756   \expandafter\@for\expandafter\glscurrententrylabel\expandafter  
4757     :\expandafter=\csname glolist@\@glo@type\endcsname\do{%  
4758     \ifdefempty{\glscurrententrylabel}  
4759     }%  
4760     {%
```

Provide a hook (for example to measure width).

```
4761     \let\glxtr@process\@firstofone  
4762     \let\printunsrtglossaryskipentry  
4763     \@glxtr@printunsrtglossaryskipentry  
4764     \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```
4765 \glxtr@process
4766 {%
4767 \ifglshasparent{\glscurrententrylabel}{}%
4768 {%
4769 \@glxtr@checkgroup\glscurrententrylabel
4770 \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
4771 {\@glxtr@groupheading}%
4772 }%
4773 \eappto\@glxtr@doglossary{%
4774 \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4775 }%
4776 }%
4777 }%
4778 \appto\@glxtr@doglossary{\end{theglossary}}%
4779 \printunsrtglossarypredoglossary
4780 \@glxtr@doglossary
4781 }%
4782 \glossarypostamble
4783 }
```

entryprocesshook

```
4784 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}
```

glossaryskipentry

```
4785 \newcommand*{\printunsrtglossaryskipentry}{%
4786 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4787 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4788 }
```

entryprocesshook

```
4789 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
4790 \let\glxtr@process\@gobble
4791 }
```

rypredoglossary

```
4792 \newcommand*{\printunsrtglossarypredoglossary}{}%
```

lossary@handler

```
4793 \newcommand{\@printunsrt@glossary@handler}[1]{%
4794 \xdef\glscurrententrylabel{#1}%
4795 \printunsrtglossaryhandler\glscurrententrylabel
4796 }
```

glossaryhandler

```
4797 \newcommand{\printunsrtglossaryhandler}[1]{%
4798 \glxtrunsrtdo{#1}%
4799 }
```

xtriflabelinlist

```
\glxtriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```
4800 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4801   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
4802   \@glxtr@doiflabelinlist{#3}{#4}}%
4803 }
```

srtglossaryunit

```
4804 \newcommand{\print@op@unsrtglossaryunit}[2][{}]{%
4805   \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
4806     \printunsrtglossaryunitsetup{#2}}%
4807   }%
4808 }
```

ossaryunitsetup

```
4809 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4810   \renewcommand{\printunsrtglossaryhandler}[1]{%
4811     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4812     {\glxtrunsrtdo{##1}}}%
4813   }%
4814   }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glslinkprefix`. The `\@gobble` part discards `\glslinkprefix`.

```
4815   \ifcsundef{theH#1}%
4816     {%
4817       \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4818     }%
4819     {%
4820       \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4821     }%
4822     \renewcommand*{\glossarysection}[2][{}]{%
4823       \appto\glossarypostamble{\glspar\medskip\glspar}%
4824 }
```

srtglossaryunit

```
4825 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
4826   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4827     requires the record=only or record=alsoindex package option}{}%
4828 }
```

t@getgrouptitle

```
4829 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4830   \protected@edef\@glxtr@titlelabel{\glxtr@grouptitle@#1}%
4831 }
```

```

4831 \@onelevel@sanitize\@glxtr@titlelabel
4832 \ifcsdef{\@glxtr@titlelabel}
4833 {\letcs{#2}{\@glxtr@titlelabel}}%
4834 {\def#2{#1}}%
4835 }

```

`\glxtrunsrtdo` Provide a user-level call to `\@glxtr@noidx@do` to make it easier to define a new handler.

```

4836 \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).

```

4837 \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.

`glxtr@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.

```

4838 \newcommand*{\@glxtr@checkgroup}[1]{%
4839   \def\@glxtr@groupheading{}%
4840   \key@ifundefined{glossentry}{group}%
4841   {%
4842     \letcs{\@gls@sort}{glo@glstdetoklabel{#1}@sort}%
4843     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
4844   }%
4845   {%
4846     \protected@edef\@glo@thislettergrp{%
4847       \csuse{glo@glstdetoklabel{#1}@glxtrgroupfield}}%
4848     }%
4849     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4850     {}%
4851     {%
4852       \ifdefempty{\@gls@currentlettergroup}{}%
4853       {\def\@glxtr@groupheading{\@gls@groupskip}}%
4854       \eappto\@glxtr@groupheading{%
4855         \noexpand\@gls@groupheading{\expandonce\@glo@thislettergrp}%
4856       }%
4857     }%
4858     \let\@gls@currentlettergroup\@glo@thislettergrp
4859 }

```

`glxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```

4860 \newcommand{\@glsxtr@noidx@do}[1]{%
4861   \ifglstryexists{#1}%
4862   {%
4863     \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
4864     \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@location}%
4865     \ifglshasparent{#1}%
4866     {%
4867       \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
4868       \ifdefvoid{\@gls@location}%
4869       {%
4870         \ifdefvoid{\@gls@loclist}%
4871         {%
4872           \subglossentry{\gls@level}{#1}{}%
4873         }%
4874         {%
4875           \subglossentry{\gls@level}{#1}%
4876           {%
4877             \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4878           }%
4879         }%
4880       }%
4881       {%
4882         \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4883       }%
4884     }%
4885   {%
4886     \ifdefvoid{\@gls@location}%
4887     {%
4888       \ifdefvoid{\@gls@loclist}
4889       {%
4890         \glossentry{#1}{}%
4891       }%
4892       {%
4893         \glossentry{#1}%
4894         {%
4895           \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4896         }%
4897       }%
4898     }%
4899     {%
4900       \glossentry{#1}%
4901       {%
4902         \glossaryentrynumbers{\@gls@location}%
4903       }%
4904     }%
4905   }%
4906 }%
4907 {}%
4908 }

```


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`

4909 `\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.)

`\@glsxtrnewgls`

`\glsxtrnewgls[\options]{\prefix}{\cs}{\inner cs name}`

```

4910 \newcommand*\@glsxtrnewgls[4]{%
4911   \ifdef{#3}%
4912   {%
4913     \PackageError{glossaries-extra}{Command \string#3\space already
4914 defined}{}%
4915   }%
4916   {%
4917     \ifcsdef{@#4like@#2}%
4918     {%
4919       \advance\@glsxtrnewgls@inner by \@ne
4920       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4921     }%
4922     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4923     \expandafter\newrobustcmd\expandafter*\expandafter
4924       #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4925     \ifstrempy{#1}%
4926     {%
4927       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
4928         \new@ifnextchar[%
4929           {\csname @#4@\endcsname{##1}{#2##2}}%
4930           {\csname @#4@\endcsname{##1}{#2##2}[]}%
4931         }%
4932       }%
4933     {%
4934       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
4935         \new@ifnextchar[%
4936           {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4937           {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
4938         }%
4939       }%
4940     }%
4941 }

```

`\glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```
4942 \newrobustcmd*{\glxstrnewgls}[3] [] {%
4943   \@glxstrnewgls{#1}{#2}{#3}{gls}%
4944 }
```

`\glxstrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4945 \newrobustcmd*{\glxstrnewglslike}[6] [] {%
4946   \@glxstrnewgls{#1}{#2}{#3}{gls}%
4947   \@glxstrnewgls{#1}{#2}{#4}{glspl}%
4948   \@glxstrnewgls{#1}{#2}{#5}{Gls}%
4949   \@glxstrnewgls{#1}{#2}{#6}{Glspl}%
4950 }
```

`\glxstrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4951 \newrobustcmd*{\glxstrnewGLSlike}[4] [] {%
4952   \@glxstrnewgls{#1}{#2}{#3}{GLS}%
4953   \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
4954 }
```

`\glxstrnewrgls` As `\glxstrnewgls` but for `\rgls`.

```
4955 \newrobustcmd*{\glxstrnewrgls}[3] [] {%
4956   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4957 }
```

`\glxstrnewrglslike` As `\glxstrnewglslike` but for `\rgls` etc.

```
4958 \newrobustcmd*{\glxstrnewrglslike}[6] [] {%
4959   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4960   \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
4961   \@glxstrnewgls{#1}{#2}{#5}{rGls}%
4962   \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
4963 }
```

`\glxstrnewrGLSlike` As `\glxstrnewGLSlike` but for `\rGLS` etc.

```
4964 \newrobustcmd*{\glxstrnewrGLSlike}[4] [] {%
4965   \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
4966   \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
4967 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
4968 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
4969 \ifcsdef{glo@glstoklabel{#1}@recordcount}%
4970 {\csname glo@glstoklabel{#1}@recordcount\endcsname}%
4971 {0}%
4972 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4973 \newcommand*{\GlsXtrRecordCount}[2]{%
4974 \ifcsdef{glo@glstoklabel{#1}@recordcount.#2}%
4975 {\csname glo@glstoklabel{#1}@recordcount.#2\endcsname}%
4976 {0}%
4977 }
```

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 \glstoklocation can be set to something sensible.

```
4978 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4979 \ifcsdef{glo@glstoklabel{#1}@recordcount.#2.\glstoklocation{#3}}%
4980 {\csname glo@glstoklabel{#1}@recordcount.#2.\glstoklocation{#3}\endcsname}%
4981 {0}%
4982 }
```

trdetoklocation

```
4983 \newcommand*{\glstoklocation}[1]{#1}
```

ablerecordcount

```
4984 \newcommand*{\glstrenablerecordcount}{%
4985 \renewcommand*{\gls}{\rgls}%
4986 \renewcommand*{\Gls}{\rGls}%
4987 \renewcommand*{\glspl}{\rglspl}%
4988 \renewcommand*{\Glspl}{\rGlspl}%
4989 \renewcommand*{\GLS}{\rGLS}%
4990 \renewcommand*{\GLSpl}{\rGLSpl}%
4991 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
4992 \newcommand*{\glstxrrecordtriggervalue}[1]{%
4993 \GlsXtrTotalRecordCount{#1}%
4994 }
```

dCountAttribute

```
4995 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4996 \@for\@glstxr@cat:=#1\do
4997 {%
4998 \ifdefempty{\@glstxr@cat}{}%
4999 }
```

```

4999   {%
5000     \glssetcategoryattribute{\@glstr@cat}{recordcount}{#2}%
5001   }%
5002 }%
5003 }

```

\glstrifrecordtrigger{<label>}{<trigger format>}{<normal>}

```

5004 \newcommand*{\glstrifrecordtrigger}[3]{%
5005   \glshasattribute{#1}{recordcount}%
5006   {%
5007     \ifnum\glstrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
5008     #3%
5009   \else
5010     #2%
5011   \fi
5012 }%
5013 {#3}%
5014 }

```

```

5015 \newcommand*{\@glstr@rglstrigger@record}[3]{%
5016   \edef\glslabel{\glsdetoklabel{#2}}%
5017   \let\@gls@link@label\glslabel
5018   \def\@glstr@thevalue{%
5019     \def\@glstr@theHvalue{\@glstr@thevalue}%
5020     \def\@glsnumberformat{glstriggerrecordformat}%
5021     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5022     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5023     \def\@glstr@thevalue{%
5024       \def\@glstr@theHvalue{\@glstr@thevalue}%
5025       \glstrinitwrgloss
5026       \glslinkpresetkeys
5027       \setkeys{glslink}{#1}%
5028       \glslinkpostsetkeys
5029       \ifdefempty{\@glstr@thevalue}%
5030       {%
5031         \@gls@saveentrycounter
5032       }%
5033     }%
5034     \let\thegl Sentrycounter\@glstr@thevalue
5035     \def\theHglSentrycounter{\@glstr@theHvalue}%
5036   }%
5037   \ifglstrinitwrglossbefore
5038     \do@wrglossary{#2}%
5039   \fi

```

```

5040 #3%
5041 \ifglxtrinitwrglossbefore
5042 \else
5043   \@do@wrglossary{#2}%
5044 \fi
5045 \ifKV@glslink@local
5046   \glsllocalunset{#2}%
5047 \else
5048   \glsunset{#2}%
5049 \fi
5050 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5051 \newcommand*{\glstriggerrecordformat}[1]{%

```

\rgls

```

5052 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

5053 \newcommand*{\@rgls}[2][{}]{%
5054   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5055 }

```

\@rgls@

```

5056 \def\@rgls@#1#2[#3]{%
5057   \glxtrifrecordtrigger{#2}%
5058   {%
5059     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5060   }%
5061   {%
5062     \@gls@{#1}{#2}[#3]%
5063   }%
5064 }%

```

\rglsp1

```

5065 \newrobustcmd*{\rglsp1}{\@gls@hyp@opt\@rglsp1}

```

\@rglsp1

```

5066 \newcommand*{\@rglsp1}[2][{}]{%
5067   \new@ifnextchar[{\@rglsp1@{#1}{#2}}{\@rglsp1@{#1}{#2}[]}%
5068 }

```

\@rglsp1@

```

5069 \def\@rglsp1@#1#2[#3]{%
5070   \glxtrifrecordtrigger{#2}%
5071   {%
5072     \@glxtr@rglstrigger@record{#1}{#2}{\rglsp1format{#2}{#3}}%

```

```

5073 }%
5074 {%
5075   \@glspl@{#1}{#2}[#3]%
5076 }%
5077 }%

```

\rGls

```

5078 \newrobustcmd*{\rGls}{\@gls@hyp@opt\rGls}

```

\@rGls

```

5079 \newcommand*{\@rGls}[2][{}]{%
5080   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5081 }

```

\@rGls@

```

5082 \def\@rGls@#1#2[#3]{%
5083   \glsxtrifrecordtrigger{#2}%
5084   {%
5085     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5086   }%
5087   {%
5088     \@Gls@{#1}{#2}[#3]%
5089   }%
5090 }%

```

\rGlspl

```

5091 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

```

\@rGlspl

```

5092 \newcommand*{\@rGlspl}[2][{}]{%
5093   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5094 }

```

\@rGlspl@

```

5095 \def\@rGlspl@#1#2[#3]{%
5096   \glsxtrifrecordtrigger{#2}%
5097   {%
5098     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5099   }%
5100   {%
5101     \@Glspl@{#1}{#2}[#3]%
5102   }%
5103 }%

```

\rGLS

```

5104 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

```

\@rGLS

```
5105 \newcommand*{\@rGLS}[2][\%  
5106   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}}]{\%  
5107 }
```

\@rGLS@

```
5108 \def\@rGLS@#1#2[#3]{\%  
5109   \glstrifrecordtrigger{#2}%  
5110   {\%  
5111     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%  
5112   }%  
5113   {\%  
5114     \@GLS@{#1}{#2}[#3}%  
5115   }%  
5116 }%
```

\rGLSpl

```
5117 \newrobustcmd*{\rGLSpl}{\@glshyp@opt\rGLSpl}
```

\@rGLSpl

```
5118 \newcommand*{\@rGLSpl}[2][\%  
5119   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}}]{\%  
5120 }
```

\@rGLSpl@

```
5121 \def\@rGLSpl@#1#2[#3]{\%  
5122   \glstrifrecordtrigger{#2}%  
5123   {\%  
5124     \@glstr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%  
5125   }%  
5126   {\%  
5127     \@GLSpl@{#1}{#2}[#3}%  
5128   }%  
5129 }%
```

\rglsformat

```
5130 \newcommand*{\rglsformat}[2]{\%  
5131   \glrifregular{#1}  
5132   {\glentryfirst{#1}}%  
5133   {\ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}}%#2%  
5134 }
```

\rglsplformat

```
5135 \newcommand*{\rglsplformat}[2]{\%  
5136   \glrifregular{#1}  
5137   {\glentryfirstplural{#1}}%  
5138   {\ifglshaslong{#1}{\glentrylongplural{#1}}{\glentryfirstplural{#1}}}%#2%  
5139 }
```

\rGlsformat

```
5140 \newcommand*\rGlsformat}[2]{%
5141   \glsifregular{#1}
5142   {\Glsentryfirst{#1}}%
5143   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5144 }
```

\rGlsplformat

```
5145 \newcommand*\rGlsplformat}[2]{%
5146   \glsifregular{#1}
5147   {\Glsentryfirstplural{#1}}%
5148   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5149 }
```

\rGLSformat

```
5150 \newcommand*\rGLSformat}[2]{%
5151   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsformat{#1}{#2}}%
5152 }
```

\rGLSplformat

```
5153 \newcommand*\rGLSplformat}[2]{%
5154   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsplformat{#1}{#2}}%
5155 }
```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into \@gls@link (through \glsxtr@inc@linkcount) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The “link” part of the name refers to \@gls@link not \hyperlink.)

\@inc@linkcount This performs the actual incrementing and counter definition. The counter is given by \c@glsxtr@linkcount@<label> where <label> is the entry’s label. Since this is performed within \@gls@link the label can be accessed with \glslabel.

```
5156 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
5157   \glsifattribute{\glslabel}{linkcount}{true}%
5158   {%
    Does the counter exist?
5159     \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5160     {%
      Counter doesn't exist, so define it.
5161       \newcounter{glsxtr@linkcount@\glslabel}%
```


If linkcountmaster is set, add to counter reset.

```
5162 \glshasattribute{\glslabel}{linkcountmaster}%
5163 {%
```

Need to ensure values are fully expanded.

```
5164 \begingroup
5165 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@glslabel}%
5166 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
5167 \x
5168 }%
5169 {}%
5170 }%
```

Increment counter:

```
5171 \glsxtrinlinkcounter{glsxtr@linkcount@glslabel}%
5172 }%
5173 {}%
5174 }
```

`\refstepcounter` May be redefined to use `\refstepcounter` if required.

```
5175 \newcommand*{\glsxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkcountervalue` Expands to the associated link counter register or 0 if not defined.

```
5176 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5177 \ifcsundef{c@glxtr@linkcount@#1}{0}{\csname c@glxtr@linkcount@#1\endcsname}%
5178 }
```

`\thelinkcounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5179 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5180 \ifcsundef{theglsxtr@linkcount@#1}{0}%
5181 {\csname theglxtr@linkcount@#1\endcsname}%
5182 }
```

`\iflinkcounterdef` Tests if the counter has been defined

```
5183 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5184 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5185 }
```

`\linkcountername` Expands to the associated link counter name. (No check for existence.)

```
5186 \newcommand*{\GlsXtrLinkCounterName}[1]{glsxtr@linkcount@#1}
```

`\enablelinkcounting` `\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
5187 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5188 \let\glsxtr@inc@linkcount\@glsxtr@do@inc@linkcount
```

```

5189 \@for\@glstr@label:=#2\do
5190 {%
5191   \glsetcategoryattribute{\@glstr@label}{linkcount}{true}%
5192   \ifstrempy{#1}{}%
5193   {%
5194     \ifcsundef{c@#1}%
5195     {\@nocounterr{#1}}%
5196     {\glsetcategoryattribute{\@glstr@label}{linkcountmaster}{#1}}%
5197   }%
5198 }%
5199 }
5200 \@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.

```

5201 \@ifpackageloaded{glossaries-accsupp}
5202 {

```

Define (or redefine) commands to use the accessibility information.

\glsaccessname Display the name value (no link and no check for existence).

```

5203   \newcommand*{\glsaccessname}[1]{%
5204     \glsnameaccessdisplay
5205     {%
5206       \glstentryname{#1}%
5207     }%
5208     {#1}%
5209   }

```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

5210   \newcommand*{\Glsaccessname}[1]{%
5211     \glsnameaccessdisplay
5212     {%
5213       \Glstentryname{#1}%
5214     }%
5215     {#1}%
5216   }

```

\GLSaccessname Display the name value (no link and no check for existence) converted to upper case.

```

5217   \newcommand*{\GLSaccessname}[1]{%
5218     \glsnameaccessdisplay
5219     {%
5220       \mfirstucMakeUppercase{\glstentryname{#1}}%

```

```

5221     }%
5222     {#1}%
5223 }

```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

5224 \newcommand*{\glsaccesstext}[1]{%
5225     \glstextaccessdisplay
5226     {%
5227         \glentrytext{#1}%
5228     }%
5229     {#1}%
5230 }

```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

5231 \newcommand*{\Glsaccesstext}[1]{%
5232     \glstextaccessdisplay
5233     {%
5234         \Glsentrytext{#1}%
5235     }%
5236     {#1}%
5237 }

```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5238 \newcommand*{\GLSaccesstext}[1]{%
5239     \glstextaccessdisplay
5240     {%
5241         \mfirstucMakeUppercase{\glentrytext{#1}}%
5242     }%
5243     {#1}%
5244 }

```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

5245 \newcommand*{\glsaccessplural}[1]{%
5246     \glspluralaccessdisplay
5247     {%
5248         \glentryplural{#1}%
5249     }%
5250     {#1}%
5251 }

```

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5252 \newcommand*{\Glsaccessplural}[1]{%
5253     \glspluralaccessdisplay
5254     {%
5255         \Glsentryplural{#1}%
5256     }%

```

```

5257     {#1}%
5258 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

5259 \newcommand*{\GLSaccessplural}[1]{%
5260   \glspluralaccessdisplay
5261   {%
5262     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5263   }%
5264   {#1}%
5265 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

5266 \newcommand*{\glsaccessfirst}[1]{%
5267   \glsfirstaccessdisplay
5268   {%
5269     \glsentryfirst{#1}%
5270   }%
5271   {#1}%
5272 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5273 \newcommand*{\Glsaccessfirst}[1]{%
5274   \glsfirstaccessdisplay
5275   {%
5276     \Glsentryfirst{#1}%
5277   }%
5278   {#1}%
5279 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

5280 \newcommand*{\GLSaccessfirst}[1]{%
5281   \glsfirstaccessdisplay
5282   {%
5283     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
5284   }%
5285   {#1}%
5286 }

```

cessfirstplural Display the firstplural value (no link and no check for existence).

```

5287 \newcommand*{\glsaccessfirstplural}[1]{%
5288   \glsfirstpluralaccessdisplay
5289   {%
5290     \glsentryfirstplural{#1}%
5291   }%
5292   {#1}%
5293 }

```

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5294 \newcommand*\Glsaccessfirstplural}[1]{%
5295   \glsfirstpluralaccessdisplay
5296   {%
5297     \Glsentryfirstplural{#1}%
5298   }%
5299   {#1}%
5300 }

```

cessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```

5301 \newcommand*\GLSaccessfirstplural}[1]{%
5302   \glsfirstpluralaccessdisplay
5303   {%
5304     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5305   }%
5306   {#1}%
5307 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

5308 \newcommand*\glsaccesssymbol}[1]{%
5309   \glssymbolaccessdisplay
5310   {%
5311     \glsentrysymbol{#1}%
5312   }%
5313   {#1}%
5314 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5315 \newcommand*\GLSaccesssymbol}[1]{%
5316   \glssymbolaccessdisplay
5317   {%
5318     \Glsentrysymbol{#1}%
5319   }%
5320   {#1}%
5321 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5322 \newcommand*\GLSaccesssymbol}[1]{%
5323   \glssymbolaccessdisplay
5324   {%
5325     \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5326   }%
5327   {#1}%
5328 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5329 \newcommand*{\glsaccesssymbolplural}[1]{%
5330   \glssymbolpluralaccessdisplay
5331   {%
5332     \glsentrysymbolplural{#1}%
5333   }%
5334   {#1}%
5335 }

```

`\esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5336 \newcommand*{\Glsaccesssymbolplural}[1]{%
5337   \glssymbolpluralaccessdisplay
5338   {%
5339     \Glsentrysymbolplural{#1}%
5340   }%
5341   {#1}%
5342 }

```

`\esssymbolplural` Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5343 \newcommand*{\GLSaccesssymbolplural}[1]{%
5344   \glssymbolpluralaccessdisplay
5345   {%
5346     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5347   }%
5348   {#1}%
5349 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5350 \newcommand*{\glsaccessdesc}[1]{%
5351   \glsdescriptionaccessdisplay
5352   {%
5353     \glsentrydesc{#1}%
5354   }%
5355   {#1}%
5356 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5357 \newcommand*{\Glsaccessdesc}[1]{%
5358   \glsdescriptionaccessdisplay
5359   {%
5360     \Glsentrydesc{#1}%
5361   }%
5362   {#1}%
5363 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5364 \newcommand*{\GLSaccessdesc}[1]{%

```

```

5365 \glsdescriptionaccessdisplay
5366 {%
5367     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5368 }%
5369 {#1}%
5370 }

```

`\glsaccessdescplural` Display the descplural value (no link and no check for existence).

```

5371 \newcommand*{\glsaccessdescplural}[1]{%
5372     \glsdescriptionpluralaccessdisplay
5373     {%
5374         \glsentrydescplural{#1}%
5375     }%
5376     {#1}%
5377 }

```

`\Glsaccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5378 \newcommand*{\Glsaccessdescplural}[1]{%
5379     \glsdescriptionpluralaccessdisplay
5380     {%
5381         \Glsentrydescplural{#1}%
5382     }%
5383     {#1}%
5384 }

```

`\glsaccessdescplurl` Display the descplurl value (no link and no check for existence) converted to upper case.

```

5385 \newcommand*{\GLSaccessdescplurl}[1]{%
5386     \glsdescriptionpluralaccessdisplay
5387     {%
5388         \mfirstucMakeUppercase{\glsentrydescplurl{#1}}%
5389     }%
5390     {#1}%
5391 }

```

`\glsaccesssshort` Display the short form (no link and no check for existence).

```

5392 \newcommand*{\glsaccesssshort}[1]{%
5393     \glsshortaccessdisplay
5394     {%
5395         \glsentryshort{#1}%
5396     }%
5397     {#1}%
5398 }

```

`\Glsaccesssshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5399 \newcommand*{\Glsaccesssshort}[1]{%
5400     \glsshortaccessdisplay

```

```

5401     {%
5402         \Glsentryshort{#1}%
5403     }%
5404     {#1}%
5405 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5406 \newcommand*{\GLSaccessshort}[1]{%
5407     \glsshortaccessdisplay
5408     {%
5409         \mfirstucMakeUppercase{\Glsentryshort{#1}}%
5410     }%
5411     {#1}%
5412 }

```

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5413 \newcommand*{\lsaccessshortpl}[1]{%
5414     \glsshortpluralaccessdisplay
5415     {%
5416         \Glsentryshortpl{#1}%
5417     }%
5418     {#1}%
5419 }

```

`lSaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5420 \newcommand*{\lSaccessshortpl}[1]{%
5421     \glsshortpluralaccessdisplay
5422     {%
5423         \Glsentryshortpl{#1}%
5424     }%
5425     {#1}%
5426 }

```

`LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5427 \newcommand*{\LSaccessshortpl}[1]{%
5428     \glsshortpluralaccessdisplay
5429     {%
5430         \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5431     }%
5432     {#1}%
5433 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

5434 \newcommand*{\glsaccesslong}[1]{%
5435     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5436 }

```



```

\GLsaccesslong   Display the long form (no link and no check for existence).
5437
5438   \newcommand*{\GLsaccesslong}[1]{%
5439     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5440   }

\GLSaccesslong   Display the long value (no link and no check for existence) converted to upper case.
5441   \newcommand*{\GLSaccesslong}[1]{%
5442     \glslongaccessdisplay
5443     {%
5444       \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5445     }%
5446     {#1}%
5447   }

glsaccesslongpl   Display the long plural form (no link and no check for existence).
5448   \newcommand*{\glsaccesslongpl}[1]{%
5449     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5450   }

Glsaccesslongpl   Display the long plural form (no link and no check for existence).
5451
5452   \newcommand*{\Glsaccesslongpl}[1]{%
5453     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5454   }

GLSaccesslongpl   Display the longplural value (no link and no check for existence) converted to upper case.
5455   \newcommand*{\GLSaccesslongpl}[1]{%
5456     \glslongpluralaccessdisplay
5457     {%
5458       \mfirstucMakeUppercase{\Glsentrylongpl{#1}}%
5459     }%
5460     {#1}%
5461   }

    Keys for accessibility support.
5462   \define@key{glsxtrabrv}{access}{%
5463     \def\@gls@nameaccess{#1}%
5464   }

5465   \define@key{glsxtrabrv}{textaccess}{%
5466     \def\@gls@textaccess{#1}%
5467   }

5468   \define@key{glsxtrabrv}{firstaccess}{%
5469     \def\@gls@firstaccess{#1}%
5470   }

5471   \define@key{glsxtrabrv}{shortaccess}{%
5472     \def\@gls@shortaccess{#1}%
5473   }

```

```

5474 \define@key{glsextrabrv}{shortpluralaccess}{%
5475   \def\@gls@shortaccesspl{#1}%
5476 }

```

@initaccesskeys

```

5477 \newcommand*\@gls@initaccesskeys{%
5478   \def\@gls@nameaccess{%
5479     \def\@gls@textaccess{%
5480       \def\@gls@firstaccess{%
5481         \def\@gls@shortaccess{%
5482           \def\@gls@shortaccesspl{}%
5483         }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```

5484 \newcommand*\@gls@ifaccessattribute@set}[3]{%
5485   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5486   {#2}%
5487   {%
5488     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5489     {#3}%
5490     {%
5491       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5492       {#2}%
5493       {#3}%
5494     }%
5495   }%
5496 }

```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5497 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```

5498   \ifdefempty\@gls@shortaccess
5499   {%
5500     \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5501     {%
5502       \@glsextr@insertdots\@gls@shortaccess{#1}%
5503       \eappto\ExtraCustomAbbreviationFields{%
5504         shortaccess={\expandonce\@gls@shortaccess},}%
5505       }%
5506     }%
5507   }%
5508 }%

```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```

5509 \ifdefempty\@gls@shortaccess
5510 {}%
5511 {%
5512 \ifdefempty\@gls@shortaccesspl
5513 {%
5514 \@gls@ifaccessattribute@set{aposplural}%
5515 {%
5516 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5517 \@gls@shortaccess'\abbrvpluralsuffix}%
5518 }%
5519 {%
5520 \@gls@ifaccessattribute@set{noshortplural}%
5521 {%
5522 \let\@gls@shortaccesspl\@gls@shortaccess
5523 }%
5524 {%
5525 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5526 \@gls@shortaccess\abbrvpluralsuffix}%
5527 }%
5528 }%
5529 \eappto\ExtraCustomAbbreviationFields{%
5530 shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5531 }%
5532 {}%
5533 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5534 \ifdefempty\@gls@nameaccess
5535 {%
5536 \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5537 {%

```

Do nothing if the shortaccess key hasn't been set.

```

5538 \ifdefempty\@gls@shortaccess
5539 {}%
5540 {%
5541 \eappto\ExtraCustomAbbreviationFields{%
5542 access={\expandonce\@gls@shortaccess},%
5543 }%
5544 }%
5545 }%
5546 {}%
5547 }%
5548 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5549 \ifdefempty\@gls@textaccess
5550 {%
5551 \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5552 {%

```

Do nothing if the shortaccess key hasn't been set.

```

5553     \ifdefempty\@gls@shortaccess
5554     {}%
5555     {%
5556         \eappto\ExtraCustomAbbreviationFields{%
5557             textaccess={\expandonce\@gls@shortaccess},%
5558         }%
5559     }%
5560 }%
5561 {}%
5562 }%
5563 {}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```

5564     \ifdefempty\@gls@firstaccess
5565     {%
5566         \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5567         {%
```

Do nothing if the shortaccess key hasn't been set.

```

5568         \ifdefempty\@gls@shortaccess
5569         {}%
5570         {%
5571             \eappto\ExtraCustomAbbreviationFields{%
5572                 firstaccess={\expandonce\@gls@shortaccess},%
5573             }%
5574         }%
5575     }%
5576     {}%
5577 }%
5578 {}%
5579 }
```

End of if accsupp part

```

5580 }
5581 {
```

No accessibility support. Just define these commands to do \glsentry<xxx>

\glsaccessname Display the name value (no link and no check for existence).

```

5582 \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.

```

5583 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```

5584 \newcommand*{\GLSaccessname}[1]{%
5585     \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

`\glsaccesstext` Display the text value (no link and no check for existence).
5586 `\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.
5587 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
5588 `\newcommand*{\GLSaccesstext}[1]{%`
5589 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
5590 `\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.
5591 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
5592 `\newcommand*{\GLSaccessplural}[1]{%`
5593 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
5594 `\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.
5595 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
5596 `\newcommand*{\GLSaccessfirst}[1]{%`
5597 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`glsaccessfirstplural` Display the firstplural value (no link and no check for existence).
5598 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`Glsaccessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5599 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`GLSaccessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5600 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5601 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).
5602 `\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.

5603 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`\GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

5604 `\newcommand*{\GLSaccesssymbol}[1]{%`
5605 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`\esssymbolplural` Display the symbolplural value (no link and no check for existence).

5606 `\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.

5607 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`\esssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.

5608 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5609 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).

5610 `\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.

5611 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.

5612 `\newcommand*{\GLSaccessdesc}[1]{%`
5613 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence).

5614 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

5615 `\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.

5616 `\newcommand*{\GLSaccessdescplural}[1]{%`
5617 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).

5618 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

5619 `\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.

5620 `\newcommand*{\GLSaccessshort}[1]{%`
5621 `\protect\mfirstucMakeUppercase{\glsentryshort{#1}}}`

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

5622 `\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}`

`lSaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

5623 `\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.

5624 `\newcommand*{\GLSaccessshortpl}[1]{%`
5625 `\protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).

5626 `\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}`

`\Glsaccesslong` Display the long form (no link and no check for existence).

5627 `\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

5628 `\newcommand*{\GLSaccesslong}[1]{%`
5629 `\protect\mfirstucMakeUppercase{\glsentrylong{#1}}}`

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

5630 `\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}`

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

5631 `\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}`

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

5632 `\newcommand*{\GLSaccesslongpl}[1]{%`
5633 `\protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}`

`@initaccesskeys` This does nothing if there's no accessibility support.

5634 `\newcommand*{\@gls@initaccesskeys}{}`

`lt@short@access` This does nothing if there's no accessibility support.

5635 `\newcommand{\@gls@setup@default@short@access}[1]{}%`
End of else part
5636 }

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
5637 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
5638 \newcommand{\glsifcategory}[4]{%
5639   \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
5640 }
```

Categories can have attributes.

`categoryattribute` `\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```
5641 \newcommand*{\glssetcategoryattribute}[3]{%
5642   \csdef{@glxtr@categoryattr@@#1@#2}{#3}%
5643 }
```

`categoryattribute` `\glsgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5644 \newcommand*{\glsgetcategoryattribute}[2]{%
5645   \csuse{@glxtr@categoryattr@@#1@#2}%
5646 }
```

`categoryattribute` `\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
5647 \newcommand*{\glshascategoryattribute}[4]{%
5648   \ifcvoid{@glxtr@categoryattr@@#1@#2}{#4}{#3}%
5649 }
```

`\glssetattribute` `\glssetattribute{<entry label>}{<attribute-label>}{<value>}`

Short cut where the category label is obtained from the entry information.

```
5650 \newcommand*{\glssetattribute}[3]{%
```



```

5651 \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5652 }

```

`\glsggetattribute` `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```

5653 \newcommand*{\glsggetattribute}[2]{%
5654 \glsssetcategoryattribute{\glscategory{#1}}{#2}%
5655 }

```

`\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.

```

5656 \newcommand*{\glshasattribute}[4]{%
5657 \ifglssentryexists{#1}%
5658 {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5659 {#4}%
5660 }

```

`\glssifcategoryattribute` `\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```

5661 \newcommand{\glssifcategoryattribute}[5]{%
5662 \ifcsundef{@glssxtr@categoryattr@#1@#2}%
5663 {#5}%
5664 {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5665 }

```

`\glssifattribute` `\glssifattribute{<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.

```

5666 \newcommand{\glssifattribute}[5]{%
5667 \ifglssentryexists{#1}%
5668 {\glssifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5669 {#5}%
5670 }

```

Set attributes for the default general category:

```
5671 \glsssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5672 \glsssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
5673 \newcommand*{\glsssetregularcategory}[1]{%  
5674   \glsssetcategoryattribute{#1}{regular}{true}}%  
5675 }
```

`ifregularcategory` `\glssifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5676 \newcommand{\glssifregularcategory}[3]{%  
5677   \glssifcategoryattribute{#1}{regular}{true}{#2}{#3}}%  
5678 }
```

`ifnotregularcategory` `\glssifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5679 \newcommand{\glssifnotregularcategory}[3]{%  
5680   \glssifcategoryattribute{#1}{regular}{false}{#2}{#3}}%  
5681 }
```

`\glssifregular` `\glssifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5682 \newcommand{\glssifregular}[3]{%  
5683   \glssifregularcategory{\glsscategor{#1}}{#2}{#3}}%  
5684 }
```

`\glssifnotregular` `\glssifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5685 \newcommand{\glssifnotregular}[3]{%  
5686   \glssifnotregularcategory{\glsscategor{#1}}{#2}{#3}}%  
5687 }
```

oreachincategory

```
\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}
{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category matches *⟨category-label⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
5688 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5689   \forallglossaries[#1]{#3}%
5690   {%
5691     \forlgsentries[#3]{#4}%
5692     {%
5693       \glsifcategory{#4}{#2}{#5}{}%
5694     }%
5695   }%
5696 }
```

achwithattribute

```
\glsforeachwithattribute[⟨glossary labels⟩]{⟨attribute-label⟩}
{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category attribute *⟨attribute-label⟩* matches *⟨attribute-value⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
5697 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5698   \forallglossaries[#1]{#4}%
5699   {%
5700     \forlgsentries[#4]{#5}%
5701     {%
5702       \glsifattribute{#5}{#2}{#3}{#6}{}%
5703     }%
5704   }%
5705 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```
5706 \ifdef\newterm
5707 {%
```

`\newterm`

```
5708   \renewcommand*{\newterm}[2][ ]{%
5709     \newglossaryentry{#2}%
5710     {type={index},category=index,name={#2},%
```

```

5711     description={\glxtrpostdescription\nopostdesc},#1}%
5712 }

```

Indexed terms are regular by default.

```

5713 \glissetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5714 \newcommand*{\glxtrpostdescindex}{}

5715 }
5716 {}

```

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.

```

5717 \ifdef\printsymbols
5718 {%

```

glxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```

5719 \newcommand*{\glxtrnewsymbol}[3] [] {%
5720 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5721 }

```

Symbols are regular by default.

```

5722 \glissetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

5723 \newcommand*{\glxtrpostdescsymbol}{}

5724 }
5725 {}

```

Similar for the numbers option.

```

5726 \ifdef\printnumbers
5727 {%

```

glxtrnewnumber

```

5728 \ifdef\printnumbers
5729 \newcommand*{\glxtrnewnumber}[3] [] {%
5730 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5731 }

```

Numbers are regular by default.

```

5732 \glissetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

5733 \newcommand*{\glxtrpostdescnumber}{}

```

```
5734 }
5735 {}
```

`\glstrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5736 \newcommand*{\glstrsetcategory}[2]{%
5737   \@for\@glstr@label:=#1\do
5738   {%
5739     \glsfieldxdef{\@glstr@label}{category}{#2}%
5740   }%
5741 }
```

`\glstrcategoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
5742 \newcommand*{\glstrsetcategoryforall}[2]{%
5743   \forallglossaries[#1]{\@glstr@type}{%
5744     \forallsentries[\@glstr@type]{\@glstr@label}%
5745     {%
5746       \glsfieldxdef{\@glstr@label}{category}{#2}%
5747     }%
5748   }%
5749 }
```

`\glstrfieldtitlecase` `\glstrfieldtitlecase{\langle label \rangle}{\langle field \rangle}`

Apply title casing to the contents of the given field.

```
5750 \newcommand*{\glstrfieldtitlecase}[2]{%
5751   \expandafter\glstrfieldtitlecasesecs\expandafter
5752   {\csname glo@glsetoklabel{#1}@#2\endcsname}%
5753 }
```

`\glstrfieldtitlecasesecs` The command used by `\glstrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
5754 \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.

```
5755 \@ifpackageloaded{glossaries-accsupp}
5756 {
5757   \renewcommand*{\glossentrydesc}[1]{%
5758     \glsdoifexistsorwarn{#1}%
5759     {%
5760       \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

5761 \glshasattribute{#1}{glossdescfont}%
5762 {%
5763 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5764 \ifcsdef{\@glxtr@attrval}%
5765 {%
5766 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5767 }%
5768 {%
5769 \GlossariesExtraWarning{Unknown control sequence name
5770 '\@glxtr@attrval' supplied in glossdescfont attribute
5771 for entry '#1'. Ignoring}%
5772 \let\@glxtr@glossdescfont\@firstofone
5773 }%
5774 }%
5775 {\let\@glxtr@glossdescfont\@firstofone}%
5776 \glusifattribute{#1}{glossdesc}{firstuc}%
5777 {%
5778 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5779 }%
5780 {%
5781 \glusifattribute{#1}{glossdesc}{title}%
5782 {%
5783 \@glxtr@do@titlecaps@warn
5784 \glsdescriptionaccessdisplay
5785 {%
5786 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5787 }%
5788 {#1}%
5789 }%
5790 {%
5791 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5792 }%
5793 }%
5794 }%
5795 }
5796 }
5797 {
5798 \renewcommand*{\glossentrydesc}[1]{%
5799 \glsdoifexistsorwarn{#1}%
5800 {%
5801 \glissetabbrvfmt{\glscategory{#1}}%
5802 \glshasattribute{#1}{glossdescfont}%
5803 {%
5804 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5805 \ifcsdef{\@glxtr@attrval}%
5806 {%
5807 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5808 }%

```

```

5809      {%
5810      \GlossariesExtraWarning{Unknown control sequence name
5811      '\@glsxtr@attrval' supplied in glossdescfont attribute
5812      for entry '#1'. Ignoring}%
5813      \let\@glsxtr@glossdescfont\@firstofone
5814      }%
5815    }%
5816    {\let\@glsxtr@glossdescfont\@firstofone}%
5817    \glsifattribute{#1}{glossdesc}{firstuc}%
5818    {%
5819      \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5820    }%
5821    {%
5822      \glsifattribute{#1}{glossdesc}{title}%
5823      {%
5824        \@glsxtr@do@titlecaps@warn
5825        \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5826      }%
5827      {%
5828        \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5829      }%
5830    }%
5831  }%
5832 }
5833 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5834 \@ifpackageloaded{glossaries-accsupp}
5835 {
5836   \renewcommand*{\glossentryname}[1]{%
5837     \@glsdoifexistsorwarn{#1}%
5838     {%
5839       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5840   \glshasattribute{#1}{glossnamefont}%
5841   {%
5842     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5843     \ifcsdef{\@glsxtr@attrval}%
5844     {%
5845       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5846     }%
5847     {%
5848       \GlossariesExtraWarning{Unknown control sequence name
5849       '\@glsxtr@attrval' supplied in glossnamefont attribute
5850       for entry '#1'. Reverting to default \string\glsnamefont}%
5851       \let\@glsxtr@glossnamefont\glsnamefont
5852     }%
5853   }%

```

```

5854     {\let\@glsxtr@glossnamefont\glsnamefont}%
5855     \glsifattribute{#1}{glossname}{firstuc}%
5856     {%
5857         \glsnameaccessdisplay
5858         {%
5859             \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5860         }%
5861         {#1}%
5862     }%
5863     {%
5864         \glsifattribute{#1}{glossname}{title}%
5865         {%
5866             \@glsxtr@do@titlecaps@warn
5867             \glsnameaccessdisplay
5868             {%
5869                 \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
5870             }%
5871             {#1}%
5872         }%
5873         {%
5874             \glsifattribute{#1}{glossname}{uc}%
5875             {%
5876                 \glsnameaccessdisplay
5877                 {%

```

Hide the label from the upper-casing command.

```

5878             \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5879             \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5880         }%
5881         {#1}%
5882     }%
5883     {%
5884         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5885         \glsnameaccessdisplay
5886         {%
5887             \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
5888         }%
5889         {#1}%
5890     }%
5891 }%
5892 }%

```

Do post-name hook:

```

5893     \glsxtrpostnamehook{#1}%
5894 }%
5895 }
5896 }
5897 {
5898     \renewcommand*{\glossentryname}[1]{%
5899         \@glsdoifexistsorwarn{#1}%

```



```

5900 {%
5901   \glsssetabbrvfmt{\glscategory{#1}}%
5902   \glshasattribute{#1}{glossnamefont}%
5903   {%
5904     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5905     \ifcsdef{\@glxtr@attrval}%
5906     {%
5907       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5908     }%
5909     {%
5910       \GlossariesExtraWarning{Unknown control sequence name
5911         '\@glxtr@attrval' supplied in glossnamefont attribute
5912         for entry '#1'. Reverting to default \string\glssnamefont}%
5913       \let\@glxtr@glossnamefont\glssnamefont
5914     }%
5915   }%
5916   {\let\@glxtr@glossnamefont\glssnamefont}%
5917   \glssifattribute{#1}{glossname}{firstuc}%
5918   {%
5919     \@glxtr@glossnamefont{\Glsentryname{#1}}%
5920   }%
5921   {%
5922     \glssifattribute{#1}{glossname}{title}%
5923     {%
5924       \@glxtr@do@titlecaps@warn
5925       \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5926     }%
5927     {%
5928       \glssifattribute{#1}{glossname}{uc}%
5929     }%

```

Hide the label from the upper-casing command.

```

5930       \letcs{\glo@name}{\glo@glssdetoklabel{#1}@name}%
5931       \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5932     }%
5933   {%

```

This little trick is used by glossaries to allow the user to redefine \glssnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```

5934       \letcs{\glo@name}{\glo@glssdetoklabel{#1}@name}%
5935       \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5936     }%
5937   }%
5938 }%

```

Do post-name hook.

```

5939   \glxtrpostnamehook{#1}%
5940 }%
5941 }
5942 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```

5943 \ifpackageloaded{glossaries-accsupp}
5944 {
5945   \renewcommand*{\Glossentryname}[1]{%
5946     \@glsdoifexistsorwarn{#1}%
5947     {%
5948       \glsetabbrvfmt{\glscategory{#1}}%
5949       \glshasattribute{#1}{glossnamefont}%
5950       {%
5951         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5952         \ifcsdef{\@glstr@attrval}%
5953         {%
5954           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5955         }%
5956         {%
5957           \GlossariesExtraWarning{Unknown control sequence name
5958             '\@glstr@attrval' supplied in glossnamefont attribute
5959             for entry '#1'. Reverting to default \string\glnamefont}%
5960           \let\@glstr@glossnamefont\glnamefont
5961         }%
5962       }%
5963       {\let\@glstr@glossnamefont\glnamefont}%
5964       \glnameaccessdisplay
5965       {%
5966         \@glstr@glossnamefont{\Glsentryname{#1}}%
5967       }%
5968     }%

```

Do post-name hook:

```

5969   \glstrpostnamehook{#1}%
5970 }%
5971 }
5972 }
5973 {
5974   \renewcommand*{\Glossentryname}[1]{%
5975     \@glsdoifexistsorwarn{#1}%
5976     {%
5977       \glsetabbrvfmt{\glscategory{#1}}%
5978       \glshasattribute{#1}{glossnamefont}%
5979       {%
5980         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5981         \ifcsdef{\@glstr@attrval}%
5982         {%
5983           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5984         }%
5985         {%
5986           \GlossariesExtraWarning{Unknown control sequence name
5987             '\@glstr@attrval' supplied in glossnamefont attribute

```

```

5988         for entry '#1'. Reverting to default \string\glsnamefont}%
5989         \let\@glsxtr@glossnamefont\glsnamefont
5990     }%
5991 }%
5992 {\let\@glsxtr@glossnamefont\glsnamefont}%
5993 \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

5994     \glsxtrpostnamehook{#1}%
5995 }%
5996 }
5997 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

5998 \newcommand*{\glsxtrpostnamehook}[1]{%
5999     \let\@glsnumberformat\@glsxtr@defaultnumberformat
6000     \glsxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

6001     \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

6002     \csuse{glsxtrpostname\glscategory{#1}}%
6003 }

```

`trapostnamehook`

```

6004 \newcommand*{\glsextrapostnamehook}[1]{%

```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```

6005 \newcommand*{\glsdefpostname}[2]{%
6006     \csdef{glsxtrpostname#1}{#2}%
6007 }

```

`etaccessdisplay`

```

6008 \@ifpackageloaded{glossaries-accsupp}
6009 {
6010     \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6011         \ifcsdef{gls#1accessdisplay}%
6012         {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
6013         {%

```

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.

```

6014         \edef\@gls@thisval{#1}%
6015         \@for\@gls@map:=\@gls@keymap\do{%

```

```

6016      \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6017      \ifdefequal{\@this@key}{\@gls@thisval}%
6018      {%
6019          \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6020          \@endfortrue
6021      }%
6022      {}%
6023  }%
6024  \ifcsdef{gls\@gls@thisval accessdisplay}%
6025  {\letcs\@glxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
6026  {\let\@glxtr@accessdisplay\@firstoftwo}%
6027  }%
6028  }
6029  }
6030  {%
6031  \newcommand*{\glxtr@setaccessdisplay}[1]{%
6032  \let\@glxtr@accessdisplay\@firstoftwo}
6033  }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

6034 \newrobustcmd*{\glossentrynameother}[2]{%
6035   \@glsdoifexistsorwarn{#1}%
6036   {%

```

Accessibility support:

```

6037   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6038   \glsssetabbrvfmt{\glscategory{#1}}%
6039   \glshasattribute{#1}{glossnamefont}%
6040   {%
6041       \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6042       \ifcsdef{\@glxtr@attrval}%
6043       {%
6044           \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6045       }%
6046       {%
6047           \GlossariesExtraWarning{Unknown control sequence name
6048           ‘\@glxtr@attrval’ supplied in glossnamefont attribute
6049           for entry ‘#1’. Reverting to default \string\glsgnamefont}%
6050           \let\@glxtr@glossnamefont\glsgnamefont
6051       }%
6052   }%
6053   {\let\@glxtr@glossnamefont\glsgnamefont}%
6054   \glssifattribute{#1}{glossname}{firstuc}%
6055   {%
6056       \@glxtr@accessdisplay
6057       {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6058       {#1}%

```

```

6059 }%
6060 {%
6061   \glsifattribute{#1}{glossname}{title}%
6062   {%
6063     \@glsxtr@do@titlecaps@warn
6064     \@glsxtr@accessdisplay
6065     {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
6066     {#1}%
6067   }%
6068   {%
6069     \glsifattribute{#1}{glossname}{uc}%
6070     {%
6071       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6072       \@glsxtr@accessdisplay
6073       {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6074       {#1}%
6075     }%
6076     {%
6077       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6078       \@glsxtr@accessdisplay
6079       {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6080       {#1}%
6081     }%
6082   }%
6083 }%

```

Do post-name hook.

```

6084   \glsxtrpostnamehook{#1}%
6085 }%
6086 }

```

`format@override` Determines if the `format` key should override the indexing attribute value.

```

6087 \newif\if@glsxtr@format@override
6088 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6089 \@ifpackageloaded{hyperref}
6090 {

```

If `hyperref`'s `hyperindex` option is on, then `hyperref` will automatically add `\hyperpage`, so don't add it.

```

6091   \ifHy@hyperindex
6092     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6093       \@glsxtr@format@override true
6094       \appto\theindex{\let\glshypernumber\@firstofone}%
6095     }
6096   \else

```

```

6097 \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6098 \@glstr@format@overridetrue
6099 \appto\theindex{\let\glshypernumber\hyperpage}%
6100 }
6101 \fi
6102 }
6103 {
6104 \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6105 \@glstr@format@overridetrue
6106 }
6107 }
6108 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6109 \newcommand*{\glstrdoautoindexname}[2]{%
6110 \glshasattribute{#1}{#2}%
6111 {%

```

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.

```

6112 \@glstr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

6113 \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6114 \if@glstr@format@override
6115 \ifx\@glstrnumberformat\@glstr@defaultnumberformat
6116 \else
6117 \let\@glstr@attrval\@glstrnumberformat
6118 \fi
6119 \fi
6120 \ifdefstring{\@glstr@attrval}{true}%
6121 {}%
6122 {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6123 \expandafter\glstrautoindex\expandafter{\@glo@name}%
6124 }%
6125 {}%
6126 }

```

glstrautoindex

```

6127 \newcommand*{\glstrautoindex}{\index}

```

xtrautoindexesc

```

6128 \newcommand{\glstrautoindexesc}{%
6129 \@gls@checkmkidxchars\@glo@sort
6130 \@glstr@autoindex@doextra@esc\@glo@sort
6131 }

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6132 \newcommand*{\@glstr@autoindex@setname}[1]{%

```

```

6133 \protected@edef\@glo@name{\glxtrautoindexentry{#1}}%
6134 \glxtrautoindexassignsort{\@glo@sort}{#1}%
6135 \glxtrautoindexesc
6136 \epreto\@glo@name{\@glo@sort\@glxtr@autoindex@at}%
6137 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6138 \newcommand*{\glxtrautoindexentry}[1]{\string\glstryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```

6139 \newcommand*{\glxtrautoindexassignsort}[2]{%
6140 \glslentryfield{#1}{#2}{sort}%
6141 }

```

dex@doextra@esc

```

6142 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%

```

Escape the escape character unless it has already been escaped.

```

6143 \ifx\@glxtr@autoindex@esc\@gl@quotechar
6144 \else
6145 \def\@gl@checkedmkidx{%
6146 \edef\@glxtr@checkspch{%
6147 \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6148 \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6149 \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6150 @@glxtr@checkspch
6151 \let#1\@gl@checkedmkidx\relax
6152 \fi

```

Escape actual character unless it has already been escaped.

```

6153 \ifx\@glxtr@autoindex@at\@gl@actualchar
6154 \else
6155 \def\@gl@checkedmkidx{%
6156 \edef\@glxtr@checkspch{%
6157 \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6158 \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6159 \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6160 @@glxtr@checkspch
6161 \let#1\@gl@checkedmkidx\relax
6162 \fi

```

Escape level character unless it has already been escaped.

```

6163 \ifx\@glxtr@autoindex@level\@gl@levelchar
6164 \else
6165 \def\@gl@checkedmkidx{%
6166 \edef\@glxtr@checkspch{%
6167 \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6168 \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6169 \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6170 @@glxtr@checkspch

```

```

6171 \let#1\@gls@checkedmkidx\relax
6172 \fi

Escape encap character unless it has already been escaped.
6173 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6174 \else
6175 \def\@gls@checkedmkidx{%
6176 \edef\@glsxtr@checkspch{%
6177 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6178 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6179 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6180 \@glsxtr@checkspch
6181 \let#1\@gls@checkedmkidx\relax
6182 \fi
6183 }

```

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`.

```
6184 \newcommand*{\@glsxtr@autoindex@at}{}

```

`trSetActualChar` Set the actual character.

```

6185 \newcommand*{\GlsXtrSetActualChar}[1]{%
6186 \gdef\@glsxtr@autoindex@at{#1}%
6187 \def\@glsxtr@autoindex@escat##1##2##3\@glsxtr@endescspch{%
6188 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6189 }%
6190 }
6191 \@onlypreamble\GlsXtrSetActualChar
6192 \makeatother
6193 \GlsXtrSetActualChar{@}
6194 \makeatletter

```

`autoindex@encap` Encap character for use with `\index`.

```
6195 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`XtrSetEncapChar` Set the encap character.

```

6196 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6197 \gdef\@glsxtr@autoindex@encap{#1}%
6198 \def\@glsxtr@autoindex@escencap##1##2##3\@glsxtr@endescspch{%
6199 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6200 }%
6201 }
6202 \GlsXtrSetEncapChar{|}
6203 \@onlypreamble\GlsXtrSetEncapChar

```

`autoindex@level` Level character for use with `\index`.

```
6204 \newcommand*{\@glsxtr@autoindex@level}{}

```


XtrSetLevelChar Set the encap character.

```
6205 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6206   \gdef\@glstr@autoindex@level{#1}%
6207   \def\@glstr@autoindex@esclevel##1##2##3\@glstr@endescspch{%
6208     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@esclevel}{##1}{##2}{##3}%
6209   }%
6210 }
6211 \GlsXtrSetLevelChar{!}
6212 \@onlypreamble\GlsXtrSetLevelChar
```

r@autoindex@esc Escape character for use with \index.

```
6213 \newcommand*{\@glstr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
6214 \newcommand*{\GlsXtrSetEscChar}[1]{%
6215   \gdef\@glstr@autoindex@esc{#1}%
6216   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
6217     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escquote}{##1}{##2}{##3}%
6218   }%
6219 }
6220 \GlsXtrSetEscChar{"}
6221 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
6222 \ifdef\actualchar
6223 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6224 {}
```

Quote character \quotechar:

```
6225 \ifdef\quotechar
6226 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6227 {}
```

Level character \levelchar:

```
6228 \ifdef\levelchar
6229 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6230 {}
```

Encap character \encapchar:

```
6231 \ifdef\encapchar
6232 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6233 {}
```

leto@endescspch

```
6234 \def\@glstr@gobbleto@endescspch#1\@glstr@endescspch{}
```

toindex@esc@spch @@glstr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}

```

6235 \newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
6236   \gls@tmpb=\expandafter{\gls@checkedmkidx}%
6237   \toks@={#3}%
6238   \ifx\@nnil#3\relax
6239     \def\@@glsxtr@checkspch{\glsxtr@gobbleto@endescspch#5\glsxtr@endescspch}%
6240   \else
6241     \ifx\@nnil#4\relax
6242       \edef\gls@checkedmkidx{\the\gls@tmpb\the\toks@}%
6243       \def\@@glsxtr@checkspch{\glsxtr@gobbleto@endescspch
6244         #4#5\glsxtr@endescspch}%
6245     \else
6246       \edef\gls@checkedmkidx{\the\gls@tmpb\the\toks@
6247         \glsxtr@autoindex@esc#1}%
6248       \def\@@glsxtr@checkspch{#2#5#1\@nnil#1\glsxtr@endescspch}%
6249     \fi
6250   \fi
6251   \@@glsxtr@checkspch
6252 }

```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```

6253 \renewcommand*{\Glossentrydesc}[1]{%
6254   \glsdoifexistsorwarn{#1}%
6255   {%
6256     \glssetabbrvfmt{\glscategory{#1}}%
6257     \Glsaccessdesc{#1}%
6258   }%
6259 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

6260 \renewcommand*{\Glossentrysymbol}[1]{%
6261   \glsdoifexistsorwarn{#1}%
6262   {%
6263     \glssetabbrvfmt{\glscategory{#1}}%
6264     \Glsaccesssymbol{#1}%
6265   }%
6266 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

6267 \renewcommand*{\Glossentrysymbol}[1]{%
6268   \glsdoifexistsorwarn{#1}%
6269   {%
6270     \glssetabbrvfmt{\glscategory{#1}}%
6271     \Glsaccesssymbol{#1}%
6272   }%
6273 }

```

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.

```
6274 \newcommand*{\GlsXtrEnableInitialTagging}{%
6275   \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
6276 }
6277 \@onlypreamble\GlsXtrEnableInitialTagging
```

`r@enabletagging` Starred version undefines command.

```
6278 \newcommand*{\s@glsxtr@enabletagging}[2]{%
6279   \undef#2%
6280   \@glsxtr@enabletagging{#1}{#2}%
6281 }
```

`r@enabletagging` Internal command.

```
6282 \newcommand*{\@glsxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6283   \@for\@glsxtr@cat:=#1\do
6284   {%
6285     \ifdefempty\@glsxtr@cat
6286     {}%
6287     {\glsssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
6288   }%
6289   \newrobustcmd*#2[1]{##1}%
6290   \def\@glsxtr@taggingcs{#2}%
6291   \renewcommand*\@glsxtr@activate@initialtagging{%
6292     \let#2\@glsxtr@tag
6293   }%
6294   \ifundef\@glg@preglossaryhook
6295   {\GlossariesExtraWarning{Initial tagging requires at least
6296     glossaries.sty v4.19 to work correctly}}%
6297   {}%
6298 }
```

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`

```
6299 \ifundef\mfu@checkword@do
6300 {
6301   \newcommand*{\mfu@checkword@do}[1]{%
6302     \ifdefstring{\mfu@checkword@arg}{#1}%
6303     {%
6304       \let\@mfu@domakefirstuc\@firstofone
6305       \listbreak
6306     }%
6307     {}%
6308   }
```

`\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.

```

6309 \ifundef\mfu@checkword
6310 {
6311   \newcommand{\@glxtr@do@titlecaps@warn}{%
6312     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6313       support not available}%
6314     \let\@glxtr@do@titlecaps@warn\relax
6315   }
6316 }
6317 {
6318   \renewcommand*\mfu@checkword[1]{%
6319     \def\mfu@checkword@arg{#1}%
6320     \let\@mfu@do@makefirstuc\makefirstuc
6321     \forlistloop\mfu@checkword@do\@mfu@nocaplist
6322   }
6323 }
6324 }
6325 {}% no patch required

```

`@titlecaps@warn` Do warning if title case not supported.

```

6326 \newcommand*\@glxtr@do@titlecaps@warn{}

```

`@initialtagging` Used in `\printglossary` but at least v4.19 of `glossaries` required.

```

6327 \newcommand*\@glxtr@activate@initialtagging{}

```

`\@glxtr@tag` Definition of tagging command when used in glossary.

```

6328 \newrobustcmd*\@glxtr@tag[1]{%
6329   \gl@ifattribute{\gl@currententrylabel}{tagging}{true}%
6330   {\glxtr@tagfont{#1}}{#1}%
6331 }

```

`\glxtr@tagfont` Used in the glossary.

```

6332 \newcommand*\glxtr@tagfont[1]{\underline{#1}}

```

`preglossaryhook` This macro was introduced in `glossaries` version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```

6333 \ifdef\@gl@preglossaryhook
6334 {
6335   \renewcommand*\@gl@preglossaryhook{%
6336     \@glxtr@activate@initialtagging

```

Since the `glossaries` are automatically scoped, `\@glxtr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

```

6337 \ifundef\@glxtr@org@postdescription
6338 {%
6339 \let\@glxtr@org@postdescription\glspostdescription
6340 \renewcommand*{\glspostdescription}{%
6341 \ifglentryexists{\glscurrententrylabel}%
6342 {%
6343 \glxtrpostdescription
6344 \@glxtr@org@postdescription
6345 }%
6346 }%
6347 }%
6348 }%
6349 {}%

```

Enable the options used by \@glxtrp:

```

6350 \glossxtrsetpopts
6351 }%
6352 }
6353 {}

```

postdescription This command will only be used if \@gls@preglossaryhook is available *and* the glossary style uses \glspostdescription without modifying it. (\nopostdesc will suppress this.) The glossaries-extra-stylemods package will add the post description hook to all the predefined styles that don't include it.

```

6354 \newcommand*{\glxtrpostdescription}{%
6355 \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
6356 }

```

postdescgeneral

```

6357 \newcommand*{\glxtrpostdescgeneral}{}

```

xtrpostdescterm

```

6358 \newcommand*{\glxtrpostdescterm}{}

```

postdescacronym

```

6359 \newcommand*{\glxtrpostdescacronym}{}

```

descabbreviation

```

6360 \newcommand*{\glxtrpostdescabbreviation}{}

```

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.

```

6361 \newcommand*{\glsdefpostdesc}[2]{%
6362 \csdef{glxtrpostdesc#1}{#2}%
6363 }

```

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.

```
6364 \renewcommand*{\glspostlinkhook}{%
6365   \ifglstryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6366 }
```

`xtrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
6367 \newcommand*{\glsxtrpostlinkhook}{%
6368   \glsxtrdiscardperiod{\glslabel}%
6369   {\glsxtrpostlinkendsentence}%
6370   {\glsxtrifcustomdiscardperiod
6371     {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
6372   {\glsxtrpostlink}%
6373 }%
6374 }
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6375 \newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

`\glsxtrpostlink`

```
6376 \newcommand*{\glsxtrpostlink}{%
6377   \csuse{glsxtrpostlink\glscategory{\glslabel}}%
6378 }
```

`\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`.

```
6379 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6380   \ifthenelse{\equal{#1}{}}{%
6381     {\PackageError{glossaries-extra}
6382       {Invalid empty category label in \string\glsdefpostlink}{}}%
6383     {\csdef{glsxtrpostlink#1}{#2}}%
6384 }
```

`linkendsentence` Done by `\glsxtrpostlinkhook` if a full stop is discarded.

```
6385 \newcommand*{\glsxtrpostlinkendsentence}{%
6386   \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
6387   {%
6388     \csuse{glsxtrpostlink\glscategory{\glslabel}}%
  Put the full stop back.
6389   .\spacefactor\sfcode'\. \relax
6390 }%
6391 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
6392 \spacefactor\sfcode'\. \relax
6393 }%
6394 }
```

DescOnFirstUse Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6395 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6396 \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}%
6397 }
```

SymbolOnFirstUse Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6398 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6399 \glxtrifwasfirstuse
6400 {%
6401 \ifglshassymbol{\glslabel}%
6402 {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
6403 }%
6404 }%
6405 {}%
6406 }
```

SymbolDescOnFirstUse 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.

```
6407 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6408 \glxtrifwasfirstuse
6409 {%
6410 \space\glxtrparen
6411 {%
6412 \ifglshassymbol{\glslabel}%
6413 {\glssaccesssymbol{\glslabel}, }%
6414 }%
6415 \glssaccessdesc{\glslabel}%
6416 }%
6417 }%
6418 {}%
6419 }
```

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).

```
6420 \newcommand*{\glxtrdiscardperiod}[3]{%
6421 \glxtrifwasfirstuse
6422 {%
6423 \glssifattribute{#1}{retainfirstuseperiod}{true}%
```

```

6424   {#3}%
6425   {%
6426     \glsifattribute{#1}{discardperiod}{true}%
6427     {%
6428       \glsifplural
6429       {%
6430         \glsifattribute{#1}{pluraldiscardperiod}{true}%
6431         {\glsxtrifperiod{#2}{#3}}%
6432         {#3}%
6433       }%
6434     }%
6435     \glsxtrifperiod{#2}{#3}%
6436   }%
6437 }%
6438 {#3}%
6439 }%
6440 }%
6441 {%
6442   \glsifattribute{#1}{discardperiod}{true}%
6443   {%
6444     \glsifplural
6445     {%
6446       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6447       {\glsxtrifperiod{#2}{#3}}%
6448       {#3}%
6449     }%
6450   }%
6451   \glsxtrifperiod{#2}{#3}%
6452 }%
6453 }%
6454 {#3}%
6455 }%
6456 }

```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

6457 \newcommand*\glsxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```

6458 \newcommand*\glsxtr@punclist{.,;?!}

```

`\punctuationmark` Add character to punctuation list.

```

6459 \newcommand*\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

6460 \newcommand*\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}

```


`\glxstrifpunc` `\glxstrifnextpunc{<true part>}{<false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```
6461 \newcommand*{\glxstrifnextpunc}[2]{%
6462   \def\reserved@a{#1}%
6463   \def\reserved@b{#2}%
6464   \futurelet\@glspunc@token\glxstr@ifnextpunc
6465 }
```

`glxstr@ifnextpunc`

```
6466 \newcommand*{\glxstr@ifnextpunc}{%
6467   \glxstr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6468   \reserved@b
6469 }
```

`glxstr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```
6470 \newcommand*{\glxstr@ifpunctoken}[1]{%
6471   \expandafter\@glxstr@ifpunctoken\expandafter#1\glxstr@punclist\@nnil
6472 }
```

`glxstr@ifpunctoken`

```
6473 \def\@glxstr@ifpunctoken#1#2{%
6474   \let\reserved@d=#2%
6475   \ifx\reserved@d\@nnil
6476     \let\glxstr@next\@glxstr@notfoundinlist
6477   \else
6478     \ifx#1\reserved@d
6479       \let\glxstr@next\@glxstr@foundinlist
6480     \else
6481       \let\glxstr@next\@glxstr@ifpunctoken
6482     \fi
6483   \fi
6484   \glxstr@next#1%
6485 }
```

`glxstr@foundinlist`

```
6486 \def\@glxstr@foundinlist#1\@nnil{\@firstoftwo}
```

`glxstr@notfoundinlist`

```
6487 \def\@glxstr@notfoundinlist#1{\@secondoftwo}
```

`glxstr@dopostpunc`

`\glxstr@dopostpunc{<code>}`

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```
6488 \newcommand{\glxtrdopostpunc}[1]{%
6489   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6490 }
```

@glxtr@swaptwo

```
6491 \newcommand{\@glxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
6492 \define@key{glxtrabbrv}{category}{%
6493   \edef\glscategorylabel{#1}%
6494   \ifcsdef{@glxabbrv@current@#1}%
6495   {%
```

Warning should already have been issued.

```
6496   \let\@glxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6497   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6498   \glxtr@applyabbrvstyle{\cename @glxabbrv@current@#1\endcename}%
6499   \let\GlsXtrWarnDeprecatedAbbrStyle\@glxtr@orgwarndep
6500 }%
6501 {}%
6502 }
```

Save the short plural form. This may be needed before the entry is defined.

```
6503 \define@key{glxtrabbrv}{shortplural}{%
6504   \def\@glx@shortpl{#1}%
6505 }
```

Similarly for the long plural form.

```
6506 \define@key{glxtrabbrv}{longplural}{%
6507   \def\@glx@longpl{#1}%
6508 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
6509 \newtoks\glsshortpltok
```

\glslongpltok

```
6510 \newtoks\glslongpltok
```

`sxtr@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.

```
6511 \newcommand*{\@glxstr@insertdots}[2]{%
6512   \def#1{}%
6513   \@glxstr@insert@dots#1#2\@nnil
6514 }
```

`xtr@insert@dots`

```
6515 \newcommand*{\@glxstr@insert@dots}[2]{%
6516   \ifx\@nnil#2\relax
6517   \let\@glxstr@insert@dots@next\@gobble
6518   \else
6519   \ifx\relax#2\relax
6520   \else
6521     \appto#1{#2.}%
6522   \fi
6523   \let\@glxstr@insert@dots@next\@glxstr@insert@dots
6524   \fi
6525   \@glxstr@insert@dots@next#1%
6526 }
```

Similarly provide a way of replacing spaces with `\glxstrwordsep`, which first needs to be defined:

`\glxstrwordsep`

```
6527 \newcommand*{\glxstrwordsep}{\space}
```

Each word is marked with

`\glxstrword`

```
6528 \newcommand*{\glxstrword}[1]{#1}
```

`tr@markwordseps`

```
6529 \newcommand*{\@glxstr@markwordseps}[2]{%
6530   \def#1{}%
6531   \@glxstr@mark@wordseps#1#2 \@nnil
6532 }
```

`r@mark@wordseps`

```
6533 \def\@glxstr@mark@wordseps#1#2 #3{%
6534   \ifdefempty{#1}%
6535   {\def#1{\protect\glxstrword{#2}}}%
6536   {\appto#1{\protect\glxstrwordsep\protect\glxstrword{#2}}}%
6537   \ifx\@nnil#3\relax
```

```

6538 \let\@glxstr@mark@wordseps@next\relax
6539 \else
6540 \def\@glxstr@mark@wordseps@next{%
6541 \@glxstr@mark@wordseps#1#3}%
6542 \fi
6543 \@glxstr@mark@wordseps@next
6544 }

```

newabbreviation Define a new generic abbreviation.

```

6545 \newcommand*{\newabbreviation}[4][]{%
6546 \glxstr@newabbreviation{#1}{#2}{#3}{#4}%
6547 }

```

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.)

```

6548 \newcommand*{\glxstr@newabbreviation}[4]{%
6549 \glskystok{#1}%
6550 \glslabeltok{#2}%
6551 \glsshorttok{#3}%
6552 \glslongtok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```

6553 \def\glxstrorgshort{#3}%
6554 \def\glxstrorglong{#4}%

```

Provide extra settings for hooks (if modified, this command must end with a comma).

```

6555 \def\ExtraCustomAbbreviationFields{}%

```

Initialise accessibility settings if required.

```

6556 \@gl@initaccesskeys

```

Get the category.

```

6557 \def\glscategorylabel{abbreviation}%
6558 \glxstr@applyabbrvstyle{\@gl@abbrv@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```

6559 \setkeys*{\glxstr@abbrv}[shortplural,longplural]{#1}%

```

Set the default long plural

```

6560 \def\@gl@longpl{#4\glspluralsuffix}%
6561 \let\@gl@default@longpl\@gl@longpl

```

Has the markwords attribute been set?

```

6562 \gl@ifcategoryattribute{\glscategorylabel}{markwords}{true}%
6563 {%
6564 \@glxstr@markwordseps\@gl@long{#4}%
6565 \expandafter\def\expandafter\@gl@longpl\expandafter
6566 {\@gl@long\glspluralsuffix}%
6567 \let\@gl@default@longpl\@gl@longpl

```

Update \glslongtok.

```

6568 \expandafter\glslongtok\expandafter{\@gls@long}%
6569 }%
6570 {}%

```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```

6571 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6572 {%
6573 \@glsxtr@markwordseps\@gls@short{#3}%
6574 }%
6575 {}%

```

Has the insertdots attribute been set?

```

6576 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6577 {%
6578 \@glsxtr@insertdots\@gls@short{#3}%
6579 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6580 }%
6581 {\def\@gls@short{#3}}%
6582 }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

6583 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6584 {%
6585 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6586 '\abbrvpluralsuffix}%
6587 }%
6588 {}%

```

Has the noshortplural attribute been set?

```

6589 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6590 {%
6591 \let\@gls@shortpl\@gls@short
6592 }%
6593 {%
6594 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6595 \abbrvpluralsuffix}%
6596 }%
6597 }%

```

Update \glsshorttok:

```

6598 \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

6599 \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%

```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```

6600 \setkeys*{glsxtrabbrv}[category]{#1}%

```

Has the plural been explicitly set?

```

6601 \ifx\@gls@default@longpl\@gls@longpl
6602 \else

```

Has the markwords attribute been set?

```
6603 \glscategoryattribute{\glscategorylabel}{markwords}{true}%
6604 {%
6605   \expandafter\@glstr@markwordseps\expandafter\@glslongpl\expandafter
6606     {\@glslongpl}%
6607   }%
6608   }%
6609 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
6610 \expandafter\glsshortpltok\expandafter{\@glsshortpl}%
6611 \expandafter\glslongpltok\expandafter{\@glslongpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6612 \@glsl@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6613 \newabbreviationhook
```

Define this entry:

```
6614 \protected@edef\@do@newglossaryentry{%
6615   \noexpand\newglossaryentry{\the\glslabeltok}%
6616   {%
6617     type=\glstrabbrvtype,%
6618     category=abbreviation,%
6619     short={\the\glsshorttok},%
6620     shortplural={\the\glsshortpltok},%
6621     long={\the\glslongtok},%
6622     longplural={\the\glslongpltok},%
6623     name={\the\glsshorttok},%
6624     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6625 \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6626 \the\glscopylisttok
6627   }%
6628 }%
6629 \@do@newglossaryentry
6630 \GlsXtrPostNewAbbreviation
6631 }
```

`\evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```
6632 \newcommand*{\glstrnewabbrevpresetkeyhook}[3]{}
```

`\NewAbbreviation` Hook used by abbreviation styles.

```
6633 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

`\bbreviationhook` Hook for use with `\newabbreviation`.

```
6634 \newcommand*{\newabbreviationhook}{}%
```

reviousFields

```
6635 \newcommand*{\CustomAbbreviationFields}{}
```

`\glstrparen` For the parenthetical styles.

```
6636 \newcommand*{\glstrparen}[1]{(#1)}
```

`lstrfullformat` Full format without case change.

```
6637 \newcommand*{\glstrfullformat}[2]{%
6638   \glstfirstlongfont{\glstaccesslong{#1}}#2\glstrfullsep{#1}%
6639   \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshort{#1}}}%
6640 }
```

`lstrfullformat` Full format with case change.

```
6641 \newcommand*{\Glsstrfullformat}[2]{%
6642   \glstfirstlongfont{\Glstaccesslong{#1}}#2\glstrfullsep{#1}%
6643   \glstrparen{\protect\glstfirstabbrvfont{\Glstaccessshort{#1}}}%
6644 }
```

`xstrfullplformat` Plural full format without case change.

```
6645 \newcommand*{\glstrfullplformat}[2]{%
6646   \glstfirstlongfont{\glstaccesslongpl{#1}}#2\glstrfullsep{#1}%
6647   \glstrparen{\protect\glstfirstabbrvfont{\glstaccessshortpl{#1}}}%
6648 }
```

`xstrfullplformat` Plural full format with case change.

```
6649 \newcommand*{\Glsstrfullplformat}[2]{%
6650   \glstfirstlongfont{\Glstaccesslongpl{#1}}#2\glstrfullsep{#1}%
6651   \glstrparen{\protect\glstfirstabbrvfont{\Glstaccessshortpl{#1}}}%
6652 }
```

`\glstrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```
6653 \newcommand*{\glstrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glstentryfull` (for example, first use suppresses the long form or uses a footnote).

`inlinefullformat` Full format without case change.

```
6654 \newcommand*{\glstrinlinefullformat}{\glstrfullformat}
```

`inlinefullformat` Full format with case change.

```
6655 \newcommand*{\Glsstrinlinefullformat}{\Glsstrfullformat}
```

`xstrfullplformat` Plural full format without case change.

```
6656 \newcommand*{\glstrinlinefullplformat}{\glstrfullplformat}
```

`inefullplformat` Plural full format with case change.

```
6657 \newcommand*{\Glsstrinlinefullplformat}{\Glsstrfullplformat}
```

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
6658 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}

\Glsentryfull
6659 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}

\glsentryfullpl
6660 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}

\Glsentryfullpl
6661 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}

\firstabbrvfont  Font changing command used for the abbreviation on first use or in the full format.
6662 \newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}

\abbrvdefaultfont  Font changing command used for the abbreviation on first use or in the full format.
6663 \newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}

\glsabbrvfont  Font changing command used for the abbreviation on subsequent use.
6664 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}

\abbrvdefaultfont
6665 \newcommand*{\glsabbrvdefaultfont}[1]{#1}

\glslongfont  Font changing command used for the long form in commands like \glsxtrlong.
6666 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}

\longdefaultfont  Default font changing command used for the long form in commands like \glsxtrlong.
6667 \newcommand*{\glslongdefaultfont}[1]{#1}

\lsfirstlongfont  Font changing command used for the long form on first use or in the full format.
6668 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}

\longdefaultfont
6669 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}

\brvpluralsuffix  Default plural suffix. Allow an alternative default suffix for abbreviations.
6670 \newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}

\brvpluralsuffix  Default plural suffix.
6671 \newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}

```


`\glxtrfull` Full form (no case-change).

```
6672 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\ns@glxtrfull}
6673 \newcommand*\ns@glxtrfull[2][{}]{%
6674   \new@ifnextchar[{\@glxtr@full{#1}{#2}}]{%
6675     {\@glxtr@full{#1}{#2}[]}%
6676 }
```

`\@glxtr@full` Low-level macro:

```
6677 \def\@glxtr@full#1#2[#3]{%
```

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

```
6678   \@glxtr@record{#1}{#2}{glslink}%
6679   \glstoifexists{#2}%
6680   {%
6681     \glsssetabbrvfmt{\glscategory{#2}}%
6682     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6683     \let\gl@ifplural\@secondoftwo
6684     \let\gl@scapscase\@firstofthree
6685     \let\gl@insert\@empty
6686     \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.

```
6687     \glxtrsetupfulldefs
6688     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6689   }%
6690   \glspostlinkhook
6691 }
```

`trsetupfulldefs`

```
6692 \newcommand*{\glxtrsetupfulldefs}{%
6693   \let\glxtrifwasfirstuse\@firstoftwo
6694 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
6695 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
6696 \newcommand*\ns@Glsxtrfull[2][{}]{%
6697   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}]{%
6698     {\@Glsxtr@full{#1}{#2}[]}%
6699 }
```

`\@Glsxtr@full` Low-level macro:

```
6700 \def\@Glsxtr@full#1#2[#3]{%
6701   \glstoifexists{#2}%
6702   {%
6703     \glsssetabbrvfmt{\glscategory{#2}}%
```

```

6704 \let\do@glS@link@checkfirsthyper\@glS@link@nocheckfirsthyper
6705 \let\glSifplural\@secondoftwo
6706 \let\glScapscase\@secondofthree
6707 \let\glSinsert\@empty
6708 \def\glScustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6709 \glSxtrsetupfulldefs
6710 \@glS@link[#1]{#2}{\csname glS@\glstype @entryfmt\endcsname}%
6711 }%
6712 \glSpostlinkhook
6713 }

```

\GLSxtrfull Full form (all uppercase).

```

6714 \newrobustcmd*{\GLSxtrfull}{\@glS@hyp@opt\ns@GLSxtrfull}
6715 \newcommand*\ns@GLSxtrfull[2][]{%
6716 \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
6717 {\@GLSxtr@full{#1}{#2}[]}%
6718 }

```

\@GLSxtr@full Low-level macro:

```

6719 \def\@GLSxtr@full#1#2[#3]{%
6720 \glSdoifexists{#2}%
6721 {%
6722 \glSsetabbrvfmt{\glScategory{#2}}%
6723 \let\do@glS@link@checkfirsthyper\@glS@link@nocheckfirsthyper
6724 \let\glSifplural\@secondoftwo
6725 \let\glScapscase\@thirdofthree
6726 \let\glSinsert\@empty
6727 \def\glScustomtext{\mfirstucMakeUppercase{\glSxtrinlinefullformat{#2}{#3}}}%
6728 \glSxtrsetupfulldefs
6729 \@glS@link[#1]{#2}{\csname glS@\glstype @entryfmt\endcsname}%
6730 }%
6731 \glSpostlinkhook
6732 }

```

\glSxtrfullpl Plural full form (no case-change).

```

6733 \newrobustcmd*{\glSxtrfullpl}{\@glS@hyp@opt\ns@glSxtrfullpl}
6734 \newcommand*\ns@glSxtrfullpl[2][]{%
6735 \new@ifnextchar[{\@glSxtr@fullpl{#1}{#2}}%
6736 {\@glSxtr@fullpl{#1}{#2}[]}%
6737 }

```

\@glSxtr@fullpl Low-level macro:

```

6738 \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).
6739 \@glSxtr@record{#1}{#2}{glSlink}%
6740 \glSdoifexists{#2}%
6741 {%

```

```

6742 \glsssetabbrvfmt{\glscategory{#2}}%
6743 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6744 \let\gl@sifplural\@firstoftwo
6745 \let\gl@scapscase\@secondoftwo
6746 \let\gl@insert\@empty
6747 \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6748 \glsxtrsetupfulldefs
6749 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6750 }%
6751 \glspostlinkhook
6752 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

6753 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\ns@Glsxtrfullpl}
6754 \newcommand*\ns@Glsxtrfullpl[2][]{%
6755 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6756 {\@Glsxtr@fullpl{#1}{#2}[]}%
6757 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6758 \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).
6759 \@gl@xtr@record{#1}{#2}{gl@link}%
6760 \gl@doifexists{#2}%
6761 {%
6762 \glsssetabbrvfmt{\glscategory{#2}}%
6763 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6764 \let\gl@sifplural\@firstoftwo
6765 \let\gl@scapscase\@secondoftwo
6766 \let\gl@insert\@empty
6767 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
6768 \glsxtrsetupfulldefs
6769 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6770 }%
6771 \glspostlinkhook
6772 }

```

`\GLSxtrfullpl` Plural full form (all upper case).

```

6773 \newrobustcmd*{\GLSxtrfullpl}{\@gl@hyp@opt\ns@GLSxtrfullpl}
6774 \newcommand*\ns@GLSxtrfullpl[2][]{%
6775 \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
6776 {\@GLSxtr@fullpl{#1}{#2}[]}%
6777 }

```

`\@GLSxtr@fullpl` Low-level macro:

```

6778 \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).

```

6779 \@glsxtr@record{#1}{#2}{glslink}%
6780 \glsdoifexists{#2}%
6781 {%
6782 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6783 \let\glsl@link@firstoftwo
6784 \let\glsl@link@thirdofthree
6785 \let\glsl@link@empty
6786 \def\glscustomtext{%
6787 \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
6788 \glsxtrsetupfulldefs
6789 \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6790 }%
6791 \glspostlinkhook
6792 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

6793 \newrobustcmd*{\glsxtrshort}{\@glsl@hyp@opt\@ns@glsl@xtrshort}

```

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

```

6794 \newcommand*{\ns@glsl@xtrshort}[2] [] {%
6795 \new@ifnextchar[\@glsl@xtrshort{#1}{#2}]{\@glsl@xtrshort{#1}{#2} []}%
6796 }

```

Read in the final optional argument:

```

6797 \def\@glsl@xtrshort#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).

```

6798 \@glsxtr@record{#1}{#2}{glslink}%
6799 \glsdoifexists{#2}%
6800 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

6801 \glsssetabbrvfmt{\glscategory{#2}}%
6802 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6803 \let\glsl@link@firstoftwo
6804 \let\glsl@link@secondoftwo
6805 \let\glsl@link@thirdofthree
6806 \let\glsl@link@empty
6807 \def\glscustomtext{%
6808 \glsabbrvfont{\glsaccessshort{#2}\ifglsl@xtrinsertinside#3\fi}%
6809 \ifglsl@xtrinsertinside\else#3\fi
6810 }%
6811 \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6812 }%
6813 \glspostlinkhook
6814 }

```



```

6849 \let\glscapscase\@thirdofthree
6850 \let\glsinsert\@empty
6851 \def\glscustomtext{%
6852     \mfirstucMakeUppercase
6853     {\glsabbrvfont{\glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6854     \ifglxtrinsertinside\else#3\fi
6855 }%
6856 }%
6857 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6858 }%
6859 \glspostlinkhook
6860 }

```

\glxtrlong

```

6861 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6862 \newcommand*{\ns@glxtrlong}[2] []{%
6863     \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
6864 }

    Read in the final optional argument:
6865 \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).
6866 \@glxtr@record{#1}{#2}{glslink}%
6867 \glsdoifexists{#2}%
6868 {%
6869     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6870     \let\glxtrifwasfirstuse\@secondoftwo
6871     \let\glsifplural\@secondoftwo
6872     \let\glscapscase\@firstofthree
6873     \let\glsinsert\@empty
6874     \def\glscustomtext{%
6875         \glslongfont{\glsaccesslong{#2}\ifglxtrinsertinside#3\fi}%
6876         \ifglxtrinsertinside\else#3\fi
6877     }%
6878     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6879 }%
6880 \glspostlinkhook
6881 }

```

\Glsxtrlong

```

6882 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6883 \newcommand*{\ns@Glsxtrlong}[2] []{%
6884     \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6885 }

```

Read in the final optional argument:

```
6886 \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).

```
6887 \@glstr@record{#1}{#2}{glslink}%
6888 \glsoifexists{#2}%
6889 {%
6890 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6891 \let\glstrifwasfirstuse\@secondoftwo
6892 \let\gl@ifplural\@secondoftwo
6893 \let\glscapscase\@secondofthree
6894 \let\glinsert\@empty
6895 \def\glscustomtext{%
6896 \glslongfont{\@gl@accesslong{#2}\ifglstrinsertinside#3\fi}%
6897 \ifglstrinsertinside\else#3\fi
6898 }%
6899 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6900 }%
6901 \glspostlinkhook
6902 }
```

\GLSxtrlong

```
6903 \newrobustcmd*{\GLSxtrlong}{\@gl@hyp@opt\@ns@GLSxtrlong}
```

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

```
6904 \newcommand*{\ns@GLSxtrlong}[2][]{%
6905 \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
6906 }
```

Read in the final optional argument:

```
6907 \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).

```
6908 \@glstr@record{#1}{#2}{glslink}%
6909 \glsoifexists{#2}%
6910 {%
6911 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6912 \let\glstrifwasfirstuse\@secondoftwo
6913 \let\gl@ifplural\@secondoftwo
6914 \let\glscapscase\@thirdofthree
6915 \let\glinsert\@empty
6916 \def\glscustomtext{%
6917 \mfirstucMakeUppercase
6918 {\glslongfont{\@gl@accesslong{#2}\ifglstrinsertinside#3\fi}%
6919 \ifglstrinsertinside\else#3\fi
6920 }%
6921 }%
6922 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

6923 }%
6924 \glspostlinkhook
6925 }

```

Plural short forms:

\glxstrshortpl

```

6926 \newrobustcmd*{\glxstrshortpl}{\@gls@hyp@opt\@ns@glxstrshortpl}

```

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

```

6927 \newcommand*{\ns@glxstrshortpl}[2] [] {%
6928   \new@ifnextchar[{\@glxstrshortpl{#1}{#2}}{\@glxstrshortpl{#1}{#2} []}%
6929 }

```

Read in the final optional argument:

```

6930 \def\@glxstrshortpl#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).

```

6931   \@glxstr@record{#1}{#2}{glslink}%
6932   \glsdoidexists{#2}%
6933   {%
6934     \glsetabbrvfmt{\glscategory{#2}}%
6935     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6936     \let\glxstrifwasfirstuse\@secondoftwo
6937     \let\gl@ifplural\@firstoftwo
6938     \let\glscapscase\@firstofthree
6939     \let\glinsert\@empty
6940     \def\glscustomtext{%
6941       \glabbrvfont{\glaccessshortpl{#2}\ifglxstrinsertinside#3\fi}%
6942       \ifglxstrinsertinside\else#3\fi
6943     }%
6944     \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6945   }%
6946   \glspostlinkhook
6947 }

```

\Glsxtrshortpl

```

6948 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

6949 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
6950   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
6951 }

```

Read in the final optional argument:

```

6952 \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).

```

6953   \@glxstr@record{#1}{#2}{glslink}%

```



```

6954 \glsdoifexists{#2}%
6955 {%
6956   \glssetabbrvfmt{\glscategory{#2}}%
6957   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6958   \let\glxtrifwasfirstuse\@secondoftwo
6959   \let\glsifplural\@firstoftwo
6960   \let\glscapscase\@secondofthree
6961   \let\glsinsert\@empty
6962   \def\glscustomtext{%
6963     \glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6964     \ifglxtrininsertinside\else#3\fi
6965   }%
6966   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6967 }%
6968 \glspostlinkhook
6969 }

```

\GLSxtrshortpl

```

6970 \newrobustcmd*{\GLSxtrshortpl}{\@gl@hyp@opt\@ns@GLSxtrshortpl}

```

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

```

6971 \newcommand*{\ns@GLSxtrshortpl}[2][{}]{%
6972   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
6973 }

```

Read in the final optional argument:

```

6974 \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).

```

6975   \@glxtr@record{#1}{#2}{glslink}%
6976   \glsdoifexists{#2}%
6977   {%
6978     \glssetabbrvfmt{\glscategory{#2}}%
6979     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6980     \let\glxtrifwasfirstuse\@secondoftwo
6981     \let\glsifplural\@firstoftwo
6982     \let\glscapscase\@thirdofthree
6983     \let\glsinsert\@empty
6984     \def\glscustomtext{%
6985       \mfirstucMakeUppercase
6986       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6987       \ifglxtrininsertinside\else#3\fi
6988     }%
6989     }%
6990     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6991   }%
6992   \glspostlinkhook
6993 }

```

Plural long forms:

`\glstrlongpl`

```
6994 \newrobustcmd*{\glstrlongpl}{\@gls@hyp@opt\@ns@glstrlongpl}
```

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

```
6995 \newcommand*{\ns@glstrlongpl}[2] [] {%
```

```
6996   \new@ifnextchar[{\@glstrlongpl{#1}{#2}}{\@glstrlongpl{#1}{#2} []}%
```

```
6997 }
```

Read in the final optional argument:

```
6998 \def\@glstrlongpl#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).

```
6999   \@glstr@record{#1}{#2}{glslink}%
```

```
7000   \glsoifexists{#2}%
```

```
7001   {%
```

```
7002     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
```

```
7003     \let\glstrifwasfirstuse\@secondoftwo
```

```
7004     \let\gl@ifplural\@firstoftwo
```

```
7005     \let\glscapscase\@firstofthree
```

```
7006     \let\glinsert\@empty
```

```
7007     \def\glscustomtext{%
```

```
7008       \glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
```

```
7009       \ifglstrinsertinside\else#3\fi
```

```
7010     }%
```

```
7011     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
7012   }%
```

```
7013   \glspostlinkhook
```

```
7014 }
```

`\Glsxtrlongpl`

```
7015 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
```

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

```
7016 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
```

```
7017   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
```

```
7018 }
```

Read in the final optional argument:

```
7019 \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).

```
7020   \@glstr@record{#1}{#2}{glslink}%
```

```
7021   \glsoifexists{#2}%
```

```
7022   {%
```

```
7023     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
```

```
7024     \let\glstrifwasfirstuse\@secondoftwo
```

```
7025     \let\gl@ifplural\@firstoftwo
```

```
7026     \let\glscapscase\@secondofthree
```

```
7027     \let\glinsert\@empty
```

```

7028 \def\glscustomtext{%
7029 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
7030 \ifglxtrinsertinside\else#3\fi
7031 }%
7032 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7033 }%
7034 \glspostlinkhook
7035 }

```

\GLSxtrlongpl

```

7036 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7037 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7038 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7039 }

```

Read in the final optional argument:

```

7040 \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).
7041 \@glsxtr@record{#1}{#2}{\glslink}%
7042 \glsdoifexists{#2}%
7043 {%
7044 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7045 \let\glsxtrifwasfirstuse\@secondoftwo
7046 \let\glsifplural\@firstoftwo
7047 \let\gls@scaps\@thirdofthree
7048 \let\glsinsert\@empty
7049 \def\glscustomtext{%
7050 \mfirstucMakeUppercase
7051 {\glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
7052 \ifglxtrinsertinside\else#3\fi
7053 }%
7054 }%
7055 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7056 }%
7057 \glspostlinkhook
7058 }

```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

7059 \newcommand*{\glssetabbrvfmt}[1]{%
7060 \ifcsdef{@glsabbrv@current@#1}%
7061 {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7062 {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7063 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7064 \newrobustcmd*{\glsuseabbrvfont}[2]{\@glssetabbrvfmt{#2}\glsabbrvfont{#1}}

```

`\glsuselongfont` Provide a way to use the long font for a given category for arbitrary text.

```

7065 \newrobustcmd*{\glsuselongfont}[2]{\glssetabbrvfmt{#2}\glslongfont{#1}}

```

`sxtrgenabbrvfmt` Similar to `\glsngenacfmt`, but for abbreviations.

```

7066 \newcommand*{\glsxtrgenabbrvfmt}{%
7067   \ifdefempty\glscustomtext
7068   {%
7069     \ifglsused\glslabel
7070     {%

```

Subsequent use:

```

7071     \glsifplural
7072     {%

```

Subsequent plural form:

```

7073     \glscapscase
7074     {%

```

Subsequent plural form, don't adjust case:

```

7075     \glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7076     }%
7077     {%

```

Subsequent plural form, make first letter upper case:

```

7078     \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7079     }%
7080     {%

```

Subsequent plural form, all caps:

```

7081     \mfirstucMakeUppercase
7082     {\glsxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7083     }%
7084     }%
7085     {%

```

Subsequent singular form

```

7086     \glscapscase
7087     {%

```

Subsequent singular form, don't adjust case:

```

7088     \glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7089     }%
7090     {%

```

Subsequent singular form, make first letter upper case:

```

7091     \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7092     }%
7093     {%

```

Subsequent singular form, all caps:

```

7094     \mfirstucMakeUppercase
7095     {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%

```

7096 }%
 7097 }%
 7098 }%
 7099 {%

First use:

7100 \glsifplural
 7101 {%

First use plural form:

7102 \glscapscase
 7103 {%

First use plural form, don't adjust case:

7104 \glstrfullplformat{\glslabel}{\glsinsert}%
 7105 }%
 7106 {%

First use plural form, make first letter upper case:

7107 \Glsxtrfullplformat{\glslabel}{\glsinsert}%
 7108 }%
 7109 {%

First use plural form, all caps:

7110 \mfirstucMakeUppercase
 7111 {\glstrfullplformat{\glslabel}{\glsinsert}}}%
 7112 }%
 7113 }%
 7114 {%

First use singular form

7115 \glscapscase
 7116 {%

First use singular form, don't adjust case:

7117 \glstrfullformat{\glslabel}{\glsinsert}%
 7118 }%
 7119 {%

First use singular form, make first letter upper case:

7120 \Glsxtrfullformat{\glslabel}{\glsinsert}%
 7121 }%
 7122 {%

First use singular form, all caps:

7123 \mfirstucMakeUppercase
 7124 {\glstrfullformat{\glslabel}{\glsinsert}}}%
 7125 }%
 7126 }%
 7127 }%
 7128 }%
 7129 {%

User supplied text.

```
7130 \glscustomtext
7131 }%
7132 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7133 \newcommand*{\glxtrsubsequentfmt}[2]{%
7134 \glabbrvfont{\glaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7135 \ifglxtrinsertinside \else#2\fi
7136 }
7137 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
7138 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7139 \glabbrvfont{\glaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7140 \ifglxtrinsertinside \else#2\fi
7141 }
7142 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt
```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```
7143 \newcommand*{\Glxtrsubsequentfmt}[2]{%
7144 \glabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7145 \ifglxtrinsertinside \else#2\fi
7146 }
7147 \let\Glxtrdefaultsubsequentfmt\Glxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```
7148 \newcommand*{\Glxtrsubsequentplfmt}[2]{%
7149 \glabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7150 \ifglxtrinsertinside \else#2\fi
7151 }
7152 \let\Glxtrdefaultsubsequentplfmt\Glxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

breiviationstyle

```
7153 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7154 \ifcsundef{@glabbrv@dispstyle@setup@#2}
7155 {%
7156 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{%
7157 }%
7158 }%
```

Have abbreviations already been defined for this category?

```
7159 \ifcsstring{@glabbrv@current@#1}{#2}%
7160 {%
```

Style already set.

```
7161 }%
7162 {%
7163 \def\@glxtr@dostylewarn{%
7164 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7165 {%
7166 \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7167 style has been switched \MessageBreak
7168 for category ‘#1’, \MessageBreak
7169 but there have already been entries \MessageBreak
7170 defined for this category. Unwanted \MessageBreak
7171 side-effects may result}}%
7172 \@endfortrue
7173 }%
7174 \@glxtr@dostylewarn
```

Set up the style for the given category.

```
7175 \csdef{@glsabbrv@current@#1}{#2}%
7176 \glxtr@applyabbrvstyle{#2}%
7177 }%
7178 }%
7179 }
```

applyabbrvstyle Apply the abbreviation style without existence check.

```
7180 \newcommand*{\glxtr@applyabbrvstyle}[1]{%
7181 \csuse{@glsabbrv@dispstyle@setup@#1}%
7182 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7183 }
```

r@applyabbrvfmt Only apply the style formats.

```
7184 \newcommand*{\glxtr@applyabbrvfmt}[1]{%
7185 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7186 }
```

breivationstyle 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.

```
7187 \newcommand*{\newabbreviationstyle}[3]{%
7188 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7189 {%
7190 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7191 defined}{}%
7192 }%
7193 {%
7194 \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```
7195 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7196 #2}%
7197 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
7198 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7199 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7200 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7201 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

Reset \glxtrsubsequentfmt etc in case a style changes this.

```
7202 \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
7203 \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
7204 \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7205 \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7206 #3}%
7207 }%
7208 }
```

breivationstyle

```
7209 \newcommand*{\renewabbreviationstyle}[3]{%
7210 \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7211 {%
7212 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7213 }%
7214 {%
7215 \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
7216 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7217 #2}%
7218 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
7219 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7220 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7221 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7222 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7223 #3}%
7224 }%
7225 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
7226 \newcommand*{\letabbreviationstyle}[2]{%
7227 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7228 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7229 }
```

ecated@abbrstyle

```
\@glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```


Define a synonym for a deprecated abbreviation style.

```

7230 \newcommand*{\@GlsXtrDeprecatedAbbrStyle}[2]{%
7231   \csdef{@Glsabbrv@dispstyle@setup@#1}{%
7232     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7233     \csuse{@Glsabbrv@dispstyle@setup@#2}%
7234   }%
7235   \csletcs{@Glsabbrv@dispstyle@fmts@#1}{@Glsabbrv@dispstyle@fmts@#2}%
7236 }
```

`\GlsXtrWarnDeprecatedAbbrStyle` Generate warning for deprecated style use.

```

7237 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7238   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7239     use ‘#2’ instead}%
7240 }
```

`\GlsXtrUseAbbrStyleSetup`

```

7241 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7242   \ifcsundef{@Glsabbrv@dispstyle@setup@#1}%
7243   {%
7244     \PackageError{glossaries-extra}%
7245       {Unknown abbreviation style definitions ‘#1’}{}%
7246   }%
7247   {%
7248     \csname @Glsabbrv@dispstyle@setup@#1\endcsname
7249   }%
7250 }
```

`\GlsXtrUseAbbrStyleFmts`

```

7251 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7252   \ifcsundef{@Glsabbrv@dispstyle@fmts@#1}%
7253   {%
7254     \PackageError{glossaries-extra}%
7255       {Unknown abbreviation style formats ‘#1’}{}%
7256   }%
7257   {%
7258     \csname @Glsabbrv@dispstyle@fmts@#1\endcsname
7259   }%
7260 }
```

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.

```
7261 \newif\ifglxtrinsertinside
7262 \glxtrinsertinsidefalse
```

trlongshortname

```
7263 \newcommand*{\glxtrlongshortname}{%
7264   \protect\glxabbrvfont{\the\glsshorttok}%
7265 }
```

long-short

```
7266 \newabbreviationstyle{long-short}%
7267 {%
7268   \renewcommand*{\CustomAbbreviationFields}{%
7269     name={\glxtrlongshortname},
7270     sort={\the\glsshorttok},
7271     first={\protect\glxfirstlongfont{\the\glslongtok}%
7272       \protect\glxtrfullsep{\the\glslabeltok}%
7273       \glxtrparen{\protect\glxfirstabbrvfont{\the\glsshorttok}}},%
7274     firstplural={\protect\glxfirstlongfont{\the\glslongpltok}%
7275       \protect\glxtrfullsep{\the\glslabeltok}%
7276       \glxtrparen{\protect\glxfirstabbrvfont{\the\glsshortpltok}}},%
7277     plural={\protect\glxabbrvfont{\the\glsshortpltok}}},%
7278     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
7279 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7280   \glshasattribute{\the\glslabeltok}{regular}%
7281   {%
7282     \glissetattribute{\the\glslabeltok}{regular}{false}%
7283   }%
7284   {%
7285   }%
7286 }%
7287 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7288 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7289 \renewcommand*{\glxabbrvfont}[1]{\glxabbrvdefaultfont{##1}}%
7290 \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvdefaultfont{##1}}%
7291 \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlongdefaultfont{##1}}%
7292 \renewcommand*{\glxlongfont}[1]{\glxlongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7293 \renewcommand*{\glxtrfullformat}[2]{%
7294   \glxfirstlongfont{\glxaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7295   \ifglxtrinsertinside\else##2\fi
7296   \glxtrfullsep{##1}}%
```

```

7297 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7298 }%
7299 \renewcommand*{\glsxtrfullplformat}[2]{%
7300 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7301 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7302 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7303 }%
7304 \renewcommand*{\Glsxtrfullformat}[2]{%
7305 \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7306 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7307 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7308 }%
7309 \renewcommand*{\Glsxtrfullplformat}[2]{%
7310 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7311 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7312 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7313 }%
7314 }

```

Set this as the default style for general abbreviations:

```

7315 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7316 \newcommand*{\glsxtrlongshortdescsort}{%
7317 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7318 }

```

ngshortdescname

```

7319 \newcommand*{\glsxtrlongshortdescname}{%
7320 \protect\glslongfont{\the\glslongtok}
7321 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}}%
7322 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7323 \newabbreviationstyle{long-short-desc}%
7324 {%
7325 \renewcommand*{\CustomAbbreviationFields}{%
7326 name={\glsxtrlongshortdescname},
7327 sort={\glsxtrlongshortdescsort},%
7328 first={\protect\glsfirstlongfont{\the\glslongtok}%
7329 \protect\glsxtrfullsep{\the\glslabeltok}%
7330 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7331 firstplural={\protect\glsfirstlongfont{\the\glslongtok}%
7332 \protect\glsxtrfullsep{\the\glslabeltok}%
7333 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7334 text={\protect\glsabbrvfont{\the\glsshorttok}},%

```

```

7335     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7336 }%

```

Unset the regular attribute if it has been set.

```

7337 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7338   \glshasattribute{\the\glslabeltok}{regular}%
7339   {%
7340     \glissetattribute{\the\glslabeltok}{regular}{false}%
7341     }%
7342   {}%
7343 }%
7344 }%
7345 {%
7346   \GlsXtrUseAbbrStyleFmts{long-short}%
7347 }

```

trshortlongname

```

7348 \newcommand*{\glxtrshortlongname}{%
7349   \protect\glsabbrvfont{\the\glsshorttok}%
7350 }

```

short-long Short form followed by long form in parenthesis on first use.

```

7351 \newabbreviationstyle{short-long}%
7352 {%
7353   \renewcommand*{\CustomAbbreviationFields}{%
7354     name={\glxtrshortlongname},
7355     sort={\the\glsshorttok},
7356     description={\the\glslongtok},%
7357     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7358       \protect\glxtrfullsep{\the\glslabeltok}%
7359       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7360     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7361       \protect\glxtrfullsep{\the\glslabeltok}%
7362       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7363     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7364 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7365   \glshasattribute{\the\glslabeltok}{regular}%
7366   {%
7367     \glissetattribute{\the\glslabeltok}{regular}{false}%
7368     }%
7369   {}%
7370 }%
7371 }%
7372 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7373 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

```

```

7374 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7375 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7376 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7377 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7378 \renewcommand*\glsxtrfullformat[2]{%
7379   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7380   \ifglsxtrininsertinside\else##2\fi
7381   \glsxtrfullsep{##1}%
7382   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7383 }%
7384 \renewcommand*\glsxtrfullplformat[2]{%
7385   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7386   \ifglsxtrininsertinside\else##2\fi
7387   \glsxtrfullsep{##1}%
7388   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7389 }%
7390 \renewcommand*\Glsxtrfullformat[2]{%
7391   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7392   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7393   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7394 }%
7395 \renewcommand*\Glsxtrfullplformat[2]{%
7396   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7397   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7398   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7399 }%
7400 }

```

ortlongdescsort

```

7401 \newcommand*\glsxtrshortlongdescsort{\the\glsshorttok}

```

ortlongdescname

```

7402 \newcommand*\glsxtrshortlongdescname{%
7403   \protect\glsabbrvfont{\the\glsshorttok}
7404   \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
7405 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7406 \newabbreviationstyle{short-long-desc}%
7407 {%
7408   \renewcommand*\CustomAbbreviationFields{%
7409     name={\glsxtrshortlongdescname},
7410     sort={\glsxtrshortlongdescsort},
7411     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7412       \protect\glsxtrfullsep{\the\glslabeltok}%
7413       \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7414     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%

```

```

7415 \protect\glstrfullsep{\the\glslabeltok}%
7416 \glstrparen{\protect\glslfirstlongfont{\the\glslongpltok}}},%
7417 text={\protect\glssabrvfont{\the\glsshorttok}},%
7418 plural={\protect\glssabrvfont{\the\glsshortpltok}}}%
7419 }%

Unset the regular attribute if it has been set.
7420 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7421 \glshasattribute{\the\glslabeltok}{regular}%
7422 {%
7423 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7424 }%
7425 {}%
7426 }%
7427 }%
7428 {%
7429 \GlsXtrUseAbbrStyleFmts{short-long}%
7430 }

```

`\glslongfootnotefont` Only used by the “footnote” styles.

```
7431 \newcommand*{\glslfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

`\glslongfootnotefont` Only used by the “footnote” styles.

```
7432 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

`\glsxtrabrvfootnote` `\glsxtrabrvfootnote{<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`).

```
7433 \newcommand*{\glsxtrabrvfootnote}[2]{\footnote{#2}}
```

`\glsxtrfootnotename`

```

7434 \newcommand*{\glsxtrfootnotename}{%
7435 \protect\glssabrvfont{\the\glsshorttok}}%
7436 }

```

`footnote` Short form followed by long form in footnote on first use.

```

7437 \newabbreviationstyle{footnote}%
7438 {%
7439 \renewcommand*{\CustomAbbreviationFields}{%
7440 name={\glsxtrfootnotename},
7441 sort={\the\glsshorttok},
7442 description={\the\glslongtok},%

```

```

7443 first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7444 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7445 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7446 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7447 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7448 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

7449 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.

```

7450 \renewcommand*\GlsXtrPostNewAbbreviation}{%
7451 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7452 \glsattribute{\the\glslabeltok}{regular}%
7453 {%
7454 \glssetattribute{\the\glslabeltok}{regular}{false}%
7455 }%
7456 {}%
7457 }%
7458 }%
7459 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7460 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7461 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7462 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7463 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7464 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7465 \renewcommand*\glsxtrfullformat[2]{%
7466 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7467 \ifglsxtrininsertinside\else##2\fi
7468 \protect\glsxtrabbrvfootnote{##1}%
7469 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7470 }%
7471 \renewcommand*\glsxtrfullplformat[2]{%
7472 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7473 \ifglsxtrininsertinside\else##2\fi
7474 \protect\glsxtrabbrvfootnote{##1}%
7475 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7476 }%
7477 \renewcommand*\Glsxtrfullformat[2]{%
7478 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7479 \ifglsxtrininsertinside\else##2\fi
7480 \protect\glsxtrabbrvfootnote{##1}%
7481 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7482 }%
7483 \renewcommand*\Glsxtrfullplformat[2]{%
7484 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

7485 \ifglxtrinsertinside\else##2\fi
7486 \protect\glxtrabbrvfootnote{##1}%
7487 {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7488 }%

```

The first use full form and the inline full form use the short (long) style.

```

7489 \renewcommand*{\glxtrinlinefullformat}[2]{%
7490 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7491 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7492 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
7493 }%
7494 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7495 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7496 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7497 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7498 }%
7499 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7500 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7501 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7502 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
7503 }%
7504 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7505 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7506 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7507 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7508 }%
7509 }

```

short-footnote

```

7510 \letabbreviationstyle{short-footnote}{footnote}

```

postfootnote Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```

7511 \newabbreviationstyle{postfootnote}%
7512 {%
7513 \renewcommand*{\CustomAbbreviationFields}{%
7514 name={\glxtrfootnotename},
7515 sort={\the\glssshorttok},
7516 description={\the\glslongtok},%
7517 first={\protect\glsfirstabbrvfont{\the\glssshorttok}},%
7518 firstplural={\protect\glsfirstabbrvfont{\the\glssshortpltok}},%
7519 plural={\protect\glssabbrvfont{\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.

```

7520 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7521 \csdef{glxtrpostlink\glscategorylabel}{%
7522 \glxtrifwasfirstuse

```


7523 {%

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
7524        \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7525        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
7526        }%
7527        {%
7528        }%
7529        \glshasattribute{\the\glslabeltok}{regular}%
7530        {%
7531        \glissetattribute{\the\glslabeltok}{regular}{false}%
7532        }%
7533        {%
7534        }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7535    \renewcommand*{\glxtrsetupfulldefs}{%
7536    \let\glxtrifwasfirstuse\@secondoftwo
7537    }%
7538 }%
7539 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7540    \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7541    \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
7542    \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7543    \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7544    \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7545    \renewcommand*{\glxtrfullformat}[2]{%
7546    \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7547    \ifglxtrininsertinside\else##2\fi
7548    }%
7549    \renewcommand*{\glxtrfullplformat}[2]{%
7550    \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7551    \ifglxtrininsertinside\else##2\fi
7552    }%
7553    \renewcommand*{\Glsxtrfullformat}[2]{%
7554    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7555    \ifglxtrininsertinside\else##2\fi
7556    }%
7557    \renewcommand*{\Glsxtrfullplformat}[2]{%
7558    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7559    \ifglxtrininsertinside\else##2\fi
7560    }%
```

The first use full form and the inline full form use the short (long) style.

```
7561    \renewcommand*{\glxtrininlinefullformat}[2]{%
```

```

7562 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7563 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7564 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7565 }%
7566 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7567 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7568 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7569 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7570 }%
7571 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7572 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7573 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7574 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7575 }%
7576 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7577 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7578 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7579 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7580 }%
7581 }

```

rt-postfootnote

```
7582 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```

7583 \newcommand*{\glxtrshortnolongname}{%
7584 \protect\glsabbrvfont{\the\glsshorttok}%
7585 }

```

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.

```

7586 \newabbreviationstyle{short}%
7587 {%
7588 \renewcommand*{\CustomAbbreviationFields}{%
7589 name={\glxtrshortnolongname},
7590 sort={\the\glsshorttok},
7591 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7592 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7593 text={\protect\glsabbrvfont{\the\glsshorttok}},
7594 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7595 description={\the\glslongtok}}%
7596 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7597 \glssetAttribute{\the\glslabeltok}{regular}{true}}%
7598 }%
7599 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7600 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7601 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
7602 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7603 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7604 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7605 \renewcommand*{\glxtrinlinefullformat}[2]{%
7606   \protect\glsfirstabbrvfont{\glssaccessshort{##1}}%
7607   \ifglxtrinsertinside##2\fi}%
7608   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7609   \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7610 }%
7611 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7612   \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}%
7613   \ifglxtrinsertinside##2\fi}%
7614   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7615   \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7616 }%
7617 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7618   \protect\glsfirstabbrvfont{\glssaccessshort{##1}}%
7619   \ifglxtrinsertinside##2\fi}%
7620   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7621   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7622 }%
7623 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7624   \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}%
7625   \ifglxtrinsertinside##2\fi}%
7626   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7627   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7628 }%

```

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.

```

7629 \renewcommand*{\glxtrfullformat}[2]{%
7630   \glsfirstabbrvfont{\glssaccessshort{##1}}\ifglxtrinsertinside##2\fi}%
7631   \ifglxtrinsertinside\else##2\fi
7632 }%
7633 \renewcommand*{\glxtrfullplformat}[2]{%
7634   \glsfirstabbrvfont{\glssaccessshortpl{##1}}\ifglxtrinsertinside##2\fi}%
7635   \ifglxtrinsertinside\else##2\fi
7636 }%
7637 \renewcommand*{\Glsxtrfullformat}[2]{%
7638   \glsfirstabbrvfont{\glssaccessshort{##1}}\ifglxtrinsertinside##2\fi}%
7639   \ifglxtrinsertinside\else##2\fi
7640 }%
7641 \renewcommand*{\Glsxtrfullplformat}[2]{%
7642   \glsfirstabbrvfont{\glssaccessshortpl{##1}}\ifglxtrinsertinside##2\fi}%
7643   \ifglxtrinsertinside\else##2\fi
7644 }%

```

7645 }

Set this as the default style for acronyms:

7646 \setabbreviationstyle[acronym]{short}

short-nolong

7647 \letabbreviationstyle{short-nolong}{short}

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

7648 \newabbreviationstyle{short-nolong-noreg}{%

7649 {%

7650 \GlsXtrUseAbbrStyleSetup{short-nolong}%

Unset the regular attribute if it has been set.

7651 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7652 \glshasattribute{\the\glslabeltok}{regular}%

7653 {%

7654 \glsselattribute{\the\glslabeltok}{regular}{false}%

7655 }%

7656 {}%

7657 }%

7658 }%

7659 {%

7660 \GlsXtrUseAbbrStyleFmts{short-nolong}%

7661 }

trshortdescname

7662 \newcommand*{\glxtrshortdescname}{%

7663 \protect\glsabbrvfont{\the\glsshorttok}%

7664 }

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.

7665 \newabbreviationstyle{short-desc}{%

7666 {%

7667 \renewcommand*{\CustomAbbreviationFields}{%

7668 name={\glxtrshortdescname},

7669 sort={\the\glsshorttok},

7670 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},

7671 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},

7672 text={\protect\glsabbrvfont{\the\glsshorttok}},

7673 plural={\protect\glsabbrvfont{\the\glsshortpltok}},

7674 description={\the\glslongtok}}%

7675 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7676 \glsselattribute{\the\glslabeltok}{regular}{true}}%

7677 }%

7678 {%

In case the user wants to mix and match font styles, these are redefined here.

```

7679 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7680 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7681 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7682 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7683 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7684 \renewcommand*{\glxtrinlinefullformat}[2]{%
7685   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7686   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7687   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7688 }%
7689 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7690   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7691   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7692   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7693 }%
7694 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7695   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7696   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7697   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7698 }%
7699 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7700   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7701   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7702   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7703 }%

```

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.

```

7704 \renewcommand*{\glxtrfullformat}[2]{%
7705   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7706   \ifglxtrininsertinside\else##2\fi
7707 }%
7708 \renewcommand*{\glxtrfullplformat}[2]{%
7709   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7710   \ifglxtrininsertinside\else##2\fi
7711 }%
7712 \renewcommand*{\Glsxtrfullformat}[2]{%
7713   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7714   \ifglxtrininsertinside\else##2\fi
7715 }%
7716 \renewcommand*{\Glsxtrfullplformat}[2]{%
7717   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7718   \ifglxtrininsertinside\else##2\fi
7719 }%
7720 }

```

ort-nolong-desc

7721 \letabbreviationstyle{short-nolong-desc}{short-desc}

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

7722 \newabbreviationstyle{short-nolong-desc-noreg}%

7723 {%

7724 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

Unset the regular attribute if it has been set.

7725 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7726 \glshasattribute{\the\glslabeltok}{regular}%

7727 {%

7728 \glsssetattribute{\the\glslabeltok}{regular}{false}%

7729 }%

7730 {}%

7731 }%

7732 }%

7733 {%

7734 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%

7735 }

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

7736 \newabbreviationstyle{nolong-short}%

7737 {%

7738 \GlsXtrUseAbbrStyleSetup{short-nolong}%

7739 }%

7740 {%

7741 \GlsXtrUseAbbrStyleFmts{short-nolong}%

The inline full form displays the long form followed by the short form in parentheses.

7742 \renewcommand*{\glsxtrinlinefullformat}[2]{%

7743 \protect\glsfirstlongfont{\glssaccesslong{##1}}%

7744 \ifglsxtrininsertinside##2\fi}%

7745 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7746 \glsxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}%

7747 }%

7748 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

7749 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%

7750 \ifglsxtrininsertinside##2\fi}%

7751 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7752 \glsxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%

7753 }%

7754 \renewcommand*{\Glsxtrinlinefullformat}[2]{%

7755 \protect\glsfirstlongfont{\glssaccesslong{##1}}%

7756 \ifglsxtrininsertinside##2\fi}%

7757 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7758 \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%

7759 }%

7760 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

7761 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%

```

7762     \ifglxtrinsertinside##2\fi}%
7763     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7764     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7765 }%
7766 }

```

long-short-noreg Like **nolong-short** but doesn't set the regular attribute.

```

7767 \newabbreviationstyle{nolong-short-noreg}%
7768 {%
7769   \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7770   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7771     \glshasattribute{\the\glslabeltok}{regular}%
7772     {%
7773       \glissetattribute{\the\glslabeltok}{regular}{false}%
7774     }%
7775   }%
7776 }%
7777 }%
7778 {%
7779   \GlsXtrUseAbbrStyleFmts{nolong-short}%
7780 }

```

noshortdescname

```

7781 \newcommand*{\glxtrlongnoshortdescname}{%
7782   \protect\glslongfont{\the\glslongtok}%
7783 }

```

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.

```

7784 \newabbreviationstyle{long-desc}%
7785 {%
7786   \renewcommand*{\CustomAbbreviationFields}{%
7787     name={\glxtrlongnoshortdescname},
7788     sort={\the\glslongtok},
7789     first={\protect\glsfirstlongfont{\the\glslongtok}},
7790     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7791     text={\glslongfont{\the\glslongtok}},
7792     plural={\glslongfont{\the\glslongpltok}}}%
7793   }%
7794   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7795     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7796 }%
7797 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7798   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

```

```

7799 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7800 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7801 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7802 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7803 \renewcommand*\glsxtrsubsequentfmt[2]{%
7804   \glslongfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
7805   \ifglsxtrinsertinside \else##2\fi
7806 }%
7807 \renewcommand*\glsxtrsubsequentplfmt[2]{%
7808   \glslongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
7809   \ifglsxtrinsertinside \else##2\fi
7810 }%
7811 \renewcommand*\Glsxtrsubsequentfmt[2]{%
7812   \glslongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
7813   \ifglsxtrinsertinside \else##2\fi
7814 }%
7815 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
7816   \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
7817   \ifglsxtrinsertinside \else##2\fi
7818 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7819 \renewcommand*\glsxtrinlinefullformat[2]{%
7820   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7821   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7822   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7823 }%
7824 \renewcommand*\glsxtrinlinefullplformat[2]{%
7825   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7826   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7827   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7828 }%
7829 \renewcommand*\Glsxtrinlinefullformat[2]{%
7830   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7831   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7832   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7833 }%
7834 \renewcommand*\Glsxtrinlinefullplformat[2]{%
7835   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7836   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7837   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7838 }%

```

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.

```

7839 \renewcommand*\glsxtrfullformat[2]{%
7840   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7841   \ifglsxtrinsertinside\else##2\fi
7842 }%

```



```

7843 \renewcommand*{\glstrfullplformat}[2]{%
7844   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7845   \ifglstrinsertinside\else##2\fi
7846 }%
7847 \renewcommand*{\glstrfullformat}[2]{%
7848   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7849   \ifglstrinsertinside\else##2\fi
7850 }%
7851 \renewcommand*{\glstrfullplformat}[2]{%
7852   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7853   \ifglstrinsertinside\else##2\fi
7854 }%
7855 }

```

`ng-noshort-desc` Provide a synonym that matches similar styles.

```
7856 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

`short-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```

7857 \newabbreviationstyle{long-noshort-desc-noreg}%
7858 {%
7859   \glstrUseAbbrStyleSetup{long-noshort-desc}%
7860   \glstrPostNewAbbreviation{%
7861     \glstrhasattribute{\the\glstrlabeltok}{regular}%
7862     {%
7863       \glstrsetattribute{\the\glstrlabeltok}{regular}{false}%
7864     }%
7865     {}%
7866   }%
7867 }%
7868 {%
7869   \glstrUseAbbrStyleFmts{long-noshort-desc}%
7870 }

```

`longnoshortname`

```

7871 \newcommand*{\glstrlongnoshortname}{%
7872   \protect\glstrabbrvfont{\the\glstrshorttok}%
7873 }

```

`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.

```

7874 \newabbreviationstyle{long}%
7875 {%
7876   \renewcommand*{\CustomAbbreviationFields}{%
7877     name={\glstrlongnoshortname},
7878     sort={\the\glstrshorttok},
7879     first={\protect\glstrfirstlongfont{\the\glstrlongtok}},

```

```

7880     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7881     text={\glslongfont{\the\glslongtok}},
7882     plural={\glslongfont{\the\glslongpltok}},%
7883     description={\the\glslongtok}%
7884 }%
7885 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7886   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7887 }%
7888 {%
7889   \GlsXtrUseAbbrStyleFmts{long-desc}%
7890 }

```

`long-noshort` Provide a synonym that matches similar styles.

```
7891 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```

7892 \newabbreviationstyle{long-noshort-noreg}%
7893 {%
7894   \GlsXtrUseAbbrStyleSetup{long-noshort}%
7895   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7896     \glsattribute{\the\glslabeltok}{regular}%
7897     {%
7898       \glssetattribute{\the\glslabeltok}{regular}{false}%
7899     }}%
7900   {}%
7901 }%
7902 }%
7903 {%
7904   \GlsXtrUseAbbrStyleFmts{long-noshort}%
7905 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glxtrscfont` Maintained for backward-compatibility.

```
7906 \newcommand*{\glxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7907 \newcommand*{\glsabbrvscfont}{\glxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
7908 \newcommand*{\glxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`irstabbrvscfont` Added for consistent naming.

```
7909 \newcommand*{\glsfirstabbrvscfont}{\glxtrfirstscfont}
```

and for the default short form suffix:

`\glxtrscsuffix`

```
7910 \newcommand*{\glxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
7911 \newabbreviationstyle{long-short-sc}%
7912 {%
7913   \renewcommand*{\CustomAbbreviationFields}{%
7914     name={\glxtrlongshortname},
7915     sort={\the\glsshorttok},
7916     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7917       \protect\glxtrfullsep{\the\glslabeltok}%
7918       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7919     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7920       \protect\glxtrfullsep{\the\glslabeltok}%
7921       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7922     plural={\protect\glssabbrvscfont{\the\glsshortpltok}}},%
7923     description={\the\glslongtok}}%
7924   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7925     \glshasattribute{\the\glslabeltok}{regular}%
7926     {%
7927       \glissetattribute{\the\glslabeltok}{regular}{false}%
7928     }%
7929   }%
7930 }%
7931 }%
7932 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
7933 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7934 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
7935 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
7936 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7937 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7938 \renewcommand*{\glxtrfullformat}[2]{%
7939   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7940   \ifglxtrininsertinside\else##2\fi
7941   \glxtrfullsep{##1}%
7942   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
7943 }%
7944 \renewcommand*{\glxtrfullplformat}[2]{%
7945   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7946   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7947   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
7948 }%
7949 \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

7950 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7951 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7952 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7953 }%
7954 \renewcommand*{\Glsxtrfullplformat}[2]{%
7955 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7956 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7957 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7958 }%
7959 }

```

g-short-sc-desc

```

7960 \newabbreviationstyle{long-short-sc-desc}%
7961 {%
7962 \renewcommand*{\CustomAbbreviationFields}{%
7963 name={\glxtrlongshortdescname},
7964 sort={\glxtrlongshortdescsort},%
7965 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7966 \protect\glxtrfullsep{\the\glslabeltok}%
7967 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7968 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7969 \protect\glxtrfullsep{\the\glslabeltok}%
7970 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7971 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7972 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7973 }%

```

Unset the regular attribute if it has been set.

```

7974 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7975 \glshasattribute{\the\glslabeltok}{regular}%
7976 {%
7977 \glissetattribute{\the\glslabeltok}{regular}{false}%
7978 }%
7979 {}%
7980 }%
7981 }%
7982 {%

```

As long-short-sc style:

```

7983 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7984 }

```

Now the short (long) version

```

7985 \newabbreviationstyle{short-sc-long}%
7986 {%
7987 \renewcommand*{\CustomAbbreviationFields}{%
7988 name={\glxtrshortlongname},
7989 sort={\the\glsshorttok},
7990 description={\the\glslongtok},%
7991 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%

```

```

7992 \protect\glxtrfullsep{\the\glslabeltok}%
7993 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7994 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7995 \protect\glxtrfullsep{\the\glslabeltok}%
7996 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7997 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7998 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7999 \glshasattribute{\the\glslabeltok}{regular}%
8000 {%
8001 \glissetattribute{\the\glslabeltok}{regular}{false}%
8002 }%
8003 {}%
8004 }%
8005 }%
8006 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8007 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8008 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8009 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8010 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8011 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8012 \renewcommand*{\glxtrfullformat}[2]{%
8013 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8014 \ifglxtrininsertinside\else##2\fi
8015 \glxtrfullsep{##1}%
8016 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8017 }%
8018 \renewcommand*{\glxtrfullplformat}[2]{%
8019 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8020 \ifglxtrininsertinside\else##2\fi
8021 \glxtrfullsep{##1}%
8022 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8023 }%
8024 \renewcommand*{\Glsxtrfullformat}[2]{%
8025 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8026 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8027 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8028 }%
8029 \renewcommand*{\Glsxtrfullplformat}[2]{%
8030 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8031 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8032 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8033 }%
8034 }

```

As before but user provides description

```

8035 \newabbreviationstyle{short-sc-long-desc}%
8036 {%
8037   \renewcommand*{\CustomAbbreviationFields}{%
8038     name={\glxtrshortlongdescname},
8039     sort={\glxtrshortlongdescsort},
8040     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8041       \protect\glxtrfullsep{\the\glslabeltok}%
8042       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8043     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8044       \protect\glxtrfullsep{\the\glslabeltok}%
8045       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8046     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8047     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8048   }%

```

Unset the regular attribute if it has been set.

```

8049   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8050     \glshasattribute{\the\glslabeltok}{regular}%
8051     {%
8052       \glissetattribute{\the\glslabeltok}{regular}{false}%
8053     }%
8054   }%
8055 }%
8056 }%
8057 {%

```

As short-sc-long style:

```

8058   \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8059 }

```

short-sc

```

8060 \newabbreviationstyle{short-sc}%
8061 {%
8062   \renewcommand*{\CustomAbbreviationFields}{%
8063     name={\glxtrshortnolongname},
8064     sort={\the\glsshorttok},
8065     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8066     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8067     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8068     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8069     description={\the\glslongtok}}%
8070   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8071     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8072 }%
8073 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8074   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8075   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8076   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%

```

```

8077 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8078 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8079 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8080   \protect\glsfirstabbrvscfont{\glsaccesssshort{##1}}%
8081   \ifglsxtrininsertinside##2\fi}%
8082   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8083   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8084 }%
8085 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8086   \protect\glsfirstabbrvscfont{\glsaccesssshortpl{##1}}%
8087   \ifglsxtrininsertinside##2\fi}%
8088   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8089   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8090 }%
8091 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8092   \protect\glsfirstabbrvscfont{\Glsaccesssshort{##1}}%
8093   \ifglsxtrininsertinside##2\fi}%
8094   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8095   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8096 }%
8097 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8098   \protect\glsfirstabbrvscfont{\Glsaccesssshortpl{##1}}%
8099   \ifglsxtrininsertinside##2\fi}%
8100   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8101   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8102 }%

```

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.

```

8103 \renewcommand*{\glsxtrfullformat}[2]{%
8104   \glsfirstabbrvscfont{\glsaccesssshort{##1}}\ifglsxtrininsertinside##2\fi}%
8105   \ifglsxtrininsertinside\else##2\fi
8106 }%
8107 \renewcommand*{\glsxtrfullplformat}[2]{%
8108   \glsfirstabbrvscfont{\glsaccesssshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8109   \ifglsxtrininsertinside\else##2\fi
8110 }%
8111 \renewcommand*{\Glsxtrfullformat}[2]{%
8112   \glsfirstabbrvscfont{\glsaccesssshort{##1}}\ifglsxtrininsertinside##2\fi}%
8113   \ifglsxtrininsertinside\else##2\fi
8114 }%
8115 \renewcommand*{\Glsxtrfullplformat}[2]{%
8116   \glsfirstabbrvscfont{\glsaccesssshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8117   \ifglsxtrininsertinside\else##2\fi
8118 }%
8119 }

```

short-sc-nolong

```
8120 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
8121 \newabbreviationstyle{short-sc-desc}{%
8122 {%
8123   \renewcommand*{\CustomAbbreviationFields}{%
8124     name={\glxtrshortdescname},
8125     sort={\the\glsshorttok},
8126     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8127     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8128     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8129     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8130     description={\the\glslongtok}}%
8131   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8132     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8133 }%
8134 }
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8135 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8136 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8137 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8138 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8139 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8140 \renewcommand*{\glxtrinlinefullformat}[2]{%
8141   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8142   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8143   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8144 }%
8145 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8146   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8147   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8148   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8149 }%
8150 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8151   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8152   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8153   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8154 }%
8155 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8156   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8157   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8158   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8159 }%
```

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.


```

8160 \renewcommand*{\glxtrfullformat}[2]{%
8161   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8162   \ifglxtrinsertinside\else##2\fi
8163 }%
8164 \renewcommand*{\glxtrfullplformat}[2]{%
8165   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8166   \ifglxtrinsertinside\else##2\fi
8167 }%
8168 \renewcommand*{\Glsxtrfullformat}[2]{%
8169   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8170   \ifglxtrinsertinside\else##2\fi
8171 }%
8172 \renewcommand*{\Glsxtrfullplformat}[2]{%
8173   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8174   \ifglxtrinsertinside\else##2\fi
8175 }%
8176 }

```

-sc-nolong-desc

```

8177 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

8178 \newabbreviationstyle{nolong-short-sc}%
8179 {%
8180   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8181 }%
8182 {%
8183   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8184 \renewcommand*{\glxtrinlinefullformat}[2]{%
8185   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8186   \ifglxtrinsertinside##2\fi}%
8187   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8188   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8189 }%
8190 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8191   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8192   \ifglxtrinsertinside##2\fi}%
8193   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8194   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8195 }%
8196 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8197   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8198   \ifglxtrinsertinside##2\fi}%
8199   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8200   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8201 }%
8202 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8203   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%

```

```

8204     \ifglxtrinsertinside##2\fi}%
8205     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8206     \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8207 }%
8208 }

```

`\long-noshort-sc` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

8209 \newabbreviationstyle{long-noshort-sc}%
8210 {%
8211   \renewcommand*{\CustomAbbreviationFields}{%
8212     name={\glxtrlongnoshortname},
8213     sort={\the\glssshorttok},
8214     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8215     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8216     text={\protect\glslongdefaultfont{\the\glslongtok}},
8217     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8218     description={\the\glslongtok}%
8219   }%
8220   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8221     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8222 }%
8223 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8224   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8225   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8226   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8227   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8228   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8229   \renewcommand*{\glxtrsubsequentfmt}[2]{%
8230     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8231     \ifglxtrinsertinside \else##2\fi
8232   }%
8233   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8234     \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8235     \ifglxtrinsertinside \else##2\fi
8236   }%
8237   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8238     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8239     \ifglxtrinsertinside \else##2\fi
8240   }%
8241   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8242     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8243     \ifglxtrinsertinside \else##2\fi
8244   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8245 \renewcommand*{\glxtrinlinefullformat}[2]{%
8246   \glsfirslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8247   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8248   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8249 }%
8250 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8251   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8252   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8253   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8254 }%
8255 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8256   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8257   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8258   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8259 }%
8260 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8261   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8262   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8263   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8264 }%

```

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.

```

8265 \renewcommand*{\glxtrfullformat}[2]{%
8266   \glsfirslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8267   \ifglxtrinsertinside\else##2\fi
8268 }%
8269 \renewcommand*{\glxtrfullplformat}[2]{%
8270   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8271   \ifglxtrinsertinside\else##2\fi
8272 }%
8273 \renewcommand*{\Glsxtrfullformat}[2]{%
8274   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8275   \ifglxtrinsertinside\else##2\fi
8276 }%
8277 \renewcommand*{\Glsxtrfullplformat}[2]{%
8278   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8279   \ifglxtrinsertinside\else##2\fi
8280 }%
8281 }

```

long-sc Backward compatibility:

```
8282 \@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`.

```

8283 \newabbreviationstyle{long-noshort-sc-desc}%
8284 {%
8285   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

8286 }%
8287 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8288 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8289 \renewcommand*\glabbrvfont[1]{\glabbrvscfont{##1}}%
8290 \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvscfont{##1}}%
8291 \renewcommand*\glfirstlongfont[1]{\glfirstlongdefaultfont{##1}}%
8292 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8293 \renewcommand*\glxtrsubsequentfmt}[2]{%
8294   \glslongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8295   \ifglxtrininsertinside \else##2\fi
8296 }%
8297 \renewcommand*\glxtrsubsequentplfmt}[2]{%
8298   \glslongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8299   \ifglxtrininsertinside \else##2\fi
8300 }%
8301 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8302   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8303   \ifglxtrininsertinside \else##2\fi
8304 }%
8305 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8306   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8307   \ifglxtrininsertinside \else##2\fi
8308 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
8309 \renewcommand*\glxtrininlinefullformat}[2]{%
8310   \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8311   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8312   \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshort{##1}}}%
8313 }%
8314 \renewcommand*\glxtrininlinefullplformat}[2]{%
8315   \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8316   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8317   \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshortpl{##1}}}%
8318 }%
8319 \renewcommand*\Glsxtrininlinefullformat}[2]{%
8320   \glfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8321   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8322   \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshort{##1}}}%
8323 }%
8324 \renewcommand*\Glsxtrininlinefullplformat}[2]{%
8325   \glfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8326   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8327   \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshortpl{##1}}}%
8328 }%
```

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.

```

8329 \renewcommand*{\glxstrfullformat}[2]{%
8330   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8331   \ifglxtrinsertinside\else##2\fi
8332 }%
8333 \renewcommand*{\glxstrfullplformat}[2]{%
8334   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8335   \ifglxtrinsertinside\else##2\fi
8336 }%
8337 \renewcommand*{\Glsxtrfullformat}[2]{%
8338   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8339   \ifglxtrinsertinside\else##2\fi
8340 }%
8341 \renewcommand*{\Glsxtrfullplformat}[2]{%
8342   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8343   \ifglxtrinsertinside\else##2\fi
8344 }%
8345 }

```

long-desc-sc Backward compatibility:

```

8346 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

8347 \newabbreviationstyle{short-sc-footnote}%
8348 {%
8349   \renewcommand*{\CustomAbbreviationFields}{%
8350     name={\glxstrfootnotename},
8351     sort={\the\glsshorttok},
8352     description={\the\glslongtok},%
8353     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8354       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8355       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8356     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8357       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8358       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8359     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.

```

8360 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8361   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8362   \glshasattribute{\the\glslabeltok}{regular}%
8363   {%
8364     \glsssetattribute{\the\glslabeltok}{regular}{false}%
8365   }%
8366   {%
8367   }%
8368 }%
8369 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8370 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8371 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
8372 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvscfont{##1}}%
8373 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
8374 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
8375 \renewcommand*{\glxtrfullformat}[2]{%
8376   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8377   \ifglxtrininsertinside\else##2\fi
8378   \protect\glxtrabbrvfootnote{##1}%
8379   {\glfirstlongfootnotefont{\glaccesslong{##1}}}%
8380 }%
8381 \renewcommand*{\glxtrfullplformat}[2]{%
8382   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8383   \ifglxtrininsertinside\else##2\fi
8384   \protect\glxtrabbrvfootnote{##1}%
8385   {\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
8386 }%
8387 \renewcommand*{\Glsxtrfullformat}[2]{%
8388   \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8389   \ifglxtrininsertinside\else##2\fi
8390   \protect\glxtrabbrvfootnote{##1}%
8391   {\glfirstlongfootnotefont{\glaccesslong{##1}}}%
8392 }%
8393 \renewcommand*{\Glsxtrfullplformat}[2]{%
8394   \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8395   \ifglxtrininsertinside\else##2\fi
8396   \protect\glxtrabbrvfootnote{##1}%
8397   {\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
8398 }%
```

The first use full form and the inline full form use the short (long) style.

```
8399 \renewcommand*{\glxtrininlinefullformat}[2]{%
8400   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8401   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8402   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
8403 }%
8404 \renewcommand*{\glxtrininlinefullplformat}[2]{%
8405   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8406   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8407   \glxtrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
8408 }%
8409 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
8410   \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8411   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8412   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
8413 }%
8414 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```

```

8415 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8416 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8417 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8418 }%
8419 }

```

footnote-sc Backward compatibility:

```

8420 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8421 \newabbreviationstyle{short-sc-postfootnote}%
8422 {%
8423 \renewcommand*{\CustomAbbreviationFields}{%
8424 name={\glxtrfootnotename},
8425 sort={\the\glsshorttok},
8426 description={\the\glslongtok},%
8427 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8428 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8429 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.

```

8430 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8431 \csdef{glxtrpostlink\glscategorylabel}{%
8432 \glxtrifwasfirstuse
8433 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8434 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8435 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
8436 }%
8437 {}%
8438 }%
8439 \glshasattribute{\the\glslabeltok}{regular}%
8440 {%
8441 \glissetattribute{\the\glslabeltok}{regular}{false}%
8442 }%
8443 {}%
8444 }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

8445 \renewcommand*{\glxtrsetupfulldefs}{%
8446 \let\glxtrifwasfirstuse\@secondoftwo
8447 }%
8448 }%
8449 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8450 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8451 \renewcommand*\glssabbrvfont[1]{\glssabbrvscfont{##1}}%
8452 \renewcommand*\glssfirstabbrvfont[1]{\glssfirstabbrvscfont{##1}}%
8453 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongfootnotefont{##1}}%
8454 \renewcommand*\glsslongfont[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8455 \renewcommand*\glxtrfullformat}[2]{%
8456   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8457   \ifglxtrininsertinside\else##2\fi
8458 }%
8459 \renewcommand*\glxtrfullplformat}[2]{%
8460   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8461   \ifglxtrininsertinside\else##2\fi
8462 }%
8463 \renewcommand*\Glsxtrfullformat}[2]{%
8464   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8465   \ifglxtrininsertinside\else##2\fi
8466 }%
8467 \renewcommand*\Glsxtrfullplformat}[2]{%
8468   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8469   \ifglxtrininsertinside\else##2\fi
8470 }%

```

The first use full form and the inline full form use the short (long) style.

```

8471 \renewcommand*\glxtrininlinefullformat}[2]{%
8472   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8473   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8474   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8475 }%
8476 \renewcommand*\glxtrininlinefullplformat}[2]{%
8477   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8478   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8479   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8480 }%
8481 \renewcommand*\Glsxtrininlinefullformat}[2]{%
8482   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8483   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8484   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8485 }%
8486 \renewcommand*\Glsxtrininlinefullplformat}[2]{%
8487   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8488   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8489   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8490 }%
8491 }

```

postfootnote-sc Backward compatibility:

```

8492 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```


1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the `relsize` package, which must be loaded by the user. These styles all use:

```
\glxtrsmfont    Maintained for backward compatibility.
8493 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabrvsmfont  Added for consistent naming.
8494 \newcommand*{\glsabrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8495 \newcommand*{\glxtrfirstsmfont}[1]{\glsabrvsmfont{#1}}

irstabrvsmfont  Added for consistent naming.
8496 \newcommand*{\glfirstabrvsmfont}{\glxtrfirstsmfont}

and for the default short form suffix:

\glxtrsmsuffix
8497 \newcommand*{\glxtrsmsuffix}{\glxtrabrvpluralsuffix}

long-short-sm
8498 \newabbreviationstyle{long-short-sm}%
8499 {%
8500   \renewcommand*{\CustomAbbreviationFields}{%
8501     name={\glxtrlongshortname},
8502     sort={\the\glsshorttok},
8503     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
8504       \protect\glxtrfullsep{\the\glslabeltok}%
8505       \glxtrparen{\protect\glfirstabrvsmfont{\the\glsshorttok}}},%
8506     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
8507       \protect\glxtrfullsep{\the\glslabeltok}%
8508       \glxtrparen{\protect\glfirstabrvsmfont{\the\glsshortpltok}}},%
8509     plural={\protect\glsabrvsmfont{\the\glsshortpltok}}},%
8510     description={\the\glslongtok}}%
8511   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8512     \glshasattribute{\the\glslabeltok}{regular}%
8513     {%
8514       \glissetattribute{\the\glslabeltok}{regular}{false}%
8515     }%
8516   }%
8517 }%
8518 }%
8519 {%
8520   \renewcommand*{\glsabrvfont}[1]{\glsabrvsmfont{##1}}%
8521   \renewcommand*{\glfirstabrvfont}[1]{\glfirstabrvsmfont{##1}}%
8522   \renewcommand*{\abrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8523 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8524 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8525 \renewcommand*{\glsxtrfullformat}[2]{%
8526   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8527   \ifglsxtrinsertinside\else##2\fi
8528   \glsxtrfullsep{##1}%
8529   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8530 }%
8531 \renewcommand*{\glsxtrfullplformat}[2]{%
8532   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8533   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8534   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8535 }%
8536 \renewcommand*{\Glsxtrfullformat}[2]{%
8537   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8538   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8539   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8540 }%
8541 \renewcommand*{\Glsxtrfullplformat}[2]{%
8542   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8543   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8544   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8545 }%
8546 }
```

g-short-sm-desc

```
8547 \newabbreviationstyle{long-short-sm-desc}%
8548 {%
8549   \renewcommand*{\CustomAbbreviationFields}{%
8550     name={\glsxtrlongshortdescname},
8551     sort={\glsxtrlongshortdescsort},%
8552     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8553       \protect\glsxtrfullsep{\the\glslabeltok}%
8554       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8555     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8556       \protect\glsxtrfullsep{\the\glslabeltok}%
8557       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8558     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8559     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8560   }%
```

Unset the regular attribute if it has been set.

```
8561 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8562   \glshasattribute{\the\glslabeltok}{regular}%
8563   {%
8564     \glssetattribute{\the\glslabeltok}{regular}{false}%
8565   }%
```

```

8566 {}%
8567 }%
8568 }%
8569 {%

```

As long-short-sm style:

```

8570 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8571 }

```

short-sm-long Now the short (long) version

```

8572 \newabbreviationstyle{short-sm-long}%
8573 {%
8574 \renewcommand*{\CustomAbbreviationFields}{%
8575   name={\glxtrshortlongname},
8576   sort={\the\glsshorttok},
8577   description={\the\gslongtok},%
8578   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8579     \protect\glxtrfullsep{\the\glslabeltok}%
8580     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongtok}}},%
8581   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8582     \protect\glxtrfullsep{\the\glslabeltok}%
8583     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongpltok}}},%
8584   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8585 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8586   \glshasattribute{\the\glslabeltok}{regular}%
8587   {%
8588     \glissetattribute{\the\glslabeltok}{regular}{false}%
8589   }%
8590   {}%
8591 }%
8592 }%
8593 {%
8594 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8595 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8596 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8597 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8598 \renewcommand*{\gslongfont}[1]{\gslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8599 \renewcommand*{\glxtrfullformat}[2]{%
8600   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8601   \ifglxtrininsertinside\else##2\fi
8602   \glxtrfullsep{##1}%
8603   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8604 }%
8605 \renewcommand*{\glxtrfullplformat}[2]{%
8606   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8607   \ifglxtrininsertinside\else##2\fi

```

```

8608 \glxtrfullsep{##1}%
8609 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8610 }%
8611 \renewcommand*{\Glsxtrfullformat}[2]{%
8612 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8613 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8614 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8615 }%
8616 \renewcommand*{\Glsxtrfullplformat}[2]{%
8617 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8618 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8619 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8620 }%
8621 }

```

rt-sm-long-desc As before but user provides description

```

8622 \newabbreviationstyle{short-sm-long-desc}%
8623 {%
8624 \renewcommand*{\CustomAbbreviationFields}{%
8625 name={\glxtrshortlongdescname},
8626 sort={\glxtrshortlongdescsort},
8627 first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}%
8628 \protect\glxtrfullsep{\the\glslabeltok}%
8629 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8630 firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}%
8631 \protect\glxtrfullsep{\the\glslabeltok}%
8632 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8633 text={\protect\glssabbrvsmfont{\the\glssshorttok}},%
8634 plural={\protect\glssabbrvsmfont{\the\glssshortpltok}}}%
8635 }%

```

Unset the regular attribute if it has been set.

```

8636 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8637 \glshasattribute{\the\glslabeltok}{regular}%
8638 {%
8639 \glissetattribute{\the\glslabeltok}{regular}{false}%
8640 }%
8641 {}%
8642 }%
8643 }%
8644 {%

```

As short-sm-long style:

```

8645 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8646 }

```

short-sm

```

8647 \newabbreviationstyle{short-sm}%
8648 {%
8649 \renewcommand*{\CustomAbbreviationFields}{%

```

```

8650     name={\glxtrshortnolongname},
8651     sort={\the\glsshorttok},
8652     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8653     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8654     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8655     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8656     description={\the\glslongtok}}%
8657 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8658   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8659 }%
8660 {%
8661   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8662   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8663   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8664   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8665   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8666 \renewcommand*\glxtrinlinefullformat[2]{%
8667   \protect\glsfirstabbrvsmfont{\glsaccesssshort{##1}}%
8668   \ifglxtrininsertinside##2\fi}%
8669 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8670 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8671 }%
8672 \renewcommand*\glxtrinlinefullplformat[2]{%
8673   \protect\glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}%
8674   \ifglxtrininsertinside##2\fi}%
8675 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8676 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8677 }%
8678 \renewcommand*\Glsxtrinlinefullformat[2]{%
8679   \protect\glsfirstabbrvsmfont{\Glsaccesssshort{##1}}%
8680   \ifglxtrininsertinside##2\fi}%
8681 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8682 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8683 }%
8684 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8685   \protect\glsfirstabbrvsmfont{\Glsaccesssshortpl{##1}}%
8686   \ifglxtrininsertinside##2\fi}%
8687 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8688 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8689 }%

```

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.

```

8690 \renewcommand*\glxtrfullformat[2]{%
8691   \glsfirstabbrvsmfont{\glsaccesssshort{##1}}\ifglxtrininsertinside##2\fi}%
8692   \ifglxtrininsertinside\else##2\fi
8693 }%

```

```

8694 \renewcommand*{\glxtrfullplformat}[2]{%
8695   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8696   \ifglxtrinsertinside\else##2\fi
8697 }%
8698 \renewcommand*{\Glsxtrfullformat}[2]{%
8699   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8700   \ifglxtrinsertinside\else##2\fi
8701 }%
8702 \renewcommand*{\Glsxtrfullplformat}[2]{%
8703   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8704   \ifglxtrinsertinside\else##2\fi
8705 }%
8706 }

```

short-sm-nolong

```

8707 \letabbreviationstyle{short-sm-nolong}{short-sm}

```

short-sm-desc

```

8708 \newabbreviationstyle{short-sm-desc}%
8709 {%
8710   \renewcommand*{\CustomAbbreviationFields}{%
8711     name={\glxtrshortdescname},
8712     sort={\the\glssshorttok},
8713     first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}},
8714     firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}},
8715     text={\protect\glssabbrvsmfont{\the\glssshorttok}},
8716     plural={\protect\glssabbrvsmfont{\the\glssshortpltok}},
8717     description={\the\glslongtok}}%
8718   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8719     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8720 }%
8721 {%
8722   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvsmfont{##1}}%
8723   \renewcommand*{\glssfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8724   \renewcommand*{\abbrvpluralsuffix}{\protect\glssxtrmsuffix}%
8725   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8726   \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8727 \renewcommand*{\glxtrinlinelinefullformat}[2]{%
8728   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8729   \ifglxtrinsertinside\else##2\fi\glssxtrfullsep{##1}%
8730   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
8731 }%
8732 \renewcommand*{\glxtrinlinelinefullplformat}[2]{%
8733   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8734   \ifglxtrinsertinside\else##2\fi\glssxtrfullsep{##1}%
8735   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8736 }%
8737 \renewcommand*{\Glsxtrinlinelinefullformat}[2]{%

```

```

8738 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8739 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8740 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8741 }%
8742 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8743 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8744 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8745 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8746 }%

```

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.

```

8747 \renewcommand*{\glxtrfullformat}[2]{%
8748 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8749 \ifglxtrinsertinside\else##2\fi
8750 }%
8751 \renewcommand*{\glxtrfullplformat}[2]{%
8752 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8753 \ifglxtrinsertinside\else##2\fi
8754 }%
8755 \renewcommand*{\Glsxtrfullformat}[2]{%
8756 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8757 \ifglxtrinsertinside\else##2\fi
8758 }%
8759 \renewcommand*{\Glsxtrfullplformat}[2]{%
8760 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8761 \ifglxtrinsertinside\else##2\fi
8762 }%
8763 }

```

-sm-nolong-desc

```

8764 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8765 \newabbreviationstyle{nolong-short-sm}%
8766 {%
8767 \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8768 }%
8769 {%
8770 \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8771 \renewcommand*{\glxtrinlinefullformat}[2]{%
8772 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8773 \ifglxtrinsertinside##2\fi}%
8774 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8775 \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8776 }%
8777 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8778 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%

```

```

8779 \ifglxtrinsertinside##2\fi}%
8780 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8781 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8782 }%
8783 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8784 \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
8785 \ifglxtrinsertinside##2\fi}%
8786 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8787 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8788 }%
8789 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8790 \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
8791 \ifglxtrinsertinside##2\fi}%
8792 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8793 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8794 }%
8795 }

```

`long-noshort-sm` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8796 \newabbreviationstyle{long-noshort-sm}%
8797 {%
8798 \renewcommand*{\CustomAbbreviationFields}{%
8799 name={\glxtrlongnoshortname},
8800 sort={\the\glsshorttok},
8801 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8802 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8803 text={\protect\glslongdefaultfont{\the\glslongtok}},
8804 plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8805 description={\the\glslongtok}%
8806 }%
8807 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8808 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8809 }%
8810 {%
8811 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvsmfont{##1}}%
8812 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8813 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
8814 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8815 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8816 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8817 \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8818 \ifglxtrinsertinside \else##2\fi
8819 }%
8820 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8821 \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8822 \ifglxtrinsertinside \else##2\fi
8823 }%

```



```

8824 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8825   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8826   \ifglxtrinsertinside \else##2\fi
8827 }%
8828 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8829   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8830   \ifglxtrinsertinside \else##2\fi
8831 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8832 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8833   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8834   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8835   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8836 }%
8837 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8838   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8839   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8840   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8841 }%
8842 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8843   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8844   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8845   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8846 }%
8847 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8848   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8849   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8850   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8851 }%

```

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.

```

8852 \renewcommand*{\Glsxtrfullformat}[2]{%
8853   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8854   \ifglxtrinsertinside\else##2\fi
8855 }%
8856 \renewcommand*{\Glsxtrfullplformat}[2]{%
8857   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8858   \ifglxtrinsertinside\else##2\fi
8859 }%
8860 \renewcommand*{\Glsxtrfullformat}[2]{%
8861   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8862   \ifglxtrinsertinside\else##2\fi
8863 }%
8864 \renewcommand*{\Glsxtrfullplformat}[2]{%
8865   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8866   \ifglxtrinsertinside\else##2\fi
8867 }%
8868 }

```

long-sm Backward compatibility:

```
8869 \@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.

```
8870 \newabbreviationstyle{long-noshort-sm-desc}%
```

```
8871 {%
```

```
8872    \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

```
8873 }%
```

```
8874 {%
```

```
8875    \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
```

```
8876    \renewcommand*\glssfirstabrvfont[1]{\glssfirstabrvsmfont{##1}}%
```

```
8877    \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
```

```
8878    \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
```

```
8879    \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%
```

 The format for subsequent use (not used when the regular attribute is set).

```
8880    \renewcommand*\glsxtrsubsequentfmt}[2]{%
```

```
8881      \glsslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8882      \ifglsxtrininsertinside    \else##2\fi
```

```
8883 }%
```

```
8884    \renewcommand*\glsxtrsubsequentplfmt}[2]{%
```

```
8885      \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8886      \ifglsxtrininsertinside    \else##2\fi
```

```
8887 }%
```

```
8888    \renewcommand*\Glsxtrsubsequentfmt}[2]{%
```

```
8889      \glsslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8890      \ifglsxtrininsertinside    \else##2\fi
```

```
8891 }%
```

```
8892    \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
```

```
8893      \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8894      \ifglsxtrininsertinside    \else##2\fi
```

```
8895 }%
```

 The inline full form displays the long format followed by the short form in parentheses.

```
8896    \renewcommand*\glsxtrinlinefullformat}[2]{%
```

```
8897      \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8898      \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8899      \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshort{##1}}}%
```

```
8900 }%
```

```
8901    \renewcommand*\glsxtrinlinefullplformat}[2]{%
```

```
8902      \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8903      \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8904      \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshortpl{##1}}}%
```

```
8905 }%
```

```
8906    \renewcommand*\Glsxtrinlinefullformat}[2]{%
```

```
8907      \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8908      \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8909      \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshort{##1}}}%
```

```
8910 }%
```

```

8911 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8912   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8913   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8914   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8915 }%

```

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.

```

8916 \renewcommand*{\glxtrfullformat}[2]{%
8917   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8918   \ifglxtrinsertinside\else##2\fi
8919 }%
8920 \renewcommand*{\glxtrfullplformat}[2]{%
8921   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8922   \ifglxtrinsertinside\else##2\fi
8923 }%
8924 \renewcommand*{\Glsxtrfullformat}[2]{%
8925   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8926   \ifglxtrinsertinside\else##2\fi
8927 }%
8928 \renewcommand*{\Glsxtrfullplformat}[2]{%
8929   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8930   \ifglxtrinsertinside\else##2\fi
8931 }%
8932 }

```

long-desc-sm Backward compatibility:

```

8933 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8934 \newabbreviationstyle{short-sm-footnote}%
8935 {%
8936   \renewcommand*{\CustomAbbreviationFields}{%
8937     name={\glxtrfootnotename},
8938     sort={\the\glsshorttok},
8939     description={\the\glslongtok},%
8940     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8941       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8942       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8943     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8944       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8945       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8946     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.

```

8947 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8948   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8949   \glshasattribute{\the\glslabeltok}{regular}%
8950 }%

```

```

8951 \glsssetattribute{\the\glslabelltok}{regular}{false}%
8952 }%
8953 {}%
8954 }%
8955 }%
8956 {%
8957 \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
8958 \renewcommand*\glssfirstabrvfont[1]{\glssfirstabrvsmfont{##1}}%
8959 \renewcommand*\glssabrvpluralsuffix{\protect\glssxtrsmsuffix}%
8960 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongfootnotefont{##1}}%
8961 \renewcommand*\glsslongfont[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8962 \renewcommand*\glssxtrfullformat[2]{%
8963 \glssfirstabrvsmfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
8964 \ifglssxtrininsertinside\else##2\fi
8965 \protect\glssxtrabrvfootnote{##1}%
8966 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8967 }%
8968 \renewcommand*\glssxtrfullplformat[2]{%
8969 \glssfirstabrvsmfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
8970 \ifglssxtrininsertinside\else##2\fi
8971 \protect\glssxtrabrvfootnote{##1}%
8972 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8973 }%
8974 \renewcommand*\Glsxtrfullformat[2]{%
8975 \glssfirstabrvsmfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
8976 \ifglssxtrininsertinside\else##2\fi
8977 \protect\glssxtrabrvfootnote{##1}%
8978 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8979 }%
8980 \renewcommand*\Glsxtrfullplformat[2]{%
8981 \glssfirstabrvsmfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
8982 \ifglssxtrininsertinside\else##2\fi
8983 \protect\glssxtrabrvfootnote{##1}%
8984 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8985 }%

```

The first use full form and the inline full form use the short (long) style.

```

8986 \renewcommand*\glssxtrininlinefullformat[2]{%
8987 \glssfirstabrvsmfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
8988 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
8989 \glssxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8990 }%
8991 \renewcommand*\glssxtrininlinefullplformat[2]{%
8992 \glssfirstabrvsmfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
8993 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
8994 \glssxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8995 }%
8996 \renewcommand*\Glsxtrininlinefullformat[2]{%

```

```

8997 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8998 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8999 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9000 }%
9001 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9002 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9003 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9004 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9005 }%
9006 }

```

footnote-sm Backward compatibility:

```
9007 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

sm-postfootnote

```

9008 \newabbreviationstyle{short-sm-postfootnote}%
9009 {%
9010 \renewcommand*{\CustomAbbreviationFields}{%
9011 name={\glxtrfootnotename},
9012 sort={\the\glsshorttok},
9013 description={\the\glslongtok},%
9014 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9015 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9016 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.

```

9017 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9018 \csdef{glxtrpostlink\glscategorylabel}{%
9019 \glxtrifwasfirstuse
9020 }%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9021 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9022 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
9023 }%
9024 {}%
9025 }%
9026 \glshasattribute{\the\glslabeltok}{regular}%
9027 {%
9028 \glissetattribute{\the\glslabeltok}{regular}{false}%
9029 }%
9030 {}%
9031 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9032 \renewcommand*{\glxtrsetupfulldefs}{%
9033 \let\glxtrifwasfirstuse\@secondoftwo

```

```

9034 }%
9035 }%
9036 {%
9037 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9038 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9039 \renewcommand*\abbrvpluralsuffix{\protect\glsxrmsuffix}%
9040 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9041 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9042 \renewcommand*\glsxtrfullformat}[2]{%
9043   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9044   \ifglsxtrininsertinside\else##2\fi
9045 }%
9046 \renewcommand*\glsxtrfullplformat}[2]{%
9047   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9048   \ifglsxtrininsertinside\else##2\fi
9049 }%
9050 \renewcommand*\Glsxtrfullformat}[2]{%
9051   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9052   \ifglsxtrininsertinside\else##2\fi
9053 }%
9054 \renewcommand*\Glsxtrfullplformat}[2]{%
9055   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9056   \ifglsxtrininsertinside\else##2\fi
9057 }%

```

The first use full form and the inline full form use the short (long) style.

```

9058 \renewcommand*\glsxtrinlinefullformat}[2]{%
9059   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9060   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9061   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9062 }%
9063 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9064   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9065   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9066   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9067 }%
9068 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9069   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9070   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9071   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9072 }%
9073 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9074   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9075   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9076   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9077 }%
9078 }

```

postfootnote-sm Backward compatibility:
 9079 \@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`
 9080 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

`\glsfirstabbrvemfont`
 9081 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

The default short form suffix:

`\glsxtremsuffix`
 9082 \newcommand*{\glsxtremsuffix}{\glsxtrabbrvpluralsuffix}

`\glsfirstlongemfont` Only used by the “long-em” styles.
 9083 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%

`\glslongemfont` Only used by the “long-em” styles.
 9084 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

`\long-short-em` The long form is just set in the default long font.
 9085 \newabbreviationstyle{long-short-em}%
 9086 {%
 9087 \renewcommand*{\CustomAbbreviationFields}{%
 9088 name={\glsxtrlongshortname},
 9089 sort={\the\glsshorttok},
 9090 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
 9091 \protect\glsxtrfullsep{\the\glslabeltok}}%
 9092 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
 9093 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
 9094 \protect\glsxtrfullsep{\the\glslabeltok}}%
 9095 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
 9096 plural={\protect\glsabbrvemfont{\the\glsshorttok}}},%
 9097 description={\the\glslongtok}}%
 9098 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
 9099 \glsattribute{\the\glslabeltok}{regular}}%
 9100 {%
 9101 \glssetattribute{\the\glslabeltok}{regular}{false}}%
 9102 }%
 9103 {}%
 9104 }%
 9105 }%
 9106 {%
 9107 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
 9108 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
 9109 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}}%

Use the default long fonts.

```
9110 \renewcommand*\glstfirstlongfont}[1]{\glstfirstlongdefaultfont{##1}}%
9111 \renewcommand*\glstlongfont}[1]{\glstlongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9112 \renewcommand*\glstxtrfullformat}[2]{%
9113   \glstfirstlongdefaultfont{\glstaccesslong{##1}\ifglstxtrininsertinside##2\fi}%
9114   \ifglstxtrininsertinside\else##2\fi
9115   \glstxtrfullsep{##1}%
9116   \glstxtrparen{\glstfirstabbrvemfont{\glstaccessshort{##1}}}%
9117 }%
9118 \renewcommand*\glstxtrfullplformat}[2]{%
9119   \glstfirstlongdefaultfont{\glstaccesslongpl{##1}\ifglstxtrininsertinside##2\fi}%
9120   \ifglstxtrininsertinside\else##2\fi\glstxtrfullsep{##1}%
9121   \glstxtrparen{\glstfirstabbrvemfont{\glstaccessshortpl{##1}}}%
9122 }%
9123 \renewcommand*\Glstxtrfullformat}[2]{%
9124   \glstfirstlongdefaultfont{\Glstaccesslong{##1}\ifglstxtrininsertinside##2\fi}%
9125   \ifglstxtrininsertinside\else##2\fi\glstxtrfullsep{##1}%
9126   \glstxtrparen{\glstfirstabbrvemfont{\glstaccessshort{##1}}}%
9127 }%
9128 \renewcommand*\Glstxtrfullplformat}[2]{%
9129   \glstfirstlongdefaultfont{\Glstaccesslongpl{##1}\ifglstxtrininsertinside##2\fi}%
9130   \ifglstxtrininsertinside\else##2\fi\glstxtrfullsep{##1}%
9131   \glstxtrparen{\glstfirstabbrvemfont{\glstaccessshortpl{##1}}}%
9132 }%
9133 }
```

g-short-em-desc

```
9134 \newabbreviationstyle{long-short-em-desc}%
9135 {%
9136   \renewcommand*\CustomAbbreviationFields{%
9137     name={\glstxtrlongshortdescname},
9138     sort={\glstxtrlongshortdescsort},%
9139     first={\protect\glstfirstlongdefaultfont{\the\glstlongtok}%
9140       \protect\glstxtrfullsep{\the\glstlabeltok}%
9141       \glstxtrparen{\protect\glstfirstabbrvemfont{\the\glstshorttok}}},%
9142     firstplural={\protect\glstfirstlongdefaultfont{\the\glstlongpltok}%
9143       \protect\glstxtrfullsep{\the\glstlabeltok}%
9144       \glstxtrparen{\protect\glstfirstabbrvemfont{\the\glstshortpltok}}},%
9145     text={\protect\glstabbrvemfont{\the\glstshorttok}},%
9146     plural={\protect\glstabbrvemfont{\the\glstshortpltok}}%
9147   }%
```

Unset the regular attribute if it has been set.

```
9148 \renewcommand*\GlsXtrPostNewAbbreviation{%
9149   \glshasattribute{\the\glstlabeltok}{regular}%
9150   {%
9151     \glstsetattribute{\the\glstlabeltok}{regular}{false}%
9152   }%
```



```

9153 {}%
9154 }%
9155 }%
9156 {%

```

As long-short-em style:

```

9157 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9158 }

```

ong-em-short-em

```

9159 \newabbreviationstyle{long-em-short-em}%
9160 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9161 \renewcommand*{\CustomAbbreviationFields}{%
9162   name={\glsxtrlongshortname},
9163   sort={\the\glsshorttok},
9164   first={\protect\glsfirstlongemfont{\the\glslongtok}%
9165     \protect\glsxtrfullsep{\the\glslabeltok}%
9166     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9167   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9168     \protect\glsxtrfullsep{\the\glslabeltok}%
9169     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9170   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9171   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9172 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9173   \glshasattribute{\the\glslabeltok}{regular}%
9174   {%
9175     \glissetattribute{\the\glslabeltok}{regular}{false}%
9176     }%
9177   }%
9178 }%
9179 }%
9180 {%
9181 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9182 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9183 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9184 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9185 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9186 \renewcommand*{\glsxtrfullformat}[2]{%
9187   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9188   \ifglsxtrininsertinside\else##2\fi
9189   \glsxtrfullsep{##1}%
9190   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9191 }%
9192 \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

9193 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9194 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9195 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9196 }%
9197 \renewcommand*{\Glsxtrfullformat}[2]{%
9198 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9199 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9200 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9201 }%
9202 \renewcommand*{\Glsxtrfullplformat}[2]{%
9203 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9204 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9205 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9206 }%
9207 }

```

m-short-em-desc

```

9208 \newabbreviationstyle{long-em-short-em-desc}%
9209 {%
9210 \renewcommand*{\CustomAbbreviationFields}{%
9211 name={\glxtrlongshortdescname},
9212 sort={\glxtrlongshortdescsort},%
9213 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9214 \protect\glxtrfullsep{\the\glslabeltok}%
9215 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9216 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9217 \protect\glxtrfullsep{\the\glslabeltok}%
9218 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9219 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9220 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9221 }%

```

Unset the regular attribute if it has been set.

```

9222 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9223 \glshasattribute{\the\glslabeltok}{regular}%
9224 {%
9225 \glssetattribute{\the\glslabeltok}{regular}{false}%
9226 }%
9227 {}%
9228 }%
9229 }%
9230 {%
9231 \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9232 }

```

short-em-long Now the short (long) version

```

9233 \newabbreviationstyle{short-em-long}%
9234 {%
9235 \renewcommand*{\CustomAbbreviationFields}{%
9236 name={\glxtrshortlongname},

```

```

9237 sort={\the\glsshorttok},
9238 description={\the\glslongtok},%
9239 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9240 \protect\glsxtrfullsep{\the\glslabeltok}%
9241 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9242 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9243 \protect\glsxtrfullsep{\the\glslabeltok}%
9244 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9245 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9246 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9247 \glsattribute{\the\glslabeltok}{regular}%
9248 {%
9249 \glssetattribute{\the\glslabeltok}{regular}{false}%
9250 }%
9251 {}%
9252 }%
9253 }%
9254 {%

```

Mostly as short-long style:

```

9255 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9256 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9257 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9258 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9259 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9260 \renewcommand*{\glsxtrfullformat}[2]{%
9261 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9262 \ifglxtrinsertinside\else##2\fi
9263 \glsxtrfullsep{##1}%
9264 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9265 }%
9266 \renewcommand*{\glsxtrfullplformat}[2]{%
9267 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9268 \ifglxtrinsertinside\else##2\fi
9269 \glsxtrfullsep{##1}%
9270 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9271 }%
9272 \renewcommand*{\Glsxtrfullformat}[2]{%
9273 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9274 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9275 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9276 }%
9277 \renewcommand*{\Glsxtrfullplformat}[2]{%
9278 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9279 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9280 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9281 }%

```

9282 }

rt-em-long-desc As before but user provides description

```
9283 \newabbreviationstyle{short-em-long-desc}%
9284 {%
9285   \renewcommand*{\CustomAbbreviationFields}{%
9286     name={\glstrshortlongdescname},
9287     sort={\glstrshortlongdescsort},
9288     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9289       \protect\glstrfullsep{\the\glslabeltok}%
9290       \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9291     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9292       \protect\glstrfullsep{\the\glslabeltok}%
9293       \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9294     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9295     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9296   }%
```

Unset the regular attribute if it has been set.

```
9297   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9298     \glshasattribute{\the\glslabeltok}{regular}%
9299     {%
9300       \glissetattribute{\the\glslabeltok}{regular}{false}%
9301     }%
9302   {}%
9303 }%
9304 }%
9305 {%
9306   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9307 }
```

hort-em-long-em

```
9308 \newabbreviationstyle{short-em-long-em}%
9309 {%
9310   \glslongemfont is used in the description since \glsdesc doesn't set the style.
9311   \renewcommand*{\CustomAbbreviationFields}{%
9312     name={\glstrshortlongname},
9313     sort={\the\glsshorttok},
9314     description={\protect\glslongemfont{\the\glslongtok}},%
9315     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9316       \protect\glstrfullsep{\the\glslabeltok}%
9317       \glstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9318     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9319       \protect\glstrfullsep{\the\glslabeltok}%
9320       \glstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9321     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9322   }%
```

Unset the regular attribute if it has been set.

```

9321 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9322   \glshasattribute{\the\glslabeltok}{regular}}%
9323   {%
9324     \glsssetAttribute{\the\glslabeltok}{regular}{false}}%
9325   }%
9326   {}%
9327 }%
9328 }%
9329 {%
9330 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9331 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9332 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9333 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongemfont{##1}}%
9334 \renewcommand*{\glsslongfont}[1]{\glsslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9335 \renewcommand*{\glsxtrfullformat}[2]{%
9336   \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9337   \ifglsxtrininsertinside\else##2\fi
9338   \glsxtrfullsep{##1}}%
9339   \glsxtrparen{\glssfirstlongemfont{\glssaccesslong{##1}}}%
9340 }%
9341 \renewcommand*{\glsxtrfullplformat}[2]{%
9342   \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9343   \ifglsxtrininsertinside\else##2\fi
9344   \glsxtrfullsep{##1}}%
9345   \glsxtrparen{\glssfirstlongemfont{\glssaccesslongpl{##1}}}%
9346 }%
9347 \renewcommand*{\Glsxtrfullformat}[2]{%
9348   \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9349   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
9350   \glsxtrparen{\glssfirstlongemfont{\glssaccesslong{##1}}}%
9351 }%
9352 \renewcommand*{\Glsxtrfullplformat}[2]{%
9353   \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9354   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
9355   \glsxtrparen{\glssfirstlongemfont{\glssaccesslongpl{##1}}}%
9356 }%
9357 }

```

em-long-em-desc

```

9358 \newabbreviationstyle{short-em-long-em-desc}%
9359 {%
9360 \renewcommand*{\CustomAbbreviationFields}{%
9361   name={\glsxtrshortlongdescname},%
9362   sort={\glsxtrshortlongdescsort},%
9363   first={\protect\glssfirstabbrvemfont{\the\glssshorttok}}%
9364   \protect\glsxtrfullsep{\the\glslabeltok}}%
9365   \glsxtrparen{\protect\glssfirstlongemfont{\the\glsslongtok}}},%
9366   firstplural={\protect\glssfirstabbrvemfont{\the\glssshortpltok}}%

```

```

9367 \protect\glxtrfullsep{\the\glslabeltok}%
9368 \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9369 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9370 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9371 }%

```

Unset the regular attribute if it has been set.

```

9372 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9373 \glshasattribute{\the\glslabeltok}{regular}%
9374 {%
9375 \glissetattribute{\the\glslabeltok}{regular}{false}%
9376 }%
9377 {}%
9378 }%
9379 }%
9380 {%
9381 \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9382 }

```

short-em

```

9383 \newabbreviationstyle{short-em}%
9384 {%
9385 \renewcommand*{\CustomAbbreviationFields}{%
9386 name={\glxtrshortnolongname},
9387 sort={\the\glsshorttok},
9388 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9389 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9390 text={\protect\glsabbrvemfont{\the\glsshorttok}},
9391 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9392 description={\the\glslongtok}}%
9393 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9394 \glissetattribute{\the\glslabeltok}{regular}{true}}%
9395 }%
9396 {%
9397 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9398 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9399 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9400 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9401 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9402 \renewcommand*{\glxtrinlinefullformat}[2]{%
9403 \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
9404 \ifglxtrinsertinside##2\fi}%
9405 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9406 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9407 }%
9408 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9409 \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}%
9410 \ifglxtrinsertinside##2\fi}%

```

```

9411 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9412 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9413 }%

9414 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9415 \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
9416 \ifglxtrinsertinside##2\fi}%
9417 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9418 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9419 }%
9420 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9421 \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
9422 \ifglxtrinsertinside##2\fi}%
9423 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9424 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9425 }%

```

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.

```

9426 \renewcommand*{\glxtrfullformat}[2]{%
9427 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9428 \ifglxtrinsertinside\else##2\fi
9429 }%
9430 \renewcommand*{\glxtrfullplformat}[2]{%
9431 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9432 \ifglxtrinsertinside\else##2\fi
9433 }%
9434 \renewcommand*{\Glsxtrfullformat}[2]{%
9435 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9436 \ifglxtrinsertinside\else##2\fi
9437 }%
9438 \renewcommand*{\Glsxtrfullplformat}[2]{%
9439 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9440 \ifglxtrinsertinside\else##2\fi
9441 }%
9442 }

```

short-em-nolong

```

9443 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9444 \newabbreviationstyle{short-em-desc}{%
9445 {%
9446 \renewcommand*{\CustomAbbreviationFields}{%
9447 name={\glxtrshortdescname},
9448 sort={\the\glssshorttok},
9449 first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9450 firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9451 text={\protect\glssabbrvemfont{\the\glssshorttok}},

```

```

9452 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9453 description={\the\glslongtok}}%
9454 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9455   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9456 }%
9457 {%
9458 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9459 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9460 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9461 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9462 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9463 \renewcommand*\glsxtrinlinefullformat[2]{%
9464   \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9465   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9466   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9467 }%
9468 \renewcommand*\glsxtrinlinefullplformat[2]{%
9469   \glsfirstabbrvemfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9470   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9471   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9472 }%
9473 \renewcommand*\Glsxtrinlinefullformat[2]{%
9474   \glsfirstabbrvemfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9475   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9476   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9477 }%
9478 \renewcommand*\Glsxtrinlinefullplformat[2]{%
9479   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9480   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9481   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9482 }%

```

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.

```

9483 \renewcommand*\glsxtrfullformat[2]{%
9484   \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9485   \ifglsxtrininsertinside\else##2\fi
9486 }%
9487 \renewcommand*\glsxtrfullplformat[2]{%
9488   \glsfirstabbrvemfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
9489   \ifglsxtrininsertinside\else##2\fi
9490 }%
9491 \renewcommand*\Glsxtrfullformat[2]{%
9492   \glsfirstabbrvemfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
9493   \ifglsxtrininsertinside\else##2\fi
9494 }%
9495 \renewcommand*\Glsxtrfullplformat[2]{%
9496   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%

```



```

9497     \ifglxtrinsertinside\else##2\fi
9498 }%
9499 }

```

-em-nolong-desc

```

9500 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9501 \newabbreviationstyle{nolong-short-em}%
9502 {%
9503   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9504 }%
9505 {%
9506   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9507 \renewcommand*{\glxtrinlinefullformat}[2]{%
9508   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9509     \ifglxtrinsertinside##2\fi}%
9510   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9511   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9512 }%
9513 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9514   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9515     \ifglxtrinsertinside##2\fi}%
9516   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9517   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9518 }%
9519 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9520   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9521     \ifglxtrinsertinside##2\fi}%
9522   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9523   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9524 }%
9525 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9526   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9527     \ifglxtrinsertinside##2\fi}%
9528   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9529   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9530 }%
9531 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9532 \newabbreviationstyle{long-noshort-em}%
9533 {%
9534   \renewcommand*{\CustomAbbreviationFields}{%
9535     name={\glxtrlongnoshortname},
9536     sort={\the\glsshorttok},
9537     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},

```

```

9538   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9539   text={\protect\glslongdefaultfont{\the\glslongtok}},
9540   plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9541   description={\the\glslongtok}%
9542 }%
9543 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9544   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9545 }%
9546 {%
9547   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9548   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9549   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9550   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9551   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9552   \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9553     \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9554     \ifglsxtrininsertinside \else##2\fi
9555   }%
9556   \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9557     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9558     \ifglsxtrininsertinside \else##2\fi
9559   }%
9560   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9561     \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9562     \ifglsxtrininsertinside \else##2\fi
9563   }%
9564   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9565     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9566     \ifglsxtrininsertinside \else##2\fi
9567   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9568   \renewcommand*{\glsxtrinlinefullformat}[2]{%
9569     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9570     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9571     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9572   }%
9573   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9574     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9575     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9576     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9577   }%
9578   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9579     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9580     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9581     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9582   }%
9583   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

```

```

9584 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9585 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9586 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9587 }%

```

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.

```

9588 \renewcommand*{\glxtrfullformat}[2]{%
9589 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9590 \ifglxtrinsertinside\else##2\fi
9591 }%
9592 \renewcommand*{\glxtrfullplformat}[2]{%
9593 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9594 \ifglxtrinsertinside\else##2\fi
9595 }%
9596 \renewcommand*{\Glsxtrfullformat}[2]{%
9597 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9598 \ifglxtrinsertinside\else##2\fi
9599 }%
9600 \renewcommand*{\Glsxtrfullplformat}[2]{%
9601 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9602 \ifglxtrinsertinside\else##2\fi
9603 }%
9604 }

```

long-em Backward compatibility:

```

9605 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9606 \newabbreviationstyle{long-em-noshort-em}%
9607 {%
9608 \renewcommand*{\CustomAbbreviationFields}{%
9609 name={\glxtrlongnoshortname},
9610 sort={\the\glsshorttok},
9611 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9612 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9613 text={\protect\glslongemfont{\the\glslongtok}},
9614 plural={\protect\glslongemfont{\the\glslongpltok}},%
9615 description={\protect\glslongemfont{\the\glslongtok}}%
9616 }%
9617 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9618 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9619 }%
9620 {%
9621 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9622 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9623 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9624 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9625 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9626 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9627   \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9628   \ifglxtrinsertinside \else##2\fi
9629 }%
9630 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9631   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9632   \ifglxtrinsertinside \else##2\fi
9633 }%
9634 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9635   \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9636   \ifglxtrinsertinside \else##2\fi
9637 }%
9638 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9639   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9640   \ifglxtrinsertinside \else##2\fi
9641 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9642 \renewcommand*{\glxtrinlinefullformat}[2]{%
9643   \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9644   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9645   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9646 }%
9647 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9648   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9649   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9650   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9651 }%
9652 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9653   \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9654   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9655   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9656 }%
9657 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9658   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9659   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9660   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9661 }%

```

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.

```

9662 \renewcommand*{\glxtrfullformat}[2]{%
9663   \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9664   \ifglxtrinsertinside\else##2\fi
9665 }%
9666 \renewcommand*{\glxtrfullplformat}[2]{%
9667   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9668   \ifglxtrinsertinside\else##2\fi
9669 }%

```

```

9670 \renewcommand*{\Glsxtrfullformat}[2]{%
9671   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9672   \ifglxtrinsertinside\else##2\fi
9673 }%
9674 \renewcommand*{\Glsxtrfullplformat}[2]{%
9675   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9676   \ifglxtrinsertinside\else##2\fi
9677 }%
9678 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

9679 \newabbreviationstyle{long-em-noshort-em-noreg}%
9680 {%
9681   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
9682   \GlsXtrPostNewAbbreviation{%
9683     \glshasattribute{\the\glslabeltok}{regular}%
9684     {%
9685       \glissetattribute{\the\glslabeltok}{regular}{false}%
9686     }%
9687   }%
9688 }%
9689 }%
9690 {%
9691   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
9692 }

```

`noshort-em-desc` The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9693 \newabbreviationstyle{long-noshort-em-desc}%
9694 {%
9695   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9696 }%
9697 {%
9698   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9699   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9700   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9701   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9702   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9703 \renewcommand*{\glxtrs subsequentfmt}[2]{%
9704   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9705   \ifglxtrinsertinside \else##2\fi
9706 }%
9707 \renewcommand*{\glxtrs subsequentplfmt}[2]{%
9708   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9709   \ifglxtrinsertinside \else##2\fi
9710 }%

```

```

9711 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9712   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9713   \ifglxtrinsertinside \else##2\fi
9714 }%
9715 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9716   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9717   \ifglxtrinsertinside \else##2\fi
9718 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9719 \renewcommand*{\glxtrinlinefullformat}[2]{%
9720   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9721   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9722   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9723 }%
9724 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9725   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9726   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9727   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9728 }%
9729 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9730   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9731   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9732   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9733 }%
9734 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9735   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9736   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9737   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9738 }%

```

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.

```

9739 \renewcommand*{\glxtrfullformat}[2]{%
9740   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9741   \ifglxtrinsertinside\else##2\fi
9742 }%
9743 \renewcommand*{\glxtrfullplformat}[2]{%
9744   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9745   \ifglxtrinsertinside\else##2\fi
9746 }%
9747 \renewcommand*{\Glsxtrfullformat}[2]{%
9748   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9749   \ifglxtrinsertinside\else##2\fi
9750 }%
9751 \renewcommand*{\Glsxtrfullplformat}[2]{%
9752   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9753   \ifglxtrinsertinside\else##2\fi
9754 }%
9755 }

```

long-desc-em Backward compatibility:

```
9756 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```
9757 \newabbreviationstyle{long-em-noshort-em-desc}%
9758 {%
9759   \renewcommand*{\CustomAbbreviationFields}{%
9760     name={\glxtrlongnoshortdescname},
9761     sort={\the\glslongtok},
9762     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9763     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9764     text={\glslongemfont{\the\glslongtok}},
9765     plural={\glslongemfont{\the\glslongpltok}}}%
9766   }%
9767   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9768     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9769 }%
9770 {%
9771   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9772   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9773   \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9774   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongemfont{##1}}%
9775   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9776 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9777   \glslongemfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9778   \ifglxtrininsertinside \else##2\fi
9779 }%
9780 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9781   \glslongemfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9782   \ifglxtrininsertinside \else##2\fi
9783 }%
9784 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9785   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9786   \ifglxtrininsertinside \else##2\fi
9787 }%
9788 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9789   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9790   \ifglxtrininsertinside \else##2\fi
9791 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9792 \renewcommand*{\glxtrininlinefullformat}[2]{%
9793   \glsfirstlongemfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9794   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9795   \glxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9796 }%
9797 \renewcommand*{\glxtrininlinefullplformat}[2]{%
```

```

9798 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9799 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9800 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9801 }%
9802 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9803 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9804 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9805 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9806 }%
9807 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9808 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9809 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9810 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9811 }%

```

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.

```

9812 \renewcommand*{\glxtrfullformat}[2]{%
9813 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9814 \ifglxtrinsertinside\else##2\fi
9815 }%
9816 \renewcommand*{\glxtrfullplformat}[2]{%
9817 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9818 \ifglxtrinsertinside\else##2\fi
9819 }%
9820 \renewcommand*{\Glsxtrfullformat}[2]{%
9821 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9822 \ifglxtrinsertinside\else##2\fi
9823 }%
9824 \renewcommand*{\Glsxtrfullplformat}[2]{%
9825 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9826 \ifglxtrinsertinside\else##2\fi
9827 }%
9828 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

9829 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9830 {%
9831 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

9832 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9833 \glshasattribute{\the\glslabeltok}{regular}%
9834 {%
9835 \glissetattribute{\the\glslabeltok}{regular}{false}%
9836 }%
9837 {}%
9838 }%
9839 }%
9840 {%

```



```

9841 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9842 }

```

ort-em-footnote

```

9843 \newabbreviationstyle{short-em-footnote}%
9844 {%
9845 \renewcommand*{\CustomAbbreviationFields}{%
9846   name={\glstrfootnotename},
9847   sort={\the\glsshorttok},
9848   description={\the\glslongtok},%
9849   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9850     \protect\glstrabbrvfootnote{\the\glslabeltok}%
9851     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9852   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9853     \protect\glstrabbrvfootnote{\the\glslabeltok}%
9854     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9855   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.

```

9856 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9857 \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9858 \glshasattribute{\the\glslabeltok}{regular}%
9859 {%
9860 \glsssetattribute{\the\glslabeltok}{regular}{false}%
9861 }%
9862 {}%
9863 }%
9864 }%
9865 {%
9866 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9867 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9868 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9869 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9870 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9871 \renewcommand*{\glstrfullformat}[2]{%
9872 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9873 \ifglstrinsertinside\else##2\fi
9874 \protect\glstrabbrvfootnote{##1}%
9875 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9876 }%
9877 \renewcommand*{\glstrfullplformat}[2]{%
9878 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9879 \ifglstrinsertinside\else##2\fi
9880 \protect\glstrabbrvfootnote{##1}%
9881 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9882 }%
9883 \renewcommand*{\GlsXtrfullformat}[2]{%

```

```

9884 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9885 \ifglxtrinsertinside\else##2\fi
9886 \protect\glxtrabbrvfootnote{##1}%
9887 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9888 }%
9889 \renewcommand*{\Glsxtrfullplformat}[2]{%
9890 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9891 \ifglxtrinsertinside\else##2\fi
9892 \protect\glxtrabbrvfootnote{##1}%
9893 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9894 }%

```

The first use full form and the inline full form use the short (long) style.

```

9895 \renewcommand*{\glxtrinlinefullformat}[2]{%
9896 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9897 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9898 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9899 }%
9900 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9901 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9902 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9903 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9904 }%
9905 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9906 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9907 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9908 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9909 }%
9910 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9911 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9912 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9913 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9914 }%
9915 }

```

footnote-em Backward compatibility:

```

9916 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9917 \newabbreviationstyle{short-em-postfootnote}%
9918 {%
9919 \renewcommand*{\CustomAbbreviationFields}{%
9920 name={\glxtrfootnotename},
9921 sort={\the\glsshorttok},
9922 description={\the\glslongtok},%
9923 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
9924 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
9925 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.

```
9926 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9927   \csdef{glxstrpostlink\glscategorylabel}{%
9928     \glxstrifwasfirstuse
9929     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
9930       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9931       {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
9932   }%
9933   {%
9934   }%
9935   \glshasattribute{\the\glslabeltok}{regular}%
9936   {%
9937     \glissetattribute{\the\glslabeltok}{regular}{false}%
9938   }%
9939   {%
9940   }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
9941 \renewcommand*{\glxtrsetupfulldefs}{%
9942   \let\glxstrifwasfirstuse\@secondoftwo
9943 }%
9944 }%
9945 {%
9946 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9947 \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9948 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9949 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
9950 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
9951 \renewcommand*{\glxtrfullformat}[2]{%
9952   \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9953   \ifglxtrininsertinside\else##2\fi
9954 }%
9955 \renewcommand*{\glxtrfullplformat}[2]{%
9956   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9957   \ifglxtrininsertinside\else##2\fi
9958 }%
9959 \renewcommand*{\Glsxtrfullformat}[2]{%
9960   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9961   \ifglxtrininsertinside\else##2\fi
9962 }%
9963 \renewcommand*{\Glsxtrfullplformat}[2]{%
9964   \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9965   \ifglxtrininsertinside\else##2\fi
```

9966 }%

The first use full form and the inline full form use the short (long) style.

```
9967 \renewcommand*{\glxstrinlinefullformat}[2]{%
9968   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxstrinsertinside##2\fi}%
9969   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9970   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9971 }%
9972 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9973   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
9974   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9975   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9976 }%
9977 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9978   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxstrinsertinside##2\fi}%
9979   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9980   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9981 }%
9982 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9983   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
9984   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9985   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9986 }%
9987 }
```

postfootnote-em Backward compatibility:

```
9988 \@glxstr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

1.7.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glxstruserfield Default is the useri field.

```
9989 \newcommand*{\glxstruserfield}{useri}
```

glxstruserparen 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.

```
9990 \ifdef\glscurrentfieldvalue
9991 {
9992   \newcommand*{\glxstruserparen}[2]{%
9993     \glxstrfullsep{#2}%
9994     \glxstrparen
9995       {#1\ifglshasfield{\glxstruserfield}{#2}{, \glscurrentfieldvalue}}}%
9996   }
9997 }
9998 {
9999   \newcommand*{\glxstruserparen}[2]{%
```

```

10000 \glxtrfullsep{#2}%
10001 \glxtrparen
10002 {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
10003 }
10004 }

```

Font used for short form:

lsabbrvuserfont

```
10005 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
10006 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
10007 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10008 \newcommand*{\glfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10009 \newcommand*{\glxtrusersuffix}{\glxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10010 \newcommand*{\gluserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```

10011 \newabbreviationstyle{long-short-user}%
10012 {%
10013 \renewcommand*{\CustomAbbreviationFields}{%
10014 name={\glxtrlongshortname},
10015 sort={\the\glsshorttok},
10016 first={\protect\glfirstlonguserfont{\the\glslongtok}%
10017 \protect\glxtruserparen{\protect\glfirstabbrvuserfont{\the\glsshorttok}}%
10018 {\the\glslabeltok}},%
10019 firstplural={\protect\glfirstlonguserfont{\the\glslongpltok}%
10020 \protect\glxtruserparen
10021 {\protect\glfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10022 plural={\protect\glabbrvuserfont{\the\glsshortpltok}},%
10023 description={\protect\gluserdescription{\the\glslongtok}%
10024 {\the\glslabeltok}}}%

```

Unset the regular attribute if it has been set.

```

10025 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10026   \glshasattribute{\the\glslabeltok}{regular}%
10027   {%
10028     \glsssetattribute{\the\glslabeltok}{regular}{false}%
10029   }%
10030   {}%
10031 }%
10032 }%
10033 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10034 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10035 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvuserfont{##1}}%
10036 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvuserfont{##1}}%
10037 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlonguserfont{##1}}%
10038 \renewcommand*{\glsslongfont}[1]{\glsslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10039 \renewcommand*{\glsxtrfullformat}[2]{%
10040   \glssfirstlonguserfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10041   \ifglsxtrininsertinside\else##2\fi
10042   \glsxtruserparen{\glssfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10043 }%
10044 \renewcommand*{\glsxtrfullplformat}[2]{%
10045   \glssfirstlonguserfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10046   \ifglsxtrininsertinside\else##2\fi
10047   \glsxtruserparen{\glssfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10048 }%
10049 \renewcommand*{\Glsxtrfullformat}[2]{%
10050   \glssfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10051   \ifglsxtrininsertinside\else##2\fi
10052   \glsxtruserparen{\glssfirstabbrvuserfont{\Glsaccessshort{##1}}}{##1}%
10053 }%
10054 \renewcommand*{\Glsxtrfullplformat}[2]{%
10055   \glssfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10056   \ifglsxtrininsertinside\else##2\fi
10057   \glsxtruserparen{\glssfirstabbrvuserfont{\Glsaccessshortpl{##1}}}{##1}%
10058 }%
10059 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10060 \newabbreviationstyle{long-postshort-user}%
10061 {%
10062   \renewcommand*{\CustomAbbreviationFields}{%
10063     name={\glsxtrlongshortname},
10064     sort={\the\glssshorttok},
10065     first={\protect\glssfirstlonguserfont{\the\glsslongtok}},%
10066     firstplural={\protect\glssfirstlonguserfont{\the\glsslongpltok}},%

```

```

10067 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10068 description={\protect\glsuserdescription{\the\glslongtok}%
10069 {\the\glslabeltok}}}%
10070 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10071 \csdef{glsxtrpostlink\glscategorylabel}{%
10072 \glsxtrifwasfirstuse
10073 {%
10074 \glsxtruserparen
10075 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10076 {\glslabel}%
10077 }%
10078 {}%
10079 }%
10080 \glschasattribute{\the\glslabeltok}{regular}%
10081 {%
10082 \glssetattribute{\the\glslabeltok}{regular}{false}%
10083 }%
10084 {}%
10085 }%
10086 }%
10087 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10088 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10089 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10090 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10091 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10092 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10093 \renewcommand*{\glsxtrfullformat}[2]{%
10094 \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10095 \ifglsxtrininsertinside\else##2\fi
10096 }%
10097 \renewcommand*{\glsxtrfullplformat}[2]{%
10098 \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10099 \ifglsxtrininsertinside\else##2\fi
10100 }%
10101 \renewcommand*{\Glsxtrfullformat}[2]{%
10102 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10103 \ifglsxtrininsertinside\else##2\fi
10104 }%
10105 \renewcommand*{\Glsxtrfullplformat}[2]{%
10106 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10107 \ifglsxtrininsertinside\else##2\fi
10108 }%

```

In-line format:

```

10109 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10110 \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%

```

```

10111 \ifglxtrinsertinside\else##2\fi
10112 \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10113 }%
10114 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10115 \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10116 \ifglxtrinsertinside\else##2\fi
10117 \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10118 }%
10119 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10120 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10121 \ifglxtrinsertinside\else##2\fi
10122 \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10123 }%
10124 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10125 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10126 \ifglxtrinsertinside\else##2\fi
10127 \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10128 }%
10129 }

```

ortuserdescname

```

10130 \newcommand*{\glxtrlongshortuserdescname}{%
10131 \protect\glslonguserfont{\the\glslongtok}%
10132 \protect\glxtruserparen
10133 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10134 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10135 \newabbreviationstyle{long-postshort-user-desc}%
10136 {%
10137 \renewcommand*{\CustomAbbreviationFields}{%
10138 name={\glxtrlongshortuserdescname},
10139 sort={\the\glslongtok},
10140 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10141 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10142 text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10143 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10144 }%
10145 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10146 \csdef{glxtrpostlink\glscategorylabel}{%
10147 \glxtrifwasfirstuse
10148 {%
10149 \glxtruserparen
10150 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10151 {\glslabel}%
10152 }%
10153 }%
10154 }%

```



```

10155 \glshasattribute{\the\glslabeltok}{regular}%
10156 {%
10157 \glissetattribute{\the\glslabeltok}{regular}{false}%
10158 }%
10159 {}%
10160 }%
10161 }%
10162 {%
10163 \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10164 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10165 \newabbreviationstyle{short-postlong-user}%
10166 {%
10167 \renewcommand*{\CustomAbbreviationFields}{%
10168   name={\glsxtrshortlongname},
10169   sort={\the\glsshorttok},
10170   first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10171   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10172   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10173   description={\protect\glsuserdescription{\the\glslongtok}%
10174     {\the\glslabeltok}}}%
10175 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10176   \csdef{glsxtrpostlink\glscategorylabel}{%
10177     \glsxtrifwasfirstuse
10178     {%
10179       \glsxtruserparen
10180       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10181       {\glslabel}%
10182     }%
10183     {}%
10184   }%
10185   \glshasattribute{\the\glslabeltok}{regular}%
10186   {%
10187     \glissetattribute{\the\glslabeltok}{regular}{false}%
10188   }%
10189   {}%
10190 }%
10191 }%
10192 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10193 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10194 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10195 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10196 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10197 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10198 \renewcommand*{\glxtrfullformat}[2]{%
10199   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10200   \ifglxtrininsertinside\else##2\fi
10201 }%
10202 \renewcommand*{\glxtrfullplformat}[2]{%
10203   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10204   \ifglxtrininsertinside\else##2\fi
10205 }%
10206 \renewcommand*{\Glsxtrfullformat}[2]{%
10207   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10208   \ifglxtrininsertinside\else##2\fi
10209 }%
10210 \renewcommand*{\Glsxtrfullplformat}[2]{%
10211   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10212   \ifglxtrininsertinside\else##2\fi
10213 }%

```

In-line format:

```

10214 \renewcommand*{\glxtrinlinefullformat}[2]{%
10215   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10216   \ifglxtrininsertinside\else##2\fi
10217   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10218 }%
10219 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10220   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10221   \ifglxtrininsertinside\else##2\fi
10222   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10223 }%
10224 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10225   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10226   \ifglxtrininsertinside\else##2\fi
10227   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10228 }%
10229 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10230   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10231   \ifglxtrininsertinside\else##2\fi
10232   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10233 }%
10234 }

```

onguserdesname

```

10235 \newcommand*{\glxtrshortlonguserdesname}{%
10236   \protect\glssabbrvuserfont{\the\glssshorttok}%
10237   \protect\glxtruserparen
10238     {\protect\glslonguserfont{\the\glslongpltok}}%
10239     {\the\glslabeltok}%
10240 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10241 \newabbreviationstyle{short-postlong-user-desc}%
10242 {%
10243   \renewcommand*{\CustomAbbreviationFields}{%
10244     name={\glxtrshortlonguserdescname},
10245     sort={\the\glsshorttok},
10246     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10247     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10248     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10249     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10250   }%
10251   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10252     \csdef{glxtrpostlink\glscategorylabel}{%
10253       \glxtrifwasfirstuse
10254       {%
10255         \glxtruserparen
10256         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}}%
10257       {\glslabel}}%
10258     }%
10259   }%
10260 }%
10261 \glshasattribute{\the\glslabeltok}{regular}%
10262 {%
10263   \glssetattribute{\the\glslabeltok}{regular}{false}%
10264 }%
10265 {}%
10266 }%
10267 }%
10268 {%
10269   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10270 }

```

short-user-desc

```

10271 \newabbreviationstyle{long-short-user-desc}%
10272 {%
10273   \renewcommand*{\CustomAbbreviationFields}{%
10274     name={\glxtrlongshortuserdescname},
10275     sort={\glxtrlongshortdescsort},%

10276     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10277       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10278     {\the\glslabeltok}},%
10279     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10280       \protect\glxtruserparen
10281       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10282     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10283     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10284   }%

```

Unset the regular attribute if it has been set.

```

10285 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10286   \glshasattribute{\the\glslabeltok}{regular}%
10287   {%
10288     \glissetattribute{\the\glslabeltok}{regular}{false}%
10289   }%
10290   {}%
10291 }%
10292 }%
10293 {%
10294   \GlsXtrUseAbbrStyleFmts{long-short-user}%
10295 }

```

short-long-user

```

10296 \newabbreviationstyle{short-long-user}%
10297 {%
    \glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in
    \glsuserdescription.)
10298 \renewcommand*{\CustomAbbreviationFields}{%
10299   name={\glsxtrshortlongname},
10300   sort={\the\glsshorttok},
10301   description={\protect\glsuserdescription{\the\glslongtok}%
10302     {\the\glslabeltok}},%
10303   first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10304     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10305     {\the\glslabeltok}},%
10306   firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10307     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10308     {\the\glslabeltok}},%
10309   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

    Unset the regular attribute if it has been set.
10310 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10311   \glshasattribute{\the\glslabeltok}{regular}%
10312   {%
10313     \glissetattribute{\the\glslabeltok}{regular}{false}%
10314   }%
10315   {}%
10316 }%
10317 }%
10318 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10319 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10320 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10321 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10322 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10323 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

10324 \renewcommand*{\glxtrfullformat}[2]{%
10325   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10326   \ifglxtrinsertinside\else##2\fi
10327   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10328 }%
10329 \renewcommand*{\glxtrfullplformat}[2]{%
10330   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10331   \ifglxtrinsertinside\else##2\fi
10332   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10333 }%
10334 \renewcommand*{\Glsxtrfullformat}[2]{%
10335   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10336   \ifglxtrinsertinside\else##2\fi
10337   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10338 }%
10339 \renewcommand*{\Glsxtrfullplformat}[2]{%
10340   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10341   \ifglxtrinsertinside\else##2\fi
10342   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10343 }%
10344 }

```

-long-user-desc

```

10345 \newabbreviationstyle{short-long-user-desc}%
10346 {%
10347   \renewcommand*{\CustomAbbreviationFields}{%
10348     name={\glxtrshortlonguserdescname},
10349     sort={\glxtrshortlongdescsort},%
10350     first={\protect\glsfirstabbrvuserfont{\the\glssshorttok}}%
10351     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10352     {\the\glslabeltok}},%
10353     firstplural={\protect\glsfirstabbrvuserfont{\the\glssshortpltok}}%
10354     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10355     {\the\glslabeltok}},%
10356     text={\protect\glsabbrvfont{\the\glssshorttok}},%
10357     plural={\protect\glsabbrvfont{\the\glssshortpltok}}}%
10358   }%

```

Unset the regular attribute if it has been set.

```

10359 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10360   \glshasattribute{\the\glslabeltok}{regular}%
10361   {%
10362     \glissetattribute{\the\glslabeltok}{regular}{false}%
10363   }%
10364   {}%
10365 }%
10366 }%
10367 {%

```

```
10368 \GlsXtrUseAbbrStyleFmts{short-long-user}%
10369 }
```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glstrlong` set `\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.

```
10370 \newrobustcmd*{\glstrifhyphenstart}[3]{%
10371   \ifx\glsinsert#1\relax
10372   \expandafter\@glstrifhyphenstart#1\relax\relax
10373   \@end@glstrifhyphenstart{#2}{#3}%
10374   \else
10375   \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%
10376   \fi
10377 }
```

`\trifhyphenstart`

```
10378 \def\@glstrifhyphenstart#1#2\@end@glstrifhyphenstart#3#4{%
10379   \ifx-#1\relax#3\else #4\fi
10380 }
```

`\trlonghyphenshort`

```
\glstrlonghyphenshort{<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.

```
10381 \newcommand*{\glstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10382 {%
```

If `<insert>` starts with a hyphen, redefine `\glstrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glstrwordsep` if `<insert>` doesn't start with a hyphen.

```
10383   \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{}%
10384   \glsfirstlonghyphenfont{#2\ifglstrinsertinside{#4}\fi}%
10385   \ifglstrinsertinside\else{#4}\fi
10386   \glstrfullsep{#1}%
10387   \glstrparen{\glsfirstabbrvhyphenfont{#3\ifglstrinsertinside{#4}\fi}%
10388   \ifglstrinsertinside\else{#4}\fi}%
10389 }%
10390 }
```

abbrvhyphenfont

```
10391 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10392 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10393 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

tlonghyphenfont

```
10394 \newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10395 \newcommand*{\glsxtrhyphensuffix}{\glsxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the markwords attribute.

```
10396 \newabbreviationstyle{long-hyphen-short-hyphen}%
10397 {%
10398   \renewcommand*{\CustomAbbreviationFields}{%
10399     name={\glsxtrlongshortname},
10400     sort={\the\glsshorttok},
10401     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10402       \protect\glsxtrfullsep{\the\glslabeltok}%
10403       \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10404     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10405       \protect\glsxtrfullsep{\the\glslabeltok}%
10406       \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10407     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10408     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10409   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10410     \glshasattribute{\the\glslabeltok}{regular}%
10411     {%
10412       \glssetattribute{\the\glslabeltok}{regular}{false}%
10413     }%
10414   }%
10415 }%
10416 }%
10417 {%
10418   \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
10419   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10420   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10421   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10422   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline form are the same for this style.

```

10423 \renewcommand*{\glxstrfullformat}[2]{%
10424   \glxstrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10425 }%
10426 \renewcommand*{\glxstrfullplformat}[2]{%
10427   \glxstrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
10428   {\glsaccessshortpl{##1}}{##2}%
10429 }%
10430 \renewcommand*{\Glsxstrfullformat}[2]{%
10431   \glxstrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10432 }%
10433 \renewcommand*{\Glsxstrfullplformat}[2]{%
10434   \glxstrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10435   {\glsaccessshortpl{##1}}{##2}%
10436 }%
10437 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10438 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10439 {%
10440   \renewcommand*{\CustomAbbreviationFields}{%
10441     name={\glxstrlongshortdescname},
10442     sort={\glxstrlongshortdescsort},
10443     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}}%
10444     \protect\glxstrfullsep{\the\glslabeltok}}%
10445     \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10446     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}}%
10447     \protect\glxstrfullsep{\the\glslabeltok}}%
10448     \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10449     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
10450     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10451   }%

```

Unset the regular attribute if it has been set.

```

10452 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10453   \glshasattribute{\the\glslabeltok}{regular}%
10454   {%
10455     \glissetattribute{\the\glslabeltok}{regular}{false}%
10456   }%
10457   {}%
10458 }%
10459 }%
10460 {%
10461   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10462 }

```

onghyphennoshort `\glxstrlonghyphennoshort{<label>}{<long>}{<insert>}`


```
10463 \newcommand*{\glxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
10464 {%
```

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.

```
10465 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
10466 \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10467 \ifglxtrininsertinside\else{#3}\fi
10468 }%
10469 }
```

`short-desc-noreg` This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```
10470 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10471 {%
10472 \renewcommand*{\CustomAbbreviationFields}{%
10473   name={\glxtrlongnoshortdescname},
10474   sort={\expandonce\glxtrorglong},
10475   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10476   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10477   plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10478 }%
```

Unset the regular attribute if it has been set.

```
10479 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10480 \glshasattribute{\the\glslabeltok}{regular}%
10481 {%
10482 \glissetattribute{\the\glslabeltok}{regular}{false}%
10483 }%
10484 {}%
10485 }%
10486 }%
10487 {%
10488 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10489 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10490 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
10491 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10492 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10493 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
10494 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10495 \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10496 }%
```

```

10497 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10498   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10499 }%
10500 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10501   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10502 }%
10503 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10504   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10505 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10506 \renewcommand*{\glxtrinlinefullformat}[2]{%
10507   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10508   \glxtrfullsep{##1}%
10509   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10510 }%
10511 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10512   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10513   \glxtrfullsep{##1}%
10514   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
10515 }%
10516 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10517   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10518   \glxtrfullsep{##1}%
10519   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10520 }%
10521 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10522   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10523   \glxtrfullsep{##1}%
10524   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
10525 }%

```

The first use full form only displays the long form.

```

10526 \renewcommand*{\glxtrfullformat}[2]{%
10527   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10528 }%
10529 \renewcommand*{\glxtrfullplformat}[2]{%
10530   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10531 }%
10532 \renewcommand*{\Glsxtrfullformat}[2]{%
10533   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10534 }%
10535 \renewcommand*{\Glsxtrfullplformat}[2]{%
10536   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10537 }%
10538 }

```

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.

```

10539 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10540 {%
10541   \renewcommand*{\CustomAbbreviationFields}{%
10542     name={\glxtrlongnoshortname},
10543     sort={\the\glsshorttok},
10544     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10545     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10546     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10547     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10548     description={\the\glslongtok}%
10549   }%

```

Unset the regular attribute if it has been set.

```

10550   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10551     \glshasattribute{\the\glslabeltok}{regular}%
10552     {%
10553       \glsetattribute{\the\glslabeltok}{regular}{false}%
10554     }%
10555   }%
10556 }%
10557 }%
10558 {%
10559   \GlsXtrUseAbbrStyleFmts{long-desc}%
10560 }

```

`\glxtrlonghyphen` `\glxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10561 \newcommand*{\glxtrlonghyphen}[3]{%
  Grouping is needed to localise the redefinitions.
10562 {%
10563   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
10564   \glsfirstlonghyphenfont{#1}%
10565 }%
10566 }

```

`\glxtrposthyphenshort` `\glxtrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glxtrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```

10567 \newcommand*{\glxtrposthyphenshort}[2]{%
10568 {%

```

```

10569 \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
10570 \ifglxtrinsertinside{\glsfirslonghyphenfont{#2}}\else{#2}\fi
10571 \glxtrfullsep{#1}%
10572 \glxtrparen
10573 {\glsfirabbrvhyphenfont{\glsentryshort{#1}}\ifglxtrinsertinside{#2}\fi}%
10574 \ifglxtrinsertinside\else{#2}\fi
10575 }%
10576 }%
10577 }

```

hyphensubsequent

```
\glxtrposthyphensubsequent{<label>}{<insert>}
```

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10578 \newcommand*{\glxtrposthyphensubsequent}[2]{%
10579 \glsabbrvfont{\ifglxtrinsertinside {#2}\fi}%
10580 \ifglxtrinsertinside \else{#2}\fi
10581 }

```

postshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10582 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10583 {%
10584 \renewcommand*{\CustomAbbreviationFields}{%
10585 name={\glxtrlongshortname},
10586 sort={\the\glsshorttok},
10587 first={\protect\glsfirslonghyphenfont{\the\glslongtok}},%
10588 firstplural={\protect\glsfirslonghyphenfont{\the\glslongpltok}},%
10589 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10590 description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10591 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10592 \csdef{glxtrpostlink\glscategorylabel}{%
10593 \glxtrifwasfirstuse
10594 {%
10595 \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10596 }%
10597 }%

```

Put the insertion into the post-link:

```

10598 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10599 }%
10600 }%
10601 \glshasattribute{\the\glslabeltok}{regular}%
10602 {%
10603 \glissetattribute{\the\glslabeltok}{regular}{false}%
10604 }%
10605 {}%
10606 }%

```

10607 }%
 10608 {%

In case the user wants to mix and match font styles, these are redefined here.

```
10609 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
10610 \renewcommand*\glsabbrvfont[1]{\glsabbrvhyphenfont{##1}}%
10611 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvhyphenfont{##1}}%
10612 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
10613 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
10614 \renewcommand*\glxtrs subsequentfmt[2]{%
10615   \glsabbrvfont{\glsaccessshort{##1}}%
10616 }%
10617 \renewcommand*\glxtrs subsequentplfmt[2]{%
10618   \glsabbrvfont{\glsaccessshortpl{##1}}%
10619 }%
10620 \renewcommand*\Glsxtrs subsequentfmt[2]{%
10621   \glsabbrvfont{\Glsaccessshort{##1}}%
10622 }%
10623 \renewcommand*\Glsxtrs subsequentplfmt[2]{%
10624   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10625 }%
```

First use full form:

```
10626 \renewcommand*\glxtrfullformat[2]{%
10627   \glxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10628 }%
10629 \renewcommand*\glxtrfullplformat[2]{%
10630   \glxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10631 }%
10632 \renewcommand*\Glsxtrfullformat[2]{%
10633   \glxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10634 }%
10635 \renewcommand*\Glsxtrfullplformat[2]{%
10636   \glxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10637 }%
```

In-line format.

```
10638 \renewcommand*\glxtrinlinefullformat[2]{%
10639   \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10640   \ifglxtrininsertinside{##2}\fi}%
10641   \ifglxtrininsertinside \else{##2}\fi
10642 }%
10643 \renewcommand*\glxtrinlinefullplformat[2]{%
10644   \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10645   \ifglxtrininsertinside{##2}\fi}%
10646   \ifglxtrininsertinside \else{##2}\fi
10647 }%
10648 \renewcommand*\Glsxtrinlinefullformat[2]{%
10649   \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
```

```

10650     \ifglxtrinsertinside{##2}\fi}%
10651     \ifglxtrinsertinside \else{##2}\fi
10652 }%
10653 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10654     \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}%
10655         \ifglxtrinsertinside{##2}\fi}%
10656     \ifglxtrinsertinside \else{##2}\fi
10657 }%
10658 }

```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10659 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10660 {%
10661     \renewcommand*{\CustomAbbreviationFields}{%
10662         name={\glxtrlongshortdescname},
10663         sort={\glxtrlongshortdescsort},%
10664         first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10665         firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10666         text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10667         plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10668 }%
10669 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10670     \csdef{glxtrpostlink\glscategorylabel}{%
10671         \glxtrifwasfirstuse
10672         {%
10673             \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10674         }%
10675         {%

```

Put the insertion into the post-link:

```

10676         \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10677     }%
10678 }%
10679 \glshasattribute{\the\glslabeltok}{regular}%
10680 {%
10681     \glissetattribute{\the\glslabeltok}{regular}{false}%
10682 }%
10683 {}%
10684 }%
10685 }%
10686 {%
10687     \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10688 }

```

`rshorthyphenlong`

```
\glxtrshorthyphenlong{<label>}{<short>}{<long>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be

the first letter uppercase form.

```
10689 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10690 {%
```

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.)

```
10691 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
```

```
10692 \glsfirtabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
```

```
10693 \ifglxtrininsertinside\else{#4}\fi
```

```
10694 \glxtrfullsep{#1}%
```

```
10695 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
```

```
10696 \ifglxtrininsertinside\else{#4}\fi}%
```

```
10697 }%
```

```
10698 }
```

hen-long-hyphen Designed for use with the markwords attribute.

```
10699 \newabbreviationstyle{short-hyphen-long-hyphen}%
```

```
10700 {%
```

```
10701 \renewcommand*{\CustomAbbreviationFields}{%
```

```
10702 name={\glxtrshortlongname},
```

```
10703 sort={\the\glsshorttok},
```

```
10704 first={\protect\glsfirtabbrvhyphenfont{\the\glsshorttok}}%
```

```
10705 \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
10706 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
```

```
10707 firstplural={\protect\glsfirtabbrvhyphenfont{\the\glsshortpltok}}%
```

```
10708 \protect\glxtrfullsep{\the\glslabeltok}}%
```

```
10709 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
```

```
10710 plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}}},%
```

```
10711 description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10712 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
10713 \glshasattribute{\the\glslabeltok}{regular}}%
```

```
10714 {%
```

```
10715 \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
10716 }%
```

```
10717 {}%
```

```
10718 }%
```

```
10719 }%
```

```
10720 {%
```

```
10721 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
```

```
10722 \renewcommand*{\glssabrvfont}[1]{\glssabrvhyphenfont{##1}}%
```

```
10723 \renewcommand*{\glsfirtabbrvfont}[1]{\glsfirtabbrvhyphenfont{##1}}%
```

```
10724 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
```

```
10725 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```

10726 \renewcommand*{\glxtrfullformat}[2]{%
10727   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10728 }%
10729 \renewcommand*{\glxtrfullplformat}[2]{%
10730   \glxtrshorthyphenlong{##1}%
10731   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10732 }%
10733 \renewcommand*{\Glsxtrfullformat}[2]{%
10734   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10735 }%
10736 \renewcommand*{\Glsxtrfullplformat}[2]{%
10737   \glxtrshorthyphenlong{##1}%
10738   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10739 }%
10740 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10741 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10742 {%
10743   \renewcommand*{\CustomAbbreviationFields}{%
10744     name={\glxtrshortlongdescname},
10745     sort={\glxtrshortlongdescsort},
10746     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
10747     \protect\glxtrfullsep{\the\glslabeltok}}%
10748     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10749     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
10750     \protect\glxtrfullsep{\the\glslabeltok}}%
10751     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10752     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
10753     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10754 }%

```

Unset the regular attribute if it has been set.

```

10755 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10756   \glshasattribute{\the\glslabeltok}{regular}%
10757   {%
10758     \glissetattribute{\the\glslabeltok}{regular}{false}%
10759   }%
10760   {}%
10761 }%
10762 }%
10763 {%
10764   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10765 }

```

\glxtrshorthyphen	\glxtrshorthyphen{<short>}{<label>}{<insert>}
-------------------	---

Used by short-hyphen-postlong-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
10766 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
10767  {%
10768   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}}%
10769   \glsfirstabbrvhyphenfont{#1}%
10770  }%
10771 }
```

trposthyphenlong `\glxtrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the *<short>* part. This always uses the singular long form.

```
10772 \newcommand*{\glxtrposthyphenlong}[2]{%
10773  {%
10774   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}}%
10775   \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10776   \glxtrfullsep{#1}%
10777   \glxtrparen
10778   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
10779   \ifglxtrininsertinside\else{#2}\fi
10780  }%
10781 }%
10782 }
```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
10783 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10784 {%
10785   \renewcommand*{\CustomAbbreviationFields}{%
10786     name={\glxtrshortlongname},
10787     sort={\the\glsshorttok},
10788     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10789     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10790     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10791     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10792   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10793     \csdef{glxtrpostlink\glscategorylabel}{%
10794       \glxtrifwasfirstuse
10795       {%
10796         \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10797       }%
10798     }%
```

Put the insertion into the post-link:

```

10799      \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10800    }%
10801  }%
10802  \glshasattribute{\the\glslabeltok}{regular}%
10803  {%
10804    \glsetattribute{\the\glslabeltok}{regular}{false}%
10805  }%
10806  {}%
10807 }%
10808}%
10809{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10810 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10811 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10812 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10813 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10814 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10815 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10816   \glsabbrvfont{\glsaccessshort{##1}}%
10817 }%
10818 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10819   \glsabbrvfont{\glsaccessshortpl{##1}}%
10820 }%
10821 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10822   \glsabbrvfont{\Glsaccessshort{##1}}%
10823 }%
10824 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10825   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10826 }%

```

First use full form:

```

10827 \renewcommand*{\glxtrfullformat}[2]{%
10828   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10829 }%
10830 \renewcommand*{\glxtrfullplformat}[2]{%
10831   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10832 }%
10833 \renewcommand*{\Glsxtrfullformat}[2]{%
10834   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10835 }%
10836 \renewcommand*{\Glsxtrfullplformat}[2]{%
10837   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10838 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10839 \renewcommand*{\glxtrinlinefullformat}[2]{%

```

```

10840 \glsfirstabbrvhyphenfont{\glsaccessshort{##1}%
10841 \ifglxtrinsertinside{##2}\fi}%
10842 \ifglxtrinsertinside \else{##2}\fi
10843 }%
10844 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10845 \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}%
10846 \ifglxtrinsertinside{##2}\fi}%
10847 \ifglxtrinsertinside \else{##2}\fi
10848 }%
10849 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10850 \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
10851 \ifglxtrinsertinside{##2}\fi}%
10852 \ifglxtrinsertinside \else{##2}\fi
10853 }%
10854 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10855 \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
10856 \ifglxtrinsertinside{##2}\fi}%
10857 \ifglxtrinsertinside \else{##2}\fi
10858 }%
10859 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

10860 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10861 {%
10862 \renewcommand*{\CustomAbbreviationFields}{%
10863 name={\glxtrshortlongdescname},
10864 sort={\glxtrshortlongdescsort},%
10865 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10866 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10867 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10868 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10869 }%
10870 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10871 \csdef{glxtrpostlink\glscategorylabel}{%
10872 \glxtrifwasfirstuse
10873 {%
10874 \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10875 }%
10876 }%

```

Put the insertion into the post-link:

```

10877 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10878 }%
10879 }%
10880 \glshasattribute{\the\glslabeltok}{regular}%
10881 {%
10882 \glissetattribute{\the\glslabeltok}{regular}{false}%
10883 }%
10884 {}%
10885 }%

```

```

10886 }%
10887 {%
10888   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10889 }

```

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

```

10890 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%

```

stabbrvonlyfont

```

10891 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%

```

glslongonlyfont

```

10892 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

```

rstlongonlyfont

```

10893 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%

```

The default short form suffix:

lsxtronlysuffix

```

10894 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}

```

\glsxtronlyname The default name format for this style.

```

10895 \newcommand*{\glsxtronlyname}{%
10896   \protect\glsabbrvonlyfont{\the\glsshorttok}%
10897 }

```

only-short-only

```

10898 \newabbreviationstyle{long-only-short-only}%
10899 {%
10900   \renewcommand*{\CustomAbbreviationFields}{%
10901     name={\glsxtronlyname},
10902     sort={\the\glsshorttok},
10903     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10904     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10905     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
10906     description={\protect\glslongonlyfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10907   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10908     \glshasattribute{\the\glslabeltok}{regular}%
10909     {%
10910       \glssetattribute{\the\glslabeltok}{regular}{false}%
10911     }%
10912   }%

```

```

10913 }%
10914 }%
10915 {%
10916 \renewcommand*{\abbrvpluralsuffix}{\protect\glxstronlysuffix}%
10917 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvonlyfont{##1}}%
10918 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvonlyfont{##1}}%
10919 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongonlyfont{##1}}%
10920 \renewcommand*{\glsslongfont}[1]{\glsslongonlyfont{##1}}%

```

The first use full form doesn't show the short form.

```

10921 \renewcommand*{\glxstrfullformat}[2]{%
10922   \glssfirstlongonlyfont{\glssaccesslong{##1}\ifglxstrinsertinside##2\fi}%
10923   \ifglxstrinsertinside\else##2\fi
10924 }%
10925 \renewcommand*{\glxstrfullplformat}[2]{%
10926   \glssfirstlongonlyfont{\glssaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
10927   \ifglxstrinsertinside\else##2\fi
10928 }%
10929 \renewcommand*{\Glsxstrfullformat}[2]{%
10930   \glssfirstlongonlyfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
10931   \ifglxstrinsertinside\else##2\fi
10932 }%
10933 \renewcommand*{\Glsxstrfullplformat}[2]{%
10934   \glssfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
10935   \ifglxstrinsertinside\else##2\fi
10936 }%

```

The inline full form does show the short form.

```

10937 \renewcommand*{\glxstrinlinefullformat}[2]{%
10938   \glssfirstlongonlyfont{\glssaccesslong{##1}\ifglxstrinsertinside##2\fi}%
10939   \ifglxstrinsertinside\else##2\fi
10940   \glxstrfullsep{##1}%
10941   \glxstrparen{\protect\glssfirstabbrvonlyfont{\glssaccessshort{##1}}}%
10942 }%
10943 \renewcommand*{\glxstrinlinefullplformat}[2]{%
10944   \glssfirstlongonlyfont{\glssaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
10945   \ifglxstrinsertinside\else##2\fi
10946   \glxstrfullsep{##1}%
10947   \glxstrparen{\protect\glssfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10948 }%
10949 \renewcommand*{\Glsxstrinlinefullformat}[2]{%
10950   \glssfirstlongonlyfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
10951   \ifglxstrinsertinside\else##2\fi
10952   \glxstrfullsep{##1}%
10953   \glxstrparen{\protect\glssfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10954 }%
10955 \renewcommand*{\Glsxstrinlinefullplformat}[2]{%
10956   \glssfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
10957   \ifglxstrinsertinside\else##2\fi
10958   \glxstrfullsep{##1}%

```

```

10959     \glstrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10960   }%
10961 }

xtronlydescsort
10962 \newcommand*{\glsxtronlydescsort}{\the\glslongtok}

xtronlydescname
10963 \newcommand*{\glsxtronlydescname}{%
10964   \protect\glslongfont{\the\glslongtok}%
10965 }

short-only-desc
10966 \newabbreviationstyle{long-only-short-only-desc}%
10967 {%
10968   \renewcommand*{\CustomAbbreviationFields}{%
10969     name={\glsxtronlydescname},
10970     sort={\glsxtronlydescsort},%
10971     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10972     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10973     text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
10974     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
10975   }%
10976   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10977     \glshasattribute{\the\glslabeltok}{regular}%
10978     {%
10979       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10980     }%
10981   }%
10982 }%
10983 }%
10984 {%
10985   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10986 }

```

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 `\glxtrshort` 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:

```
10987 \let\@glxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
10988 \renewcommand*{\markright}[1]{%
10989   \glxtrmarkhook
10990   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10991   \glxtrrestoremarkhook
10992 }
```

`\markboth` Save original definition:

```
10993 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10994 \renewcommand*{\markboth}[2]{%
10995   \glxtrmarkhook
10996   \@glxtr@org@markboth
10997   {\@glxtrinmark#1\@glxtrnotinmark}%
10998   {\@glxtrinmark#2\@glxtrnotinmark}%
10999   \glxtrrestoremarkhook
11000 }
```

Also do this for `\@starttoc`

`\@starttoc` Save original definition:

```
11001 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
11002 \renewcommand*{\@starttoc}[1]{%
11003   \glxtrmarkhook
11004   \@glxtrinmark
11005   \@glxtr@org@@starttoc{#1}%
11006   \@glxtrnotinmark
11007   \glxtrrestoremarkhook
11008 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```
11009 \newcommand*{\glxstrRevertMarks}{%
11010   \let\markright\@glxstr@org@markright
11011   \let\markboth\@glxstr@org@markboth
11012   \let\@starttoc\@glxstr@org@@starttoc
11013 }
```

rRevertTocMarks Just restores \@starttoc.

```
11014 \newcommand*{\glxstrRevertTocMarks}{%
11015   \let\@starttoc\@glxstr@org@@starttoc
11016 }
```

\glxstrifinmark

```
11017 \newcommand*{\glxstrifinmark}[2]{#2}
```

\@glxstrinmark

```
11018 \newrobustcmd*{\@glxstrinmark}{%
11019   \let\glxstrifinmark\@firstoftwo
11020 }
```

glxstrnotinmark

```
11021 \newrobustcmd*{\@glxstrnotinmark}{%
11022   \let\glxstrifinmark\@secondoftwo
11023 }
```

eorpdforheading

```
11024 \ifdef\texorpdfstring
11025 {
11026   \newcommand*{\glxstrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11027 }
11028 {
11029   \newcommand*{\glxstrtitleorpdforheading}[3]{#1}
11030 }
```

\glxstrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11031 \newcommand*{\glxstrmarkhook}{%
```

Save current definitions:

```
11032   \let\@glxstr@org@MakeUppercase\MakeUppercase
11033   \let\@glxstr@org@glxstrtitleorpdforheading\glxstrtitleorpdforheading
11034   \let\@glxstr@org@glxstrtitleshort\glxstrtitleshort
11035   \let\@glxstr@org@glxstrtitleshortpl\glxstrtitleshortpl
11036   \let\@glxstr@org@Glsxstrtitleshort\Glsxstrtitleshort
11037   \let\@glxstr@org@Glsxstrtitleshortpl\Glsxstrtitleshortpl
11038   \let\@glxstr@org@glxstrtitlename\glxstrtitlename
11039   \let\@glxstr@org@Glsxstrtitlename\Glsxstrtitlename
```



```

11040 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11041 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11042 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11043 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11044 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11045 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11046 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11047 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11048 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11049 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11050 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11051 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11052 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11053 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11054 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11055 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

11056 \let\glsxtrifinmark\@firstoftwo
11057 \let\MakeUppercase\MakeTextUppercase
11058 \let\glsxtrtitleorpdforheading\@thirdofthree
11059 \let\glsxtrtitleshort\glsxtrheadshort
11060 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11061 \let\Glsxtrtitleshort\Glsxtrheadshort
11062 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11063 \let\glsxtrtitlename\glsxtrheadname
11064 \let\Glsxtrtitlename\Glsxtrheadname
11065 \let\glsxtrtitletext\glsxtrheadtext
11066 \let\Glsxtrtitletext\Glsxtrheadtext
11067 \let\glsxtrtitleplural\glsxtrheadplural
11068 \let\Glsxtrtitleplural\Glsxtrheadplural
11069 \let\glsxtrtitlefirst\glsxtrheadfirst
11070 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11071 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11072 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11073 \let\glsxtrtitlelong\glsxtrheadlong
11074 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11075 \let\Glsxtrtitlelong\Glsxtrheadlong
11076 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11077 \let\glsxtrtitlefull\glsxtrheadfull
11078 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11079 \let\Glsxtrtitlefull\Glsxtrheadfull
11080 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11081 }

```

`\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.)

```

11082 \newcommand*{\glxtrrestoremarkhook}{%
11083   \let\glxtrifinmark\@secondoftwo
11084   \let\MakeUppercase\@glxtr@org@MakeUppercase
11085   \let\glxtrtitleorpdforheading\@glxtr@org@glxtrtitleorpdforheading
11086   \let\glxtrtitleshort\@glxtr@org@glxtrtitleshort
11087   \let\glxtrtitleshortpl\@glxtr@org@glxtrtitleshortpl
11088   \let\Glsxtrtitleshort\@glxtr@org@Glsxtrtitleshort
11089   \let\Glsxtrtitleshortpl\@glxtr@org@Glsxtrtitleshortpl
11090   \let\glxtrtitlename\@glxtr@org@glxtrtitlename
11091   \let\Glsxtrtitlename\@glxtr@org@Glsxtrtitlename
11092   \let\glxtrtitletext\@glxtr@org@glxtrtitletext
11093   \let\Glsxtrtitletext\@glxtr@org@Glsxtrtitletext
11094   \let\glxtrtitleplural\@glxtr@org@glxtrtitleplural
11095   \let\Glsxtrtitleplural\@glxtr@org@Glsxtrtitleplural
11096   \let\glxtrtitlefirst\@glxtr@org@glxtrtitlefirst
11097   \let\Glsxtrtitlefirst\@glxtr@org@Glsxtrtitlefirst
11098   \let\glxtrtitlefirstplural\@glxtr@org@glxtrtitlefirstplural
11099   \let\Glsxtrtitlefirstplural\@glxtr@org@Glsxtrtitlefirstplural
11100   \let\glxtrtitlelong\@glxtr@org@glxtrtitlelong
11101   \let\glxtrtitlelongpl\@glxtr@org@glxtrtitlelongpl
11102   \let\Glsxtrtitlelong\@glxtr@org@Glsxtrtitlelong
11103   \let\Glsxtrtitlelongpl\@glxtr@org@Glsxtrtitlelongpl
11104   \let\glxtrtitlefull\@glxtr@org@glxtrtitlefull
11105   \let\glxtrtitlefullpl\@glxtr@org@glxtrtitlefullpl
11106   \let\Glsxtrtitlefull\@glxtr@org@Glsxtrtitlefull
11107   \let\Glsxtrtitlefullpl\@glxtr@org@Glsxtrtitlefullpl
11108 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glxtrheadshort Command used to display short form in the page header.

```

11109 \newcommand*{\glxtrheadshort}[1]{%
11110   \protect\NoCaseChange
11111   {%
11112     \gl@ifattribute{#1}{headuc}{true}%
11113     {%
11114       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11115     }%
11116     {%
11117       \glxtrshort[noindex,hyper=false]{#1}[]%
11118     }%
11119   }%
11120 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11121 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11122   \glxtrshort[noindex,hyper=false]{#1}[]%
11123 }

```

`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.

```
11124 \newcommand*{\glsxtrheadshortpl}[1]{%
11125   \protect\NoCaseChange
11126   {%
11127     \glusifattribute{#1}{headuc}{true}%
11128     {%
11129       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11130     }%
11131     {%
11132       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11133     }%
11134   }%
11135 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11136 \newrobustcmd*{\glsxtrtitleshortpl}[1]{%
11137   \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11138 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11139 \newcommand*{\Glsxtrheadshort}[1]{%
11140   \protect\NoCaseChange
11141   {%
11142     \glusifattribute{#1}{headuc}{true}%
11143     {%
11144       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11145     }%
11146     {%
11147       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11148     }%
11149   }%
11150 }
```

`lxsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11151 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11152   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11153 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11154 \newcommand*{\Glsxtrheadshortpl}[1]{%
11155   \protect\NoCaseChange
11156   {%
11157     \glusifattribute{#1}{headuc}{true}%
11158     {%
11159       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11160     }%
11161     {%
11162       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11163     }%
11164   }%
11165 }
```

```

11158   {%
11159     \GLSxtrshortpl [noindex,hyper=false] {#1} []%
11160   }%
11161   {%
11162     \Glsxtrshortpl [noindex,hyper=false] {#1} []%
11163   }%
11164 }%
11165 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11166 \newrobustcmd*{\Glsxtrtitleshortpl} [1] {%
11167   \Glsxtrshortpl [noindex,hyper=false] {#1} []%
11168 }

```

\glsxtrheadname As above but for the name value.

```

11169 \newcommand*{\glsxtrheadname} [1] {%
11170   \protect\NoCaseChange
11171   {%
11172     \glsifattribute{#1}{headuc}{true}%
11173     {%
11174       \GLSname [noindex,hyper=false] {#1} []%
11175     }%
11176     {%
11177       \glsname [noindex,hyper=false] {#1} []%
11178     }%
11179   }%
11180 }

```

glsxtrtitlename Command to display name value in section title and table of contents.

```

11181 \newrobustcmd*{\glsxtrtitlename} [1] {%
11182   \glsname [noindex,hyper=false] {#1} []%
11183 }

```

\Glsxtrheadname First letter converted to upper case

```

11184 \newcommand*{\Glsxtrheadname} [1] {%
11185   \protect\NoCaseChange
11186   {%
11187     \glsifattribute{#1}{headuc}{true}%
11188     {%
11189       \GLSname [noindex,hyper=false] {#1} []%
11190     }%
11191     {%
11192       \Glsname [noindex,hyper=false] {#1} []%
11193     }%
11194   }%
11195 }

```

`Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```
11196 %\changes{1.21}{2017-11-03}{new}
11197 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11198   \Glsname[noindex,hyper=false]{#1}[]%
11199 }
```

`\glsxtrheadtext` As above but for the text value.

```
11200 \newcommand*{\glsxtrheadtext}[1]{%
11201   \protect\NoCaseChange
11202   {%
11203     \glsifattribute{#1}{headuc}{true}%
11204     {%
11205       \GLStext[noindex,hyper=false]{#1}[]%
11206     }%
11207     {%
11208       \glstext[noindex,hyper=false]{#1}[]%
11209     }%
11210   }%
11211 }
```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
11212 \newrobustcmd*{\glsxtrtitletext}[1]{%
11213   \glstext[noindex,hyper=false]{#1}[]%
11214 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
11215 \newcommand*{\Glsxtrheadtext}[1]{%
11216   \protect\NoCaseChange
11217   {%
11218     \glsifattribute{#1}{headuc}{true}%
11219     {%
11220       \GLStext[noindex,hyper=false]{#1}[]%
11221     }%
11222     {%
11223       \Glstext[noindex,hyper=false]{#1}[]%
11224     }%
11225   }%
11226 }
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
11227 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11228   \Glstext[noindex,hyper=false]{#1}[]%
11229 }
```

`lsxtrheadplural` As above but for the plural value.

```
11230 \newcommand*{\lsxtrheadplural}[1]{%
```

```

11231 \protect\NoCaseChange
11232 {%
11233   \glsifattribute{#1}{headuc}{true}%
11234   {%
11235     \GLSplural[noindex,hyper=false]{#1}[]%
11236   }%
11237   {%
11238     \Glsplural[noindex,hyper=false]{#1}[]%
11239   }%
11240 }%
11241 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

11242 \newrobustcmd*{\glsxtrtitleplural}[1]{%
11243   \Glsplural[noindex,hyper=false]{#1}[]%
11244 }

```

lsxtrheadplural Convert first letter to upper case.

```

11245 \newcommand*{\Glsxtrheadplural}[1]{%
11246   \protect\NoCaseChange
11247   {%
11248     \glsifattribute{#1}{headuc}{true}%
11249     {%
11250       \GLSplural[noindex,hyper=false]{#1}[]%
11251     }%
11252     {%
11253       \Glsplural[noindex,hyper=false]{#1}[]%
11254     }%
11255   }%
11256 }

```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

11257 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11258   \Glsplural[noindex,hyper=false]{#1}[]%
11259 }

```

glsxtrheadfirst As above but for the first value.

```

11260 \newcommand*{\glsxtrheadfirst}[1]{%
11261   \protect\NoCaseChange
11262   {%
11263     \glsifattribute{#1}{headuc}{true}%
11264     {%
11265       \GLSfirst[noindex,hyper=false]{#1}[]%
11266     }%
11267     {%
11268       \glsfirst[noindex,hyper=false]{#1}[]%
11269     }%
11270   }%

```

11271 }

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
11272 \newrobustcmd*{\lsxtrtitlefirst}[1]{%
11273   \glsfirst[noindex,hyper=false]{#1}[]%
11274 }
```

Glsxtrheadfirst First letter converted to upper case

```
11275 \newcommand*{\Glsxtrheadfirst}[1]{%
11276   \protect\NoCaseChange
11277   {%
11278     \glsifattribute{#1}{headuc}{true}%
11279     {%
11280       \GLSfirst[noindex,hyper=false]{#1}[]%
11281     }%
11282     {%
11283       \Glsfirst[noindex,hyper=false]{#1}[]%
11284     }%
11285   }%
11286 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11287 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11288   \Glsfirst[noindex,hyper=false]{#1}[]%
11289 }
```

headfirstplural As above but for the firstplural value.

```
11290 \newcommand*{\glsxtrheadfirstplural}[1]{%
11291   \protect\NoCaseChange
11292   {%
11293     \glsifattribute{#1}{headuc}{true}%
11294     {%
11295       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11296     }%
11297     {%
11298       \glsfirstplural[noindex,hyper=false]{#1}[]%
11299     }%
11300   }%
11301 }
```

itlefirstplural Command to display firstplural value in section title and table of contents.

```
11302 \newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
11303   \glsfirstplural[noindex,hyper=false]{#1}[]%
11304 }
```

headfirstplural First letter converted to upper case

```
11305 \newcommand*{\Glsxtrheadfirstplural}[1]{%
```

```

11306 \protect\NoCaseChange
11307 {%
11308   \glsifattribute{#1}{headuc}{true}%
11309   {%
11310     \GLSfirstplural[noindex,hyper=false]{#1}[]%
11311   }%
11312   {%
11313     \Glsfirstplural[noindex,hyper=false]{#1}[]%
11314   }%
11315 }%
11316 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11317 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
11318   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11319 }

```

`\glxtrheadlong` Command used to display long form in the page header.

```

11320 \newcommand*{\glxtrheadlong}[1]{%
11321   \protect\NoCaseChange
11322   {%
11323     \glsifattribute{#1}{headuc}{true}%
11324     {%
11325       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11326     }%
11327     {%
11328       \glxtrlong[noindex,hyper=false]{#1}[]%
11329     }%
11330   }%
11331 }

```

`\glxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

11332 \newrobustcmd*{\glxtrtitlelong}[1]{%
11333   \glxtrlong[noindex,hyper=false]{#1}[]%
11334 }

```

`\glxtrheadlongpl` 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.

```

11335 \newcommand*{\glxtrheadlongpl}[1]{%
11336   \protect\NoCaseChange
11337   {%
11338     \glsifattribute{#1}{headuc}{true}%
11339     {%
11340       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11341     }%
11342     {%

```



```

11343     \glxstrlongpl[noindex,hyper=false]{#1}[]%
11344 }%
11345 }%
11346 }

```

xsxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents.

```

11347 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
11348   \glxstrlongpl[noindex,hyper=false]{#1}[]%
11349 }

```

\Glsxtrheadlong Command used to display long form in the page header with the first letter converted to upper case.

```

11350 \newcommand*{\Glsxtrheadlong}[1]{%
11351   \protect\NoCaseChange
11352   {%
11353     \glsifattribute{#1}{headuc}{true}%
11354     {%
11355       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11356     }%
11357     {%
11358       \Glsxtrlong[noindex,hyper=false]{#1}[]%
11359     }%
11360   }%
11361 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11362 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11363   \Glsxtrlong[noindex,hyper=false]{#1}[]%
11364 }

```

lsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11365 \newcommand*{\Glsxtrheadlongpl}[1]{%
11366   \protect\NoCaseChange
11367   {%
11368     \glsifattribute{#1}{headuc}{true}%
11369     {%
11370       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11371     }%
11372     {%
11373       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11374     }%
11375   }%
11376 }

```

xsxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11377 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11378   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11379 }

```

\glxtrheadfull Command used to display full form in the page header.

```

11380 \newcommand*{\glxtrheadfull}[1]{%
11381   \protect\NoCaseChange
11382   {%
11383     \glsifattribute{#1}{headuc}{true}%
11384     {%
11385       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11386     }%
11387     {%
11388       \glxtrfull[noindex,hyper=false]{#1}[]%
11389     }%
11390   }%
11391 }

```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```

11392 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11393   \glxtrfull[noindex,hyper=false]{#1}[]%
11394 }

```

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.

```

11395 \newcommand*{\lsxtrheadfullpl}[1]{%
11396   \protect\NoCaseChange
11397   {%
11398     \glsifattribute{#1}{headuc}{true}%
11399     {%
11400       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11401     }%
11402     {%
11403       \glxtrfullpl[noindex,hyper=false]{#1}[]%
11404     }%
11405   }%
11406 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

11407 \newrobustcmd*{\sxtrtitlefullpl}[1]{%
11408   \glxtrfullpl[noindex,hyper=false]{#1}[]%
11409 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

11410 \newcommand*{\Glsxtrheadfull}[1]{%
11411   \protect\NoCaseChange

```

```

11412 {%
11413   \glsifattribute{#1}{headuc}{true}%
11414   {%
11415     \GLSxtrfull[noindex,hyper=false]{#1}[]%
11416   }%
11417   {%
11418     \GLSxtrfull[noindex,hyper=false]{#1}[]%
11419   }%
11420 }%
11421 }

```

GLSxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11422 \newrobustcmd*{\GLSxtrtitlefull}[1]{%
11423   \GLSxtrfull[noindex,hyper=false]{#1}[]%
11424 }

```

GLSxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

11425 \newcommand*{\GLSxtrheadfullpl}[1]{%
11426   \protect\NoCaseChange
11427   {%
11428     \glsifattribute{#1}{headuc}{true}%
11429     {%
11430       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11431     }%
11432     {%
11433       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11434     }%
11435   }%
11436 }

```

GLSxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11437 \newrobustcmd*{\GLSxtrtitlefullpl}[1]{%
11438   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11439 }

```

\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.

```

11440 \ifdef\texorpdfstring
11441 {
11442   \newcommand*{\glsfmtshort}[1]{%
11443     \texorpdfstring
11444     {\glsxtrtitleshort{#1}}%
11445     {\glsentryshort{#1}}%
11446   }
11447 }

```

```

11448 {
11449   \newcommand*{\glsfmtshort}[1]{%
11450     \glstrtitleshort{#1}}
11451 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11452 \ifdef\texorpdfstring
11453 {
11454   \newcommand*{\glsfmtshortpl}[1]{%
11455     \texorpdfstring
11456       {\glstrtitleshortpl{#1}}%
11457       {\glsentryshortpl{#1}}%
11458   }
11459 }
11460 {
11461   \newcommand*{\glsfmtshortpl}[1]{%
11462     \glstrtitleshortpl{#1}}
11463 }

```

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).

```

11464 \ifdef\texorpdfstring
11465 {
11466   \newcommand*{\Glsfmtshort}[1]{%
11467     \texorpdfstring
11468       {\Glsxtrtitleshort{#1}}%
11469       {\glsentryshort{#1}}%
11470   }
11471 }
11472 {
11473   \newcommand*{\Glsfmtshort}[1]{%
11474     \Glsxtrtitleshort{#1}}
11475 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11476 \ifdef\texorpdfstring
11477 {
11478   \newcommand*{\Glsfmtshortpl}[1]{%
11479     \texorpdfstring
11480       {\Glsxtrtitleshortpl{#1}}%
11481       {\glsentryshortpl{#1}}%
11482   }
11483 }
11484 {
11485   \newcommand*{\Glsfmtshortpl}[1]{%
11486     \Glsxtrtitleshortpl{#1}}

```

11487 }

`\glsfmtname` As above but for the name value.

```
11488 \ifdef\txorpdfstring
11489 {
11490   \newcommand*\glsfmtname[1]{%
11491     \txorpdfstring
11492     {\glsxtrtitlename{#1}}%
11493     {\glsentryname{#1}}%
11494   }
11495 }
11496 {
11497   \newcommand*\glsfmtname[1]{%
11498     \glsxtrtitlename{#1}}
11499 }
```

`\Glsfmtname` First letter converted to upper case.

```
11500 \ifdef\txorpdfstring
11501 {
11502   \newcommand*\Glsfmtname[1]{%
11503     \txorpdfstring
11504     {\Glsxtrtitlename{#1}}%
11505     {\glsentryname{#1}}%
11506   }
11507 }
11508 {
11509   \newcommand*\Glsfmtname[1]{%
11510     \Glsxtrtitlename{#1}}
11511 }
```

`\glsfmttext` As above but for the text value.

```
11512 \ifdef\txorpdfstring
11513 {
11514   \newcommand*\glsfmttext[1]{%
11515     \txorpdfstring
11516     {\glsxtrtitletext{#1}}%
11517     {\glsentrytext{#1}}%
11518   }
11519 }
11520 {
11521   \newcommand*\glsfmttext[1]{%
11522     \glsxtrtitletext{#1}}
11523 }
```

`\Glsfmttext` First letter converted to upper case.

```
11524 \ifdef\txorpdfstring
11525 {
11526   \newcommand*\Glsfmttext[1]{%
11527     \txorpdfstring
```

```

11528     {\Glsxtrtitletext{#1}}%
11529     {\glsentrytext{#1}}%
11530   }
11531 }
11532 {
11533   \newcommand*{\Glsfmttext}[1]{%
11534     \Glsxtrtitletext{#1}}
11535 }

```

`\glsfmtplural` As above but for the plural value.

```

11536 \ifdef\teorpdfstring
11537 {
11538   \newcommand*{\glsfmtplural}[1]{%
11539     \teorpdfstring
11540     {\Glsxtrtitleplural{#1}}%
11541     {\glsentryplural{#1}}%
11542   }
11543 }
11544 {
11545   \newcommand*{\glsfmtplural}[1]{%
11546     \Glsxtrtitleplural{#1}}
11547 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11548 \ifdef\teorpdfstring
11549 {
11550   \newcommand*{\Glsfmtplural}[1]{%
11551     \teorpdfstring
11552     {\Glsxtrtitleplural{#1}}%
11553     {\glsentryplural{#1}}%
11554   }
11555 }
11556 {
11557   \newcommand*{\Glsfmtplural}[1]{%
11558     \Glsxtrtitleplural{#1}}
11559 }

```

`\glsfmtfirst` As above but for the first value.

```

11560 \ifdef\teorpdfstring
11561 {
11562   \newcommand*{\glsfmtfirst}[1]{%
11563     \teorpdfstring
11564     {\Glsxtrtitlefirst{#1}}%
11565     {\glsentryfirst{#1}}%
11566   }
11567 }
11568 {
11569   \newcommand*{\glsfmtfirst}[1]{%
11570     \Glsxtrtitlefirst{#1}}

```

11571 }

`\Glsfmtfirst` First letter converted to upper case.

```
11572 \ifdef\texorpdfstring
11573 {
11574   \newcommand*\Glsfmtfirst}[1]{%
11575     \texorpdfstring
11576     {\Glsxtrtitlefirst{#1}}%
11577     {\glstryfirst{#1}}%
11578   }
11579 }
11580 {
11581   \newcommand*\Glsfmtfirst}[1]{%
11582     \Glsxtrtitlefirst{#1}}
11583 }
```

`\glsfmtfirstpl` As above but for the firstplural value.

```
11584 \ifdef\texorpdfstring
11585 {
11586   \newcommand*\glsfmtfirstpl}[1]{%
11587     \texorpdfstring
11588     {\glsxtrtitlefirstplural{#1}}%
11589     {\glstryfirstplural{#1}}%
11590   }
11591 }
11592 {
11593   \newcommand*\glsfmtfirstpl}[1]{%
11594     \glsxtrtitlefirstplural{#1}}
11595 }
```

`\Glsfmtfirstpl` First letter converted to upper case.

```
11596 \ifdef\texorpdfstring
11597 {
11598   \newcommand*\Glsfmtfirstpl}[1]{%
11599     \texorpdfstring
11600     {\Glsxtrtitlefirstplural{#1}}%
11601     {\glstryfirstplural{#1}}%
11602   }
11603 }
11604 {
11605   \newcommand*\Glsfmtfirstpl}[1]{%
11606     \Glsxtrtitlefirstplural{#1}}
11607 }
```

`\glsfmtlong` As above but for the long value.

```
11608 \ifdef\texorpdfstring
11609 {
11610   \newcommand*\glsfmtlong}[1]{%
11611     \texorpdfstring
```

```

11612     {\glsxtrtitlelong{#1}}%
11613     {\glsentrylong{#1}}%
11614   }
11615 }
11616 {
11617   \newcommand*{\glsfmtlong}[1]{%
11618     \glsxtrtitlelong{#1}}
11619 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11620 \ifdef\texorpdfstring
11621 {
11622   \newcommand*{\Glsfmtlong}[1]{%
11623     \texorpdfstring
11624     {\Glsxtrtitlelong{#1}}%
11625     {\glsentrylong{#1}}%
11626   }
11627 }
11628 {
11629   \newcommand*{\Glsfmtlong}[1]{%
11630     \Glsxtrtitlelong{#1}}
11631 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11632 \ifdef\texorpdfstring
11633 {
11634   \newcommand*{\glsfmtlongpl}[1]{%
11635     \texorpdfstring
11636     {\glsxtrtitlelongpl{#1}}%
11637     {\glsentrylongpl{#1}}%
11638   }
11639 }
11640 {
11641   \newcommand*{\glsfmtlongpl}[1]{%
11642     \glsxtrtitlelongpl{#1}}
11643 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11644 \ifdef\texorpdfstring
11645 {
11646   \newcommand*{\Glsfmtlongpl}[1]{%
11647     \texorpdfstring
11648     {\Glsxtrtitlelongpl{#1}}%
11649     {\glsentrylongpl{#1}}%
11650   }
11651 }
11652 {
11653   \newcommand*{\Glsfmtlongpl}[1]{%
11654     \Glsxtrtitlelongpl{#1}}

```


11655 }

`\glsfmtfull` In-line full format.

```
11656 \ifdef\texorpdfstring
11657 {
11658   \newcommand*{\glsfmtfull}[1]{%
11659     \texorpdfstring
11660     {\glsxtrtitlefull{#1}}%
11661     {\glsxtrinlinefullformat{#1}-{}}%
11662   }
11663 }
11664 {
11665   \newcommand*{\glsfmtfull}[1]{%
11666     \glsxtrtitlefull{#1}}
11667 }
```

`\Glsfmtfull` First letter converted to upper case.

```
11668 \ifdef\texorpdfstring
11669 {
11670   \newcommand*{\Glsfmtfull}[1]{%
11671     \texorpdfstring
11672     {\Glsxtrtitlefull{#1}}%
11673     {\Glsxtrinlinefullformat{#1}-{}}%
11674   }
11675 }
11676 {
11677   \newcommand*{\Glsfmtfull}[1]{%
11678     \Glsxtrtitlefull{#1}}
11679 }
```

`\glsfmtfullpl` In-line full plural format.

```
11680 \ifdef\texorpdfstring
11681 {
11682   \newcommand*{\glsfmtfullpl}[1]{%
11683     \texorpdfstring
11684     {\glsxtrtitlefullpl{#1}}%
11685     {\glsxtrinlinefullplformat{#1}-{}}%
11686   }
11687 }
11688 {
11689   \newcommand*{\glsfmtfullpl}[1]{%
11690     \glsxtrtitlefullpl{#1}}
11691 }
```

`\Glsfmtfullpl` First letter converted to upper case.

```
11692 \ifdef\texorpdfstring
11693 {
11694   \newcommand*{\Glsfmtfullpl}[1]{%
11695     \texorpdfstring
```

```

11696     {\Glsxtrtitlefullpl{#1}}}%
11697     {\Glsxtrinlinefullplformat{#1}-{}}}%
11698   }
11699 }
11700 {
11701   \newcommand*{\Glsfmtfullpl}[1]{%
11702     \Glsxtrtitlefullpl{#1}}
11703 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11704 \newcommand*{\RequireGlossariesExtraLang}[1]{%
11705   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11706 }

```

`sariesExtraLang`

```

11707 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11708   \ProvidesFile{glossariesxtr-#1.ldf}%
11709 }

```

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.

```

11710 \newcommand{\glxtr@loaddialect}{%
11711   \IfTrackedLanguageFileExists{\this@dialect}%
11712   {glossariesxtr-}% prefix
11713   {.ldf}%
11714   {%
11715     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11716   }%
11717   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11718   \@glxtrdialecthook
11719 }

11720 \@ifpackageloaded{tracklang}
11721 {%
11722   \AnyTrackedLanguages
11723   {%

```

```

11724 \ForEachTrackedDialect{\this@dialect}{\glstr@loaddialect}%
11725 }%
11726 {}%
11727 }
11728 {}

```

Load glossaries-extra-stylemods if required.

```
11729 \@glstr@redefstyles
```

and set the style:

```
11730 \@glstr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for bib2gls and is automatically loaded by the record option.

```
11731 \NeedsTeXFormat{LaTeX2e}
```

```
11732 \ProvidesPackage{glossaries-extra-bib2gls}[2018/08/18 v1.36 (NLCT)]
```

These are some convenient macros for use with custom rules.

`\glshex`

```
11733 \newcommand*{\glshex}{\string\u}
```

`\glscapturedgroup`

```
11734 \newcommand*{\glscapturedgroup}{\string\$}
```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```
11735 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
```

```
11736 \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
```

```
11737 }
```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```
11738 \newcommand*{\GlsXtrProvideCommand}{\providecommand}
```

`\GlsXtrGlossaryLocation` For use with indexcounter and bib2gls.

```
11739 \newcommand*{\GlsXtr@wrglossarylocation}[2]{#1}
```

`\GlsXtrIndexCounterLink` `\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```
11740 \ifdef\hyperref
```

```
11741 {%
```

```
11742 \newcommand*{\GlsXtrIndexCounterLink}[2]{%
```

```

11743 \glstrifhasfield{indexcounter}{#2}%
11744 {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11745 {#1}%
11746 }
11747 }
11748 {
11749 \newcommand*\GlsXtrIndexCounterLink}[2]{#1}
11750 }

```

`\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.

```

11751 \newcommand*\GlsXtrDualField}{dual}

```

`\GlsXtrDualBackLink` `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```

11752 \newcommand*\GlsXtrDualBackLink}[2]{%
11753 \glstrifhasfield{\GlsXtrDualField}{#2}%
11754 {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11755 {#2}%
11756 }

```

`\TeXEntryAliases` Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```

11757 \newcommand*\GlsXtrBibTeXEntryAliases}{%
11758 article=bibtexentry,
11759 book=bibtexentry,
11760 booklet=bibtexentry,
11761 conference=bibtexentry,
11762 inbook=bibtexentry,
11763 incollection=bibtexentry,
11764 inproceedings=bibtexentry,
11765 manual=bibtexentry,
11766 mastersthesis=bibtexentry,
11767 misc=bibtexentry,
11768 phdthesis=bibtexentry,
11769 proceedings=bibtexentry,
11770 techreport=bibtexentry,
11771 unpublished=bibtexentry
11772 }

```

ideBibTeXFields Convenient shortcut to define the standard BibTeX fields.

```
11773 \newcommand*\GlsXtrProvideBibTeXFields}{%
11774   \glsaddstoragekey{address}{}{\glsxtrbibaddress}%
11775   \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
11776   \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}%
11777   \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
11778   \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
11779   \glsaddstoragekey{howpublished}{}{\glsxtrbibhowpublished}%
11780   \glsaddstoragekey{institution}{}{\glsxtrbibinstitution}%
11781   \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
11782   \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
11783   \glsaddstoragekey{note}{}{\glsxtrbibnote}%
11784   \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
11785   \glsaddstoragekey{organization}{}{\glsxtrbiborganization}%
11786   \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
11787   \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}%
11788   \glsaddstoragekey{school}{}{\glsxtrbibschooll}%
11789   \glsaddstoragekey{series}{}{\glsxtrbibseries}%
11790   \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
11791   \glsaddstoragekey{bibtextype}{}{\glsxtrbibtype}%
11792   \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
11793 }
```

Multiple supplementary references are only supported with bib2gls.

ltisuppllocation 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.

```
11794 \newcommand*\glsxtrmultisuppllocation}[3]{%
11795   {%
11796     \def\glsxtrsuppllocationurl{#2}%
11797     \glsxhypernumber{#1}%
11798   }%
11799 }
```

trdisplaysupplloc `\glsxtrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like \glsnoidxdisplayloc but is used for supplementary locations and so requires an extra argument.

```
11800 \newcommand*\glsxtrdisplaysupplloc[5]{%
11801   \setentrycounter{#1}{#2}%
11802   \glsxtrmultisuppllocation{#5}{#4}{#3}%
11803 }
```

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.

\backslash Alpha

```
11804 \providecommand*\Alpha{\mathrm{A}}
```

\backslash Beta

```
11805 \providecommand*\Beta{\mathrm{B}}
```

\backslash Epsilon

```
11806 \providecommand*\Epsilon{\mathrm{E}}
```

\backslash Zeta

```
11807 \providecommand*\Zeta{\mathrm{Z}}
```

\backslash Eta

```
11808 \providecommand*\Eta{\mathrm{H}}
```

\backslash Iota

```
11809 \providecommand*\Iota{\mathrm{I}}
```

\backslash Kappa

```
11810 \providecommand*\Kappa{\mathrm{K}}
```

\backslash Mu

```
11811 \providecommand*\Mu{\mathrm{M}}
```

\backslash Nu

```
11812 \providecommand*\Nu{\mathrm{N}}
```

\backslash Omicron

```
11813 \providecommand*\Omicron{\mathrm{O}}
```

\backslash Rho

```
11814 \providecommand*\Rho{\mathrm{P}}
```

\backslash Tau

```
11815 \providecommand*\Tau{\mathrm{T}}
```

\backslash Chi

```
11816 \providecommand*\Chi{\mathrm{X}}
```

\backslash Digamma

```
11817 \providecommand*\Digamma{\mathrm{F}}
```

```

\omicron
11818 \providecommand*\{\omicron}\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11819 \@ifpackageloaded{upgreek}%
11820 {

\Upalpha
11821 \providecommand*\{\Upalpha}\mathrm{A}}

\Upbeta
11822 \providecommand*\{\Upbeta}\mathrm{B}}

\Upsilon
11823 \providecommand*\{\Upsilon}\mathrm{E}}

\Upzeta
11824 \providecommand*\{\Upzeta}\mathrm{Z}}

\Upeta
11825 \providecommand*\{\Upeta}\mathrm{H}}

\Upiota
11826 \providecommand*\{\Upiota}\mathrm{I}}

\Upkappa
11827 \providecommand*\{\Upkappa}\mathrm{K}}

\Upmu
11828 \providecommand*\{\Upmu}\mathrm{M}}

\Upnu
11829 \providecommand*\{\Upnu}\mathrm{N}}

\Upomicron
11830 \providecommand*\{\Upomicron}\mathrm{O}}

\Uprho
11831 \providecommand*\{\Uprho}\mathrm{P}}

\Uptau
11832 \providecommand*\{\Uptau}\mathrm{T}}

\Upchi
11833 \providecommand*\{\Upchi}\mathrm{X}}

```

`\upomicron`

```
11834 \providecommand*\upomicron{\mathrm{o}}
11835 }%
11836 {}% 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.

```
11837 \newcommand*\glxtrcontrolrules{%
11838 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
11839 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
11840 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
11841 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
11842 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
11843 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
11844 0010\string'\string=\glshex 0011
11845 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11846 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11847 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11848 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11849 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
```



```

11850 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11851 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11852 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11853 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11854 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11855 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11856 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11857 }

```

lsxtrspacerules These are space characters.

```

11858 \newcommand*{\glsxtrspacerules}{%
11859 \string' \string'\string;
11860 \string'\glshex 00A0\string'\string;
11861 \string'\glshex 2000\string'\string;
11862 \string'\glshex 2001\string'\string;
11863 \string'\glshex 2002\string'\string;
11864 \string'\glshex 2003\string'\string;
11865 \string'\glshex 2004\string'\string;
11866 \string'\glshex 2005\string'\string;
11867 \string'\glshex 2006\string'\string;
11868 \string'\glshex 2007\string'\string;
11869 \string'\glshex 2008\string'\string;
11870 \string'\glshex 2009\string'\string;
11871 \string'\glshex 200A\string'\string;
11872 \string'\glshex 3000\string'
11873 }

```

nonprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11874 \newcommand*{\glsxtrnonprintablerules}{%
11875 \string'\glshex FEFF\string'\string;
11876 \string'\glshex 000A\string'\string;
11877 \string'\glshex 0009\string'\string;
11878 \string'\glshex 000C\string'\string;
11879 \string'\glshex 000B\string'
11880 }

```

diacriticrules Combining diacritic marks. This is split into multiple macros.

```

11881 \newcommand*{\glsxtrcombiningdiacriticrules}{%
11882 \glsxtrcombiningdiacriticIrules\string;
11883 \glsxtrcombiningdiacriticIIrules\string;
11884 \glsxtrcombiningdiacriticIIIrules\string;
11885 \glsxtrcombiningdiacriticIVrules
11886 }

```

diacriticIrules First set of combining diacritic marks.

```

11887 \newcommand*{\glsxtrcombiningdiacriticIrules}{%
11888 \glshex 0301\string;% combining acute
11889 \glshex 0300\string;% combining grave
11890 \glshex 0306\string;% combining breve

```

```

11891 \glshex 0302\string;% combining circumflex
11892 \glshex 030C\string;% combining caron
11893 \glshex 030A\string;% combining ring
11894 \glshex 030D\string;% combining vertical line above
11895 \glshex 0308\string;% combining diaeresis
11896 \glshex 030B\string;% combining double acute
11897 \glshex 0303\string;% combining tilde
11898 \glshex 0307\string;% combining dot above
11899 \glshex 0304% combining macron
11900 }

```

iacriticIIrules Second set of combining diacritic marks.

```

11901 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
11902 \glshex 0337\string;% combining short solidus overlay
11903 \glshex 0327\string;% combining cedilla
11904 \glshex 0328\string;% combining ogonek
11905 \glshex 0323\string;% combining dot below
11906 \glshex 0332\string;% combining low line
11907 \glshex 0305\string;% combining overline
11908 \glshex 0309\string;% combining hook above
11909 \glshex 030E\string;% combining double vertical line above
11910 \glshex 030F\string;% combining double grave accent
11911 \glshex 0310\string;% combining candrabindu
11912 \glshex 0311\string;% combining inverted breve
11913 \glshex 0312\string;% combining turned comma above
11914 \glshex 0313\string;% combining comma above
11915 \glshex 0314\string;% combining reversed comma above
11916 \glshex 0315\string;% combining comma above right
11917 \glshex 0316\string;% combining grave accent below
11918 \glshex 0317% combining acute accent below
11919 }

```

acriticIIIrules Third set of combining diacritic marks.

```

11920 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
11921 \glshex 0318\string;% combining left tack below
11922 \glshex 0319\string;% combining right tack below
11923 \glshex 031A\string;% combining left angle above
11924 \glshex 031B\string;% combining horn
11925 \glshex 031C\string;% combining left half ring below
11926 \glshex 031D\string;% combining up tack below
11927 \glshex 031E\string;% combining down tack below
11928 \glshex 031F\string;% combining plus sign below
11929 \glshex 0320\string;% combining minus sign below
11930 \glshex 0321\string;% combining palatalized hook below
11931 \glshex 0322\string;% combining retroflex hook below
11932 \glshex 0324\string;% combining diaeresis below
11933 \glshex 0325\string;% combining ring below
11934 \glshex 0326\string;% combining comma below
11935 \glshex 0329\string;% combining vertical line below

```

11936 \glshex 032A\string;% combining bridge below
 11937 \glshex 032B\string;% combining inverted double arch below
 11938 \glshex 032C\string;% combining caron below
 11939 \glshex 032D\string;% combining circumflex accent below
 11940 \glshex 032E\string;% combining breve below
 11941 \glshex 032F\string;% combining inverted breve below
 11942 \glshex 0330\string;% combining tilde below
 11943 \glshex 0331\string;% combining macron below
 11944 \glshex 0333\string;% combining double low line
 11945 \glshex 0334\string;% combining tilde overlay
 11946 \glshex 0335\string;% combining short stroke overlay
 11947 \glshex 0336\string;% combining long stroke overlay
 11948 \glshex 0338\string;% combining long solidus overlay
 11949 \glshex 0339\string;% combining combining right half ring below
 11950 \glshex 033A\string;% combining inverted bridge below
 11951 \glshex 033B\string;% combining square below
 11952 \glshex 033C\string;% combining seagull below
 11953 \glshex 033D\string;% combining x above
 11954 \glshex 033E\string;% combining vertical tilde
 11955 \glshex 033F\string;% combining double overline
 11956 \glshex 0342\string;% combining Greek perispomeni
 11957 \glshex 0344\string;% combining Greek dialytika tonos
 11958 \glshex 0345\string;% combining Greek ypogegrammeni
 11959 \glshex 0360\string;% combining double tilde
 11960 \glshex 0361\string;% combining double inverted breve
 11961 \glshex 0483\string;% combining Cyrillic titlo
 11962 \glshex 0484\string;% combining Cyrillic palatalization
 11963 \glshex 0485\string;% combining Cyrillic dasia pneumata
 11964 \glshex 0486% combining Cyrillic psili pneumata
 11965 }

iacriticIVrules Fourth set of combining diacritic marks.

11966 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
 11967 \glshex 20D0\string;% combining left harpoon above
 11968 \glshex 20D1\string;% combining right harpoon above
 11969 \glshex 20D2\string;% combining long vertical line overlay
 11970 \glshex 20D3\string;% combining short vertical line overlay
 11971 \glshex 20D4\string;% combining anticlockwise arrow above
 11972 \glshex 20D5\string;% combining clockwise arrow above
 11973 \glshex 20D6\string;% combining left arrow above
 11974 \glshex 20D7\string;% combining right arrow above
 11975 \glshex 20D8\string;% combining ring overlay
 11976 \glshex 20D9\string;% combining clockwise ring overlay
 11977 \glshex 20DA\string;% combining anticlockwise ring overlay
 11978 \glshex 20DB\string;% combining three dots above
 11979 \glshex 20DC\string;% combining four dots above
 11980 \glshex 20DD\string;% combining enclosing circle
 11981 \glshex 20DE\string;% combining enclosing square
 11982 \glshex 20DF\string;% combining enclosing diamond

```

11983 \glsheX 20E0\string;% combining enclosing circle backslash
11984 \glsheX 20E1% combining left right arrow above
11985 }

```

sxtrhyphenrules Hyphens.

```

11986 \newcommand*{\glsxtrhyphenrules}{%
11987 \string'\string-\string'\string;% ASCII hyphen
11988 \glsheX 00AD\string;% soft hyphen
11989 \glsheX 2010\string;% hyphen
11990 \glsheX 2011\string;% non-breaking hyphen
11991 \glsheX 2012\string;% figure dash
11992 \glsheX 2013\string;% en dash
11993 \glsheX 2014\string;% em dash
11994 \glsheX 2015\string;% horizontal bar
11995 \glsheX 2212\string=\glsheX 207B\string=\glsheX 208B% minus sign
11996 }

```

eneralpuncrules General punctuation.

```

11997 \newcommand*{\glsxtrgeneralpuncrules}{%
11998 \glsxtrgeneralpuncIrules
11999 \string<\glsxtrcurrencyrules
12000 \string<\glsxtrgeneralpuncIIrules
12001 }

```

eneralpuncIrules First set of general punctuation.

```

12002 \newcommand*{\glsxtrgeneralpuncIrules}{%
12003 \string'\glsheX 005F\string'% underscore
12004 \string<\glsheX 00AF% macron
12005 \string<\string'\glsheX 002C\string'% comma
12006 \string<\string'\glsheX 003B\string'% semi-colon
12007 \string<\string'\glsheX 003A\string'% colon
12008 \string<\string'\glsheX 0021\string'% exclamation mark
12009 \string<\glsheX 00A1% inverted exclamation mark
12010 \string<\string'\glsheX 003F\string'% question mark
12011 \string<\glsheX 00BF% inverted question mark
12012 \string<\string'\glsheX 002F\string'% solidus
12013 \string<\string'\glsheX 002E\string'% full stop
12014 \string<\glsheX 00B4% acute accent
12015 \string<\string'\glsheX 0060\string'% grave accent
12016 \string<\string'\glsheX 005E\string'% circumflex accent
12017 \string<\glsheX 00A8% diaeresis
12018 \string<\string'\glsheX 007E\string'% tilde
12019 \string<\glsheX 00B7% middle dot
12020 \string<\glsheX 00B8% cedilla
12021 \string<\string'\glsheX 0027\string'% straight apostrophe
12022 \string<\string'\glsheX 0022\string'% straight double quote
12023 \string<\glsheX 00AB% left guillemet
12024 \string<\glsheX 00BB% right guillemet
12025 \string<\string'\glsheX 0028\string'% left parenthesis

```

```

12026 \string=\glshex 207D\string=\glshex 208D% super/subscript left parenthesis
12027 \string<\string'\glshex 0029\string'% right parenthesis
12028 \string=\glshex 207E\string=\glshex 208E% super/subscript right parenthesis
12029 \string<\string'\glshex 005B\string'% left square bracket
12030 \string<\string'\glshex 005D\string'% right square bracket
12031 \string<\string'\glshex 007B\string'% left curly bracket
12032 \string<\string'\glshex 007D\string'% right curly bracket
12033 \string<\glshex 00A7% section sign
12034 \string<\glshex 00B6% pilcrow sign
12035 \string<\glshex 00A9% copyright sign
12036 \string<\glshex 00AE% registered sign
12037 \string<\string'\glshex 0040\string'% at sign
12038 }

```

trcurrencyrules General punctuation.

```

12039 \newcommand*{\glxtrcurrencyrules}{%
12040 \glshex 00A4% currency sign
12041 \string<\glshex 0E3F% Thai currency symbol baht
12042 \string<\glshex 00A2% cent sign
12043 \string<\glshex 20A1% colon sign
12044 \string<\glshex 20A2% cruzeiro sign
12045 \string<\string'\glshex 0024\string'% dollar sign
12046 \string<\glshex 20AB% dong sign
12047 \string<\glshex 20AC% euro sign
12048 \string<\glshex 20A3% French franc sign
12049 \string<\glshex 20A4% lira sign
12050 \string<\glshex 20A5% mill sign
12051 \string<\glshex 20A6% naira sign
12052 \string<\glshex 20A7% peseta sign
12053 \string<\glshex 00A3% pound sign
12054 \string<\glshex 20A8% rupee sign
12055 \string<\glshex 20AA% new sheqel sign
12056 \string<\glshex 20A9% won sign
12057 \string<\glshex 00A5% yen sign
12058 }

```

eralpuncIIrules Second set of general punctuation.

```

12059 \newcommand*{\glxtrgeneralpuncIIrules}{%
12060 \string'\glshex 002A\string'% asterisk
12061 \string<\string'\glshex 005C\string'% backslash
12062 \string<\string'\glshex 0026\string'% ampersand
12063 \string<\string'\glshex 0023\string'% hash sign
12064 \string<\string'\glshex 0025\string'% percent sign
12065 \string<\string'\glshex 002B\string'% plus sign
12066 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12067 \string<\glshex 00B1% plus-minus sign
12068 \string<\glshex 00F7% division sign
12069 \string<\glshex 00D7% multiplication sign
12070 \string<\string'\glshex 003C\string'% less-than sign

```

```

12071 \string<\string'\glshex 003D\string'% equals sign
12072 \string<\string'\glshex 003E\string'% greater-than sign
12073 \string<\glshex 00AC% not sign
12074 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12075 \string<\glshex 00A6% broken bar
12076 \string<\glshex 00B0% degree sign
12077 \string<\glshex 00B5% micron sign
12078 }

```

GeneralLatinIrules Basic Latin alphabet.

```

12079 \newcommand*{\glxtrGeneralLatinIrules}{%
12080 \glxtrLatinA
12081 \string<b,B%
12082 \string<c,C%
12083 \string<d,D%
12084 \string<\glxtrLatinE
12085 \string<f,F%
12086 \string<g,G%
12087 \string<\glxtrLatinH
12088 \string<\glxtrLatinI
12089 \string<j,J%
12090 \string<\glxtrLatinK
12091 \string<\glxtrLatinL
12092 \string<\glxtrLatinM
12093 \string<\glxtrLatinN
12094 \string<\glxtrLatinO
12095 \string<\glxtrLatinP
12096 \string<q,Q%
12097 \string<r,R%
12098 \string<\glxtrLatinS
12099 \string<\glxtrLatinT
12100 \string<u,U%
12101 \string<v,V%
12102 \string<w,W%
12103 \string<\glxtrLatinX
12104 \string<y,Y%
12105 \string<z,Z
12106 }

```

GeneralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12107 \newcommand*{\glxtrGeneralLatinIIrules}{%
12108 \glxtrLatinA
12109 \string<b,B%
12110 \string<c,C%
12111 \string<d,D%
12112 \string<\glxtrLatinEth
12113 \string<\glxtrLatinE
12114 \string<f,F%
12115 \string<g,G%

```

```

12116 \string<\glxtrLatinH
12117 \string<\glxtrLatinI
12118 \string<j,J%
12119 \string<\glxtrLatinK
12120 \string<\glxtrLatinL
12121 \string<\glxtrLatinM
12122 \string<\glxtrLatinN
12123 \string<\glxtrLatinO
12124 \string<\glxtrLatinP
12125 \string<q,Q%
12126 \string<r,R%
12127 \string<\glxtrLatinS
12128 \string& SS \string, \glxtrLatinEszettSs
12129 \string<\glxtrLatinT
12130 \string<u,U%
12131 \string<v,V%
12132 \string<w,W%
12133 \string<\glxtrLatinX
12134 \string<y,Y%
12135 \string<z,Z%
12136 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12137 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12138 \glxtrLatinA
12139 \string<b,B%
12140 \string<c,C%
12141 \string<d,D%
12142 \string<\glxtrLatinEth
12143 \string<\glxtrLatinE
12144 \string<f,F%
12145 \string<g,G%
12146 \string<\glxtrLatinH
12147 \string<\glxtrLatinI
12148 \string<j,J%
12149 \string<\glxtrLatinK
12150 \string<\glxtrLatinL
12151 \string<\glxtrLatinM
12152 \string<\glxtrLatinN
12153 \string<\glxtrLatinO
12154 \string<\glxtrLatinP
12155 \string<q,Q%
12156 \string<r,R%
12157 \string<\glxtrLatinS
12158 \string& SZ, \glxtrLatinEszettSz
12159 \string<\glxtrLatinT
12160 \string<u,U%
12161 \string<v,V%
12162 \string<w,W%

```

```

12163 \string<\glxtrLatinX
12164 \string<y,Y%
12165 \string<z,Z%
12166 }

```

GeneralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12167 \newcommand*{\glxtrGeneralLatinIVrules}{%
12168 \glxtrLatinA
12169 \string& AE , \glxtrLatinAELigature
12170 \string<b,B%
12171 \string<c,C%
12172 \string<d,D%
12173 \string<\glxtrLatinEth
12174 \string<\glxtrLatinE
12175 \string<f,F%
12176 \string<g,G%
12177 \string<\glxtrLatinH
12178 \string<\glxtrLatinI
12179 \string<j,J%
12180 \string<\glxtrLatinK
12181 \string<\glxtrLatinL
12182 \string<\glxtrLatinM
12183 \string<\glxtrLatinN
12184 \string<\glxtrLatinO
12185 \string& OE , \glxtrLatinOELigature
12186 \string<\glxtrLatinP
12187 \string<q,Q%
12188 \string<r,R%
12189 \string<\glxtrLatinS
12190 \string& SS , \glxtrLatinEszettSs
12191 \string<\glxtrLatinT
12192 \string& th =\glshex 00DE
12193 \string& TH =\glshex 00FE
12194 \string<u,U%
12195 \string<v,V%
12196 \string<w,W%
12197 \string<\glxtrLatinX
12198 \string<y,Y%
12199 \string<z,Z%
12200 }

```

GeneralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12201 \newcommand*{\glxtrGeneralLatinVrules}{%
12202 \glxtrLatinA
12203 \string<b,B%
12204 \string<c,C%
12205 \string<d,D%
12206 \string<\glxtrLatinEth

```



```

12207 \string<\glxtrLatinE
12208 \string<f,F%
12209 \string<g,G%
12210 \string<\glxtrLatinH
12211 \string<\glxtrLatinI
12212 \string<j,J%
12213 \string<\glxtrLatinK
12214 \string<\glxtrLatinL
12215 \string<\glxtrLatinM
12216 \string<\glxtrLatinN
12217 \string<\glxtrLatinO
12218 \string<\glxtrLatinP
12219 \string<q,Q%
12220 \string<r,R%
12221 \string<\glxtrLatinS
12222 \string& SS , \glxtrLatinEszettSs
12223 \string<\glxtrLatinT
12224 \string& th =\glshex 00DE
12225 \string& TH =\glshex 00FE
12226 \string<u,U%
12227 \string<v,V%
12228 \string<w,W%
12229 \string<\glxtrLatinX
12230 \string<y,Y%
12231 \string<z,Z%
12232 }

```

raalLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12233 \newcommand*{\glxtrGeneralLatinVIrules}{%
12234 \glxtrLatinA
12235 \string<b,B%
12236 \string<c,C%
12237 \string<d,D%
12238 \string<\glxtrLatinEth
12239 \string<\glxtrLatinE
12240 \string<f,F%
12241 \string<g,G%
12242 \string<\glxtrLatinH
12243 \string<\glxtrLatinI
12244 \string<j,J%
12245 \string<\glxtrLatinK
12246 \string<\glxtrLatinL
12247 \string<\glxtrLatinM
12248 \string<\glxtrLatinN
12249 \string<\glxtrLatinO
12250 \string<\glxtrLatinP
12251 \string<q,Q%
12252 \string<r,R%
12253 \string<\glxtrLatinS

```

```

12254 \string& SZ , \glxtrLatinEszettSz
12255 \string<\glxtrLatinT
12256 \string& th =\glshex 00DE
12257 \string& TH =\glshex 00FE
12258 \string<u,U%
12259 \string<v,V%
12260 \string<w,W%
12261 \string<\glxtrLatinX
12262 \string<y,Y%
12263 \string<z,Z%
12264 }

```

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).

```

12265 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12266 \glxtrLatinA
12267 \string<\glxtrLatinAELigature
12268 \string<b,B%
12269 \string<c,C%
12270 \string<d,D%
12271 \string<\glxtrLatinEth
12272 \string<\glxtrLatinE
12273 \string<f,F%
12274 \string<\glxtrLatinInsularG
12275 \string<\glxtrLatinH
12276 \string<\glxtrLatinI
12277 \string<j,J%
12278 \string<\glxtrLatinK
12279 \string<\glxtrLatinL
12280 \string<\glxtrLatinM
12281 \string<\glxtrLatinN
12282 \string<\glxtrLatinO
12283 \string<\glxtrLatinOELigature
12284 \string<\glxtrLatinP
12285 \string<q,Q%
12286 \string<r,R%
12287 \string<\glshex 017F=\glxtrLatinS % s and long s
12288 \string<\glxtrLatinT
12289 \string<\glxtrLatinThorn
12290 \string<u,U%
12291 \string<v,V%
12292 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12293 \string<\glxtrLatinX
12294 \string<y,Y%
12295 \string<z,Z%
12296 }

```

allLatinVIIIrules General Latin alphabet (Æ treated as AE and Œ treated as OE, Þ treated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

12297 \newcommand*{\glxtrGeneralLatinVIIIRules}{%
12298   \glxtrLatinA
12299   \string& AE , \glxtrLatinAELigature
12300   \string<b,B%
12301   \string<c,C%
12302   \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12303   \string<\glxtrLatinE
12304   \string<f,F%
12305   \string<g,G%
12306   \string<\glxtrLatinH
12307   \string<\glxtrLatinI
12308   \string<j,J%
12309   \string<\glxtrLatinK
12310   \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12311   \string<\glxtrLatinM
12312   \string<\glxtrLatinN
12313   \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12314   \string& OE , \glxtrLatinOELigature
12315   \string<\glxtrLatinP
12316   \string<q,Q%
12317   \string<r,R%
12318   \string<\glxtrLatinS
12319   \string& SS , \glxtrLatinEszettSs
12320   \string<\glxtrLatinT
12321   \string& th =\glshex 00DE
12322   \string& TH =\glshex 00FE
12323   \string<u,U%
12324   \string<v,V%
12325   \string<w,W%
12326   \string<\glxtrLatinX
12327   \string<y,Y%
12328   \string<z,Z%
12329 }

```

\glxtrLatinA

```

12330 \newcommand*{\glxtrLatinA}{%
12331   a\string=\glshex 00AA\string=\glshex 2090,A
12332 }

```

\glxtrLatinE

```

12333 \newcommand*{\glxtrLatinE}{%
12334   e\string=\glshex 2091,E
12335 }

```

\glxtrLatinH

```

12336 \newcommand*{\glxtrLatinH}{%
12337   h\string=\glshex 2095,H
12338 }

```

```

\glxtrLatinI
12339 \newcommand*{\glxtrLatinI}{%
12340   i\string=\glshex 2071,I
12341 }

\glxtrLatinK
12342 \newcommand*{\glxtrLatinK}{%
12343   k\string=\glshex 2096,K
12344 }

\glxtrLatinL
12345 \newcommand*{\glxtrLatinL}{%
12346   l\string=\glshex 2097,L
12347 }

\glxtrLatinM
12348 \newcommand*{\glxtrLatinM}{%
12349   m\string=\glshex 2098,M
12350 }

\glxtrLatinN
12351 \newcommand*{\glxtrLatinN}{%
12352   n\string=\glshex 207F\string=\glshex 2099,N
12353 }

\glxtrLatinO
12354 \newcommand*{\glxtrLatinO}{%
12355   o\string=\glshex 00BA\string=\glshex 2092,O
12356 }

\glxtrLatinP
12357 \newcommand*{\glxtrLatinP}{%
12358   p\string=\glshex 209A,P
12359 }

\glxtrLatinS
12360 \newcommand*{\glxtrLatinS}{%
12361   s\string=\glshex 209B,S
12362 }

\glxtrLatinT
12363 \newcommand*{\glxtrLatinT}{%
12364   t\string=\glshex 209C,T
12365 }

\glxtrLatinX
12366 \newcommand*{\glxtrLatinX}{%
12367   x\string=\glshex 2093,X
12368 }

```

lsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).

```
12369 \newcommand*{\glxtrLatinSchwa}{%
12370   \glshex 0259\string=\glshex 2094,\glshex 018F
12371 }
```

trLatinEszettSs

```
12372 \newcommand*{\glxtrLatinEszettSs}{%
12373   \glshex 00DF% eszett
12374   \string=\glshex 017Fs % long S s
12375 }
```

trLatinEszettSz

```
12376 \newcommand*{\glxtrLatinEszettSz}{%
12377   \glshex 00DF% eszett
12378   \string= \glshex 017Fz % long S z
12379 }
```

\glxtrLatinEth

```
12380 \newcommand*{\glxtrLatinEth}{%
12381   \glshex 00F0,\glshex 00D0% eth
12382 }
```

lsxtrLatinThorn

```
12383 \newcommand*{\glxtrLatinThorn}{%
12384   \glshex 00FE,\glshex 00DE% thorn
12385 }
```

LatinAELigature

```
12386 \newcommand*{\glxtrLatinAELigature}{%
12387   \glshex 00E6,\glshex 00C6% AE-ligature
12388 }
```

LatinOELigature

```
12389 \newcommand*{\glxtrLatinOELigature}{%
12390   \glshex 0153,\glshex 0152% OE-ligature
12391 }
```

\glxtrLatinAA

```
12392 \newcommand*{\glxtrLatinAA}{%
12393   \glshex 00E5=a\glshex 030A,% \aa
12394   \glshex 00C5=A\glshex 030A% \AA
12395 }
```

glxtrLatinWynn

```
12396 \newcommand*{\glxtrLatinWynn}{%
12397   \glshex 01BF,\glshex 01F7% wynn
12398 }
```

trLatinInsularG

```
12399 \newcommand*{\glxtrLatinInsularG}{%
12400 \glshex 1D79,\glshex A77D% insular G
12401 \string; g, G
12402 }
```

sxtrLatinOslash

```
12403 \newcommand*{\glxtrLatinOslash}{%
12404 \glshex 00F8,\glshex 00D8% \o, \O
12405 }
```

sxtrLatinLslash

```
12406 \newcommand*{\glxtrLatinLslash}{%
12407 \glshex 0142,\glshex 0141% \l, \L
12408 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12409 \newcommand*{\glxtrMathUpGreekIrules}{%
12410 \glxtrUpAlpha
12411 \string<\glxtrUpBeta
12412 \string<\glxtrUpGamma
12413 \string<\glxtrUpDelta
12414 \string<\glxtrUpEpsilon
12415 \string<\glxtrUpDigamma
12416 \string<\glxtrUpZeta
12417 \string<\glxtrUpEta
12418 \string<\glxtrUpTheta
12419 \string<\glxtrUpIota
12420 \string<\glxtrUpKappa
12421 \string<\glxtrUpLambda
12422 \string<\glxtrUpMu
12423 \string<\glxtrUpNu
12424 \string<\glxtrUpXi
12425 \string<\glxtrUpOmicron
12426 \string<\glxtrUpPi
12427 \string<\glxtrUpRho
12428 \string<\glxtrUpSigma
12429 \string<\glxtrUpTau
12430 \string<\glxtrUpUpsilon
12431 \string<\glxtrUpPhi
12432 \string<\glxtrUpChi
12433 \string<\glxtrUpPsi
12434 \string<\glxtrUpOmega
12435 }
```

hUpGreekIIrules Doesn't include digamma.

```
12436 \newcommand*{\glxtrMathUpGreekIIrules}{%
12437 \glxtrUpAlpha
12438 \string<\glxtrUpBeta
```

```

12439 \string<\glxtrUpGamma
12440 \string<\glxtrUpDelta
12441 \string<\glxtrUpEpsilon
12442 \string<\glxtrUpZeta
12443 \string<\glxtrUpEta
12444 \string<\glxtrUpTheta
12445 \string<\glxtrUpIota
12446 \string<\glxtrUpKappa
12447 \string<\glxtrUpLambda
12448 \string<\glxtrUpMu
12449 \string<\glxtrUpNu
12450 \string<\glxtrUpXi
12451 \string<\glxtrUpOmicron
12452 \string<\glxtrUpPi
12453 \string<\glxtrUpRho
12454 \string<\glxtrUpSigma
12455 \string<\glxtrUpTau
12456 \string<\glxtrUpUpsilon
12457 \string<\glxtrUpPhi
12458 \string<\glxtrUpChi
12459 \string<\glxtrUpPsi
12460 \string<\glxtrUpOmega
12461 }

```

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.

```

12462 \newcommand*{\glxtrMathItalicGreekIrules}{%
12463 \glxtrMathItalicAlpha
12464 \string<\glxtrMathItalicBeta
12465 \string<\glxtrMathItalicGamma
12466 \string<\glxtrMathItalicDelta
12467 \string<\glxtrMathItalicEpsilon
12468 \string<\glxtrUpDigamma
12469 \string<\glxtrMathItalicZeta
12470 \string<\glxtrMathItalicEta
12471 \string<\glxtrMathItalicTheta
12472 \string<\glxtrMathItalicIota
12473 \string<\glxtrMathItalicKappa
12474 \string<\glxtrMathItalicLambda
12475 \string<\glxtrMathItalicMu
12476 \string<\glxtrMathItalicNu
12477 \string<\glxtrMathItalicXi
12478 \string<\glxtrMathItalicOmicron
12479 \string<\glxtrMathItalicPi
12480 \string<\glxtrMathItalicRho
12481 \string<\glxtrMathItalicSigma
12482 \string<\glxtrMathItalicTau
12483 \string<\glxtrMathItalicUpsilon
12484 \string<\glxtrMathItalicPhi

```

```

12485 \string<\glxtrMathItalicChi
12486 \string<\glxtrMathItalicPsi
12487 \string<\glxtrMathItalicOmega
12488 }

```

licGreekIIrules Doesn't include digamma.

```

12489 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12490 \glxtrMathItalicAlpha
12491 \string<\glxtrMathItalicBeta
12492 \string<\glxtrMathItalicGamma
12493 \string<\glxtrMathItalicDelta
12494 \string<\glxtrMathItalicEpsilon
12495 \string<\glxtrMathItalicZeta
12496 \string<\glxtrMathItalicEta
12497 \string<\glxtrMathItalicTheta
12498 \string<\glxtrMathItalicIota
12499 \string<\glxtrMathItalicKappa
12500 \string<\glxtrMathItalicLambda
12501 \string<\glxtrMathItalicMu
12502 \string<\glxtrMathItalicNu
12503 \string<\glxtrMathItalicXi
12504 \string<\glxtrMathItalicOmicron
12505 \string<\glxtrMathItalicPi
12506 \string<\glxtrMathItalicRho
12507 \string<\glxtrMathItalicSigma
12508 \string<\glxtrMathItalicTau
12509 \string<\glxtrMathItalicUpsilon
12510 \string<\glxtrMathItalicPhi
12511 \string<\glxtrMathItalicChi
12512 \string<\glxtrMathItalicPsi
12513 \string<\glxtrMathItalicOmega
12514 }

```

pperGreekIrules Upper case only (includes upright digamma).

```

12515 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12516 \glshex 1D6E2% upper case alpha (maths italic)
12517 \string<\glshex 1D6E3% upper case beta (maths italic)
12518 \string<\glshex 1D6E4% upper case gamma (maths italic)
12519 \string<\glshex 1D6E5% upper case delta (maths italic)
12520 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12521 \string<\glshex 03DC% upper case digamma
12522 \string<\glshex 1D6E7% upper case zeta (maths italic)
12523 \string<\glshex 1D6E8% upper case eta (maths italic)
12524 \string<\glshex 1D6E9% upper case theta (maths italic)
12525 \string=<\glshex 1D6F3% upper case theta variant (maths italic)
12526 \string<\glshex 1D6EA% upper case iota (maths italic)
12527 \string<\glshex 1D6EB% upper case kappa (maths italic)
12528 \string<\glshex 1D6EC% upper case lambda (maths italic)
12529 \string<\glshex 1D6ED% upper case mu (maths italic)

```



```

12530 \string<\glshex 1D6EE% upper case nu (maths italic)
12531 \string<\glshex 1D6EF% upper case xi (maths italic)
12532 \string<\glshex 1D6F0% upper case omicron (maths italic)
12533 \string<\glshex 1D6F1% upper case pi (maths italic)
12534 \string<\glshex 1D6F2% upper case rho (maths italic)
12535 \string<\glshex 1D6F4% upper case sigma (maths italic)
12536 \string<\glshex 1D6F5% upper case tau (maths italic)
12537 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12538 \string<\glshex 1D6F7% upper case phi (maths italic)
12539 \string<\glshex 1D6F8% upper case chi (maths italic)
12540 \string<\glshex 1D6F9% upper case psi (maths italic)
12541 \string<\glshex 1D6FA% upper case omega (maths italic)
12542 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

12543 \newcommand*{\glstrMathItalicUpperGreekIIrules}{%
12544 \glshex 1D6E2% upper case alpha (maths italic)
12545 \string<\glshex 1D6E3% upper case beta (maths italic)
12546 \string<\glshex 1D6E4% upper case gamma (maths italic)
12547 \string<\glshex 1D6E5% upper case delta (maths italic)
12548 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12549 \string<\glshex 1D6E7% upper case zeta (maths italic)
12550 \string<\glshex 1D6E8% upper case eta (maths italic)
12551 \string<\glshex 1D6E9% upper case theta (maths italic)
12552 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12553 \string<\glshex 1D6EA% upper case iota (maths italic)
12554 \string<\glshex 1D6EB% upper case kappa (maths italic)
12555 \string<\glshex 1D6EC% upper case lambda (maths italic)
12556 \string<\glshex 1D6ED% upper case mu (maths italic)
12557 \string<\glshex 1D6EE% upper case nu (maths italic)
12558 \string<\glshex 1D6EF% upper case xi (maths italic)
12559 \string<\glshex 1D6F0% upper case omicron (maths italic)
12560 \string<\glshex 1D6F1% upper case pi (maths italic)
12561 \string<\glshex 1D6F2% upper case rho (maths italic)
12562 \string<\glshex 1D6F4% upper case sigma (maths italic)
12563 \string<\glshex 1D6F5% upper case tau (maths italic)
12564 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12565 \string<\glshex 1D6F7% upper case phi (maths italic)
12566 \string<\glshex 1D6F8% upper case chi (maths italic)
12567 \string<\glshex 1D6F9% upper case psi (maths italic)
12568 \string<\glshex 1D6FA% upper case omega (maths italic)
12569 }

```

lowerGreekIrules Lower case only (includes upright digamma).

```

12570 \newcommand*{\glstrMathItalicLowerGreekIrules}{%
12571 \glshex 1D6FC% lower case alpha (maths italic)
12572 \string<\glshex 1D6FD% lower case beta (maths italic)
12573 \string<\glshex 1D6FE% lower case gamma (maths italic)
12574 \string<\glshex 1D6FF% lower case delta (maths italic)

```

```

12575 \string<\glshex 1D700% lower case epsilon (maths italic)
12576 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12577 \string<\glshex 03DD% lower case digamma
12578 \string<\glshex 1D701% lower case zeta (maths italic)
12579 \string<\glshex 1D702% lower case eta (maths italic)
12580 \string<\glshex 1D703% lower case theta (maths italic)
12581 \string=\glshex 1D717% lower case theta variant (maths italic)
12582 \string<\glshex 1D704% lower case iota (maths italic)
12583 \string<\glshex 1D705% lower case kappa (maths italic)
12584 \string=\glshex 1D718% lower case kappa variant (maths italic)
12585 \string<\glshex 1D706% lower case lambda (maths italic)
12586 \string<\glshex 1D707% lower case mu (maths italic)
12587 \string<\glshex 1D708% lower case nu (maths italic)
12588 \string<\glshex 1D709% lower case xi (maths italic)
12589 \string<\glshex 1D70A% lower case omicron (maths italic)
12590 \string<\glshex 1D70B% lower case pi (maths italic)
12591 \string=\glshex 1D71B% lower case pi variant (maths italic)
12592 \string<\glshex 1D70C% lower case rho (maths italic)
12593 \string=\glshex 1D71A% lower case rho variant (maths italic)
12594 \string<\glshex 1D70D% lower case final sigma (maths italic)
12595 \string=\glshex 1D70E% lower case sigma (maths italic)
12596 \string<\glshex 1D70F% lower case tau (maths italic)
12597 \string<\glshex 1D710% lower case upsilon (maths italic)
12598 \string<\glshex 1D711% lower case phi (maths italic)
12599 \string=\glshex 1D719% lower case phi variant (maths italic)
12600 \string<\glshex 1D712% lower case chi (maths italic)
12601 \string<\glshex 1D713% lower case psi (maths italic)
12602 \string<\glshex 1D714% lower case omega (maths italic)
12603 }

```

werGreekIIrules Lower case only (doesn't includes upright digamma).

```

12604 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
12605 \glshex 1D6FC% lower case alpha (maths italic)
12606 \string<\glshex 1D6FD% lower case beta (maths italic)
12607 \string<\glshex 1D6FE% lower case gamma (maths italic)
12608 \string<\glshex 1D6FF% lower case delta (maths italic)
12609 \string<\glshex 1D700% lower case epsilon (maths italic)
12610 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12611 \string<\glshex 1D701% lower case zeta (maths italic)
12612 \string<\glshex 1D702% lower case eta (maths italic)
12613 \string<\glshex 1D703% lower case theta (maths italic)
12614 \string=\glshex 1D717% lower case theta variant (maths italic)
12615 \string<\glshex 1D704% lower case iota (maths italic)
12616 \string<\glshex 1D705% lower case kappa (maths italic)
12617 \string=\glshex 1D718% lower case kappa variant (maths italic)
12618 \string<\glshex 1D706% lower case lambda (maths italic)
12619 \string<\glshex 1D707% lower case mu (maths italic)
12620 \string<\glshex 1D708% lower case nu (maths italic)
12621 \string<\glshex 1D709% lower case xi (maths italic)

```

```

12622 \string<\glshex 1D70A% lower case omicron (maths italic)
12623 \string<\glshex 1D70B% lower case pi (maths italic)
12624 \string=<\glshex 1D71B% lower case pi variant (maths italic)
12625 \string<\glshex 1D70C% lower case rho (maths italic)
12626 \string=<\glshex 1D71A% lower case rho variant (maths italic)
12627 \string<\glshex 1D70D% lower case final sigma (maths italic)
12628 \string=<\glshex 1D70E% lower case sigma (maths italic)
12629 \string<\glshex 1D70F% lower case tau (maths italic)
12630 \string<\glshex 1D710% lower case upsilon (maths italic)
12631 \string<\glshex 1D711% lower case phi (maths italic)
12632 \string=<\glshex 1D719% lower case phi variant (maths italic)
12633 \string<\glshex 1D712% lower case chi (maths italic)
12634 \string<\glshex 1D713% lower case psi (maths italic)
12635 \string<\glshex 1D714% lower case omega (maths italic)
12636 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12637 \newcommand*{\glxtrMathGreekIrules}{%
12638 \glxtrMathItalicAlpha
12639 \string;\glxtrUpAlpha
12640 \string<\glxtrMathItalicBeta
12641 \string;\glxtrUpBeta
12642 \string<\glxtrMathItalicGamma
12643 \string;\glxtrUpGamma
12644 \string<\glxtrMathItalicDelta
12645 \string;\glxtrUpDelta
12646 \string<\glxtrMathItalicEpsilon
12647 \string;\glxtrUpEpsilon
12648 \string<\glxtrUpDigamma
12649 \string<\glxtrMathItalicZeta
12650 \string;\glxtrUpZeta
12651 \string<\glxtrMathItalicEta
12652 \string;\glxtrUpEta
12653 \string<\glxtrMathItalicTheta
12654 \string;\glxtrUpTheta
12655 \string<\glxtrMathItalicIota
12656 \string;\glxtrUpIota
12657 \string<\glxtrMathItalicKappa
12658 \string;\glxtrUpKappa
12659 \string<\glxtrMathItalicLambda
12660 \string;\glxtrUpLambda
12661 \string<\glxtrMathItalicMu
12662 \string;\glxtrUpMu
12663 \string<\glxtrMathItalicNu
12664 \string;\glxtrUpNu
12665 \string<\glxtrMathItalicXi
12666 \string;\glxtrUpXi
12667 \string<\glxtrMathItalicOmicron
12668 \string;\glxtrUpOmicron

```

```

12669 \string<\glxtrMathItalicPi
12670 \string;\glxtrUpPi
12671 \string<\glxtrMathItalicRho
12672 \string;\glxtrUpRho
12673 \string<\glxtrMathItalicSigma
12674 \string;\glxtrUpSigma
12675 \string<\glxtrMathItalicTau
12676 \string;\glxtrUpTau
12677 \string<\glxtrMathItalicUpsilon
12678 \string;\glxtrUpUpsilon
12679 \string<\glxtrMathItalicPhi
12680 \string;\glxtrUpPhi
12681 \string<\glxtrMathItalicChi
12682 \string;\glxtrUpChi
12683 \string<\glxtrMathItalicPsi
12684 \string;\glxtrUpPsi
12685 \string<\glxtrMathItalicOmega
12686 \string;\glxtrUpOmega
12687 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

12688 \newcommand*{\glxtrMathGreekIIrules}{%
12689 \glxtrMathItalicAlpha
12690 \string;\glxtrUpAlpha
12691 \string<\glxtrMathItalicBeta
12692 \string;\glxtrUpBeta
12693 \string<\glxtrMathItalicGamma
12694 \string;\glxtrUpGamma
12695 \string<\glxtrMathItalicDelta
12696 \string;\glxtrUpDelta
12697 \string<\glxtrMathItalicEpsilon
12698 \string;\glxtrUpEpsilon
12699 \string<\glxtrMathItalicZeta
12700 \string;\glxtrUpZeta
12701 \string<\glxtrMathItalicEta
12702 \string;\glxtrUpEta
12703 \string<\glxtrMathItalicTheta
12704 \string;\glxtrUpTheta
12705 \string<\glxtrMathItalicIota
12706 \string;\glxtrUpIota
12707 \string<\glxtrMathItalicKappa
12708 \string;\glxtrUpKappa
12709 \string<\glxtrMathItalicLambda
12710 \string;\glxtrUpLambda
12711 \string<\glxtrMathItalicMu
12712 \string;\glxtrUpMu
12713 \string<\glxtrMathItalicNu
12714 \string;\glxtrUpNu
12715 \string<\glxtrMathItalicXi

```

```

12716 \string;\glxtrUpXi
12717 \string<\glxtrMathItalicOmicron
12718 \string;\glxtrUpOmicron
12719 \string<\glxtrMathItalicPi
12720 \string;\glxtrUpPi
12721 \string<\glxtrMathItalicRho
12722 \string;\glxtrUpRho
12723 \string<\glxtrMathItalicSigma
12724 \string;\glxtrUpSigma
12725 \string<\glxtrMathItalicTau
12726 \string;\glxtrUpTau
12727 \string<\glxtrMathItalicUpsilon
12728 \string;\glxtrUpUpsilon
12729 \string<\glxtrMathItalicPhi
12730 \string;\glxtrUpPhi
12731 \string<\glxtrMathItalicChi
12732 \string;\glxtrUpChi
12733 \string<\glxtrMathItalicPsi
12734 \string;\glxtrUpPsi
12735 \string<\glxtrMathItalicOmega
12736 \string;\glxtrUpOmega
12737 }

```

\glxtrUpAlpha

```

12738 \newcommand*{\glxtrUpAlpha}{%
12739 \glshex 03B1,% lower case alpha
12740 \glshex 0391% upper case alpha
12741 }

```

\glxtrUpBeta

```

12742 \newcommand*{\glxtrUpBeta}{%
12743 \glshex 03B2,% lower case beta
12744 \glshex 0392% upper case beta
12745 }

```

\glxtrUpGamma

```

12746 \newcommand*{\glxtrUpGamma}{%
12747 \glshex 03B3,% lower case gamma
12748 \glshex 0393% upper case gamma
12749 }

```

\glxtrUpDelta

```

12750 \newcommand*{\glxtrUpDelta}{%
12751 \glshex 03B4,% lower case delta
12752 \glshex 0394% upper case delta
12753 }

```

\glxtrUpEpsilon

```

12754 \newcommand*{\glxtrUpEpsilon}{%
12755   \glshex 03B5% lower case epsilon
12756   \string=\glshex 03F5,% lower case epsilon variant
12757   \glshex 0395% upper case epsilon
12758 }

```

glxtrUpDigamma

```

12759 \newcommand*{\glxtrUpDigamma}{%
12760   \glshex 03DD,% lower case digamma
12761   \glshex 03DC% upper case digamma
12762 }

```

\glxtrUpZeta

```

12763 \newcommand*{\glxtrUpZeta}{%
12764   \glshex 03B6,% lower case zeta
12765   \glshex 0396% upper case zeta
12766 }

```

\glxtrUpEta

```

12767 \newcommand*{\glxtrUpEta}{%
12768   \glshex 03B7,% lower case eta
12769   \glshex 0397% upper case eta
12770 }

```

\glxtrUpTheta

```

12771 \newcommand*{\glxtrUpTheta}{%
12772   \glshex 03B8% lower case theta
12773   \string=\glshex 03D1,% lower case theta variant
12774   \glshex 0398% upper case theta
12775 }

```

\glxtrUpIota

```

12776 \newcommand*{\glxtrUpIota}{%
12777   \glshex 03B9,% lower case iota
12778   \glshex 0399% upper case iota
12779 }

```

\glxtrUpKappa

```

12780 \newcommand*{\glxtrUpKappa}{%
12781   \glshex 03BA% lower case kappa
12782   \string=\glshex 03F0,% lower case kappa variant
12783   \glshex 039A% upper case kappa
12784 }

```

\glxtrUpLambda

```

12785 \newcommand*{\glxtrUpLambda}{%
12786   \glshex 03BB,% lower lambda
12787   \glshex 039B% upper case lambda
12788 }

```

\backslash glxtrUpMu

```
12789 \newcommand*{\glxtrUpMu}{%
12790 \glshex 03BC,% lower case mu
12791 \glshex 039C% upper case mu
12792 }
```

\backslash glxtrUpNu

```
12793 \newcommand*{\glxtrUpNu}{%
12794 \glshex 03BD,% lower case nu
12795 \glshex 039D% upper case nu
12796 }
```

\backslash glxtrUpXi

```
12797 \newcommand*{\glxtrUpXi}{%
12798 \glshex 03BE,% lower case xi
12799 \glshex 039E% upper case xi
12800 }
```

glxtrUpOmicron

```
12801 \newcommand*{\glxtrUpOmicron}{%
12802 \glshex 03BF,% lower case omicron
12803 \glshex 039F% upper case omicron
12804 }
```

\backslash glxtrUpPi

```
12805 \newcommand*{\glxtrUpPi}{%
12806 \glshex 03C0% lower case pi
12807 \string=\glshex 03D6,% lower case pi variant
12808 \glshex 03A0% upper case pi
12809 }
```

\backslash glxtrUpRho

```
12810 \newcommand*{\glxtrUpRho}{%
12811 \glshex 03C1% lower case rho
12812 \string=\glshex 03F1,% lower case rho variant
12813 \glshex 03A1% upper case rho
12814 }
```

\backslash glxtrUpSigma

```
12815 \newcommand*{\glxtrUpSigma}{%
12816 \glshex 03C2% lower case sigma
12817 \string=\glshex 03C3,% lower case sigma
12818 \glshex 03A3% upper case sigma
12819 }
```

\backslash glxtrUpTau

```
12820 \newcommand*{\glxtrUpTau}{%
12821 \glshex 03C4,% lower case tau
```

```
12822 \glshex 03A4% upper case tau
12823 }
```

glsxtrUpUpsilon

```
12824 \newcommand*{\glxtrUpUpsilon}{%
12825 \glshex 03C5,% lower case upsilon
12826 \glshex 03A5% upper case upsilon
12827 }
```

\glxtrUpPhi

```
12828 \newcommand*{\glxtrUpPhi}{%
12829 \glshex 03C6% lower case phi
12830 \string=\glshex 03D5,% lower case phi variant
12831 \glshex 03A6% upper case phi
12832 }
```

\glxtrUpChi

```
12833 \newcommand*{\glxtrUpChi}{%
12834 \glshex 03C7,% lower case chi
12835 \glshex 03A7% upper case chi
12836 }
```

\glxtrUpPsi

```
12837 \newcommand*{\glxtrUpPsi}{%
12838 \glshex 03C8,% lower case psi
12839 \glshex 03A8% upper case psi
12840 }
```

\glxtrUpOmega

```
12841 \newcommand*{\glxtrUpOmega}{%
12842 \glshex 03C9,% lower case omega
12843 \glshex 03A9% upper case omega
12844 }
```

MathItalicAlpha

```
12845 \newcommand*{\glxtrMathItalicAlpha}{%
12846 \glshex 1D6FC,% lower case alpha (maths italic)
12847 \glshex 1D6E2% upper case alpha (maths italic)
12848 }
```

rMathItalicBeta

```
12849 \newcommand*{\glxtrMathItalicBeta}{%
12850 \glshex 1D6FD,% lower case beta (maths italic)
12851 \glshex 1D6E3% upper case beta (maths italic)
12852 }
```

MathItalicGamma

```
12853 \newcommand*{\glxtrMathItalicGamma}{%
```



```

12854 \glshex 1D6FE,% lower case gamma (maths italic)
12855 \glshex 1D6E4% upper case gamma (maths italic)
12856 }

```

MathItalicDelta

```

12857 \newcommand*{\glsxtrMathItalicDelta}{%
12858 \glshex 1D6FF,% lower case delta (maths italic)
12859 \glshex 1D6E5% upper case delta (maths italic)
12860 }

```

thItalicEpsilon

```

12861 \newcommand*{\glsxtrMathItalicEpsilon}{%
12862 \glshex 1D700% lower case epsilon (maths italic)
12863 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12864 \glshex 1D6E6% upper case epsilon (maths italic)
12865 }

```

rMathItalicZeta

```

12866 \newcommand*{\glsxtrMathItalicZeta}{%
12867 \glshex 1D701,% lower case zeta (maths italic)
12868 \glshex 1D6E7% upper case zeta (maths italic)
12869 }

```

trMathItalicEta

```

12870 \newcommand*{\glsxtrMathItalicEta}{%
12871 \glshex 1D702,% lower case eta (maths italic)
12872 \glshex 1D6E8% upper case eta (maths italic)
12873 }

```

MathItalicTheta

```

12874 \newcommand*{\glsxtrMathItalicTheta}{%
12875 \glshex 1D703% lower case theta (maths italic)
12876 \string=\glshex 1D717,% lower case theta variant (maths italic)
12877 \glshex 1D6E9% upper case theta (maths italic)
12878 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12879 }

```

rMathItalicIota

```

12880 \newcommand*{\glsxtrMathItalicIota}{%
12881 \glshex 1D704,% lower case iota (maths italic)
12882 \glshex 1D6EA% upper case iota (maths italic)
12883 }

```

MathItalicKappa

```

12884 \newcommand*{\glsxtrMathItalicKappa}{%
12885 \glshex 1D705% lower case kappa (maths italic)
12886 \string=\glshex 1D718,% lower case kappa variant (maths italic)
12887 \glshex 1D6EB% upper case kappa (maths italic)
12888 }

```

athItalicLambda

```
12889 \newcommand*{\glxtrMathItalicLambda}{%
12890 \glshex 1D706,% lower case lambda (maths italic)
12891 \glshex 1D6EC% upper case lambda (maths italic)
12892 }
```

xtrMathItalicMu

```
12893 \newcommand*{\glxtrMathItalicMu}{%
12894 \glshex 1D707,% lower case mu (maths italic)
12895 \glshex 1D6ED% upper case mu (maths italic)
12896 }
```

xtrMathItalicNu

```
12897 \newcommand*{\glxtrMathItalicNu}{%
12898 \glshex 1D708,% lower case nu (maths italic)
12899 \glshex 1D6EE% upper case nu (maths italic)
12900 }
```

xtrMathItalicXi

```
12901 \newcommand*{\glxtrMathItalicXi}{%
12902 \glshex 1D709,% lower case xi (maths italic)
12903 \glshex 1D6EF% upper case xi (maths italic)
12904 }
```

thItalicOmicron

```
12905 \newcommand*{\glxtrMathItalicOmicron}{%
12906 \glshex 1D70A,% lower case omicron (maths italic)
12907 \glshex 1D6F0% upper case omicron (maths italic)
12908 }
```

xtrMathItalicPi

```
12909 \newcommand*{\glxtrMathItalicPi}{%
12910 \glshex 1D70B% lower case pi (maths italic)
12911 \string=\glshex 1D71B,% lower case pi variant (maths italic)
12912 \glshex 1D6F1% upper case pi (maths italic)
12913 }
```

trMathItalicRho

```
12914 \newcommand*{\glxtrMathItalicRho}{%
12915 \glshex 1D70C% lower case rho (maths italic)
12916 \string=\glshex 1D71A,% lower case rho variant (maths italic)
12917 \glshex 1D6F2% upper case rho (maths italic)
12918 }
```

MathItalicSigma

```
12919 \newcommand*{\glxtrMathItalicSigma}{%
12920 \glshex 1D70D% lower case final sigma (maths italic)
12921 \string=\glshex 1D70E,% lower case sigma (maths italic)
```

```

12922 \glshex 1D6F4% upper case sigma (maths italic)
12923 }

```

trMathItalicTau

```

12924 \newcommand*{\glxtrMathItalicTau}{%
12925 \glshex 1D70F,% lower case tau (maths italic)
12926 \glshex 1D6F5% upper case tau (maths italic)
12927 }

```

thItalicUpsilon

```

12928 \newcommand*{\glxtrMathItalicUpsilon}{%
12929 \glshex 1D710,% lower case upsilon (maths italic)
12930 \glshex 1D6F6% upper case upsilon (maths italic)
12931 }

```

trMathItalicPhi

```

12932 \newcommand*{\glxtrMathItalicPhi}{%
12933 \glshex 1D711% lower case phi (maths italic)
12934 \string=\glshex 1D719,% lower case phi variant (maths italic)
12935 \glshex 1D6F7% upper case phi (maths italic)
12936 }

```

trMathItalicChi

```

12937 \newcommand*{\glxtrMathItalicChi}{%
12938 \glshex 1D712,% lower case chi (maths italic)
12939 \glshex 1D6F8% upper case chi (maths italic)
12940 }

```

trMathItalicPsi

```

12941 \newcommand*{\glxtrMathItalicPsi}{%
12942 \glshex 1D713,% lower case psi (maths italic)
12943 \glshex 1D6F9% upper case psi (maths italic)
12944 }

```

MathItalicOmega

```

12945 \newcommand*{\glxtrMathItalicOmega}{%
12946 \glshex 1D714,% lower case omega (maths italic)
12947 \glshex 1D6FA% upper case omega (maths italic)
12948 }

```

thItalicPartial

```

12949 \newcommand*{\glxtrMathItalicPartial}{%
12950 \glshex 1D715% partial differential (maths italic)
12951 }

```

MathItalicNabla

```

12952 \newcommand*{\glxtrMathItalicNabla}{%
12953 \glshex 1D6FB% nabla (maths italic)
12954 }

```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12955 \newcommand*{\glxtrdigitrules}{%
12956 0\string=\glshex 2080\string=\glshex 2070
12957 \string<1\string=\glshex 2081\string=\glshex 00B9
12958 \string<2\string=\glshex 2082\string=\glshex 00B2
12959 \string<3\string=\glshex 2083\string=\glshex 00B3
12960 \string<4\string=\glshex 2084\string=\glshex 2074
12961 \string<5\string=\glshex 2085\string=\glshex 2075
12962 \string<6\string=\glshex 2086\string=\glshex 2076
12963 \string<7\string=\glshex 2087\string=\glshex 2077
12964 \string<8\string=\glshex 2088\string=\glshex 2078
12965 \string<9\string=\glshex 2089\string=\glshex 2079
12966 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12967 \newcommand*{\glxtrBasicDigitrules}{%
12968 0\string<1\string<2\string<3\string<4%
12969 \string<5\string<6\string<7\string<8\string<9%
12970 }
```

criptDigitrules Subscript digits.

```
12971 \newcommand*{\glxtrSubScriptDigitrules}{%
12972 \glshex 2080% subscript 0
12973 \string<\glshex 2081% subscript 1
12974 \string<\glshex 2082% subscript 2
12975 \string<\glshex 2083% subscript 3
12976 \string<\glshex 2084% subscript 4
12977 \string<\glshex 2085% subscript 5
12978 \string<\glshex 2086% subscript 6
12979 \string<\glshex 2087% subscript 7
12980 \string<\glshex 2088% subscript 8
12981 \string<\glshex 2089% subscript 9
12982 }
```

criptDigitrules Superscript digits.

```
12983 \newcommand*{\glxtrSuperScriptDigitrules}{%
12984 \glshex 2070% superscript 0
12985 \string<\glshex 00B9% superscript 1
12986 \string<\glshex 00B2% superscript 2
12987 \string<\glshex 00B3% superscript 3
12988 \string<\glshex 2074% superscript 4
12989 \string<\glshex 2075% superscript 5
12990 \string<\glshex 2076% superscript 6
12991 \string<\glshex 2077% superscript 7
12992 \string<\glshex 2078% superscript 8
12993 \string<\glshex 2079% superscript 9
12994 }
```

trfractionrules Vulgar fractions.

```
12995 \newcommand*{\glxtrfractionrules}{%
12996   \glshex 215F% fraction numerator one (1/)
12997   \string<\glshex 2189% zero thirds (0/3 = 0)
12998   \string<\glshex 2152% one tenth (1/10 = 0.1)
12999   \string<\glshex 2151% one ninth (1/9 ~ 0.111)
13000   \string<\glshex 215B% one eighth (1/8 = 0.125)
13001   \string<\glshex 2150% one seventh (1/7 ~ 0.143)
13002   \string<\glshex 2159% one sixth (1/6 ~ 0.167)
13003   \string<\glshex 2155% one fifth (1/5 = 0.2)
13004   \string<\glshex 00BC% one quarter (1/4 = 0.25)
13005   \string<\glshex 2153% one third (1/3 ~ 0.333)
13006   \string<\glshex 215C% three eighths (3/8 = 0.375)
13007   \string<\glshex 2156% two fifths (2/5 = 0.4)
13008   \string<\glshex 00BD% one half (1/2 = 0.5)
13009   \string<\glshex 2157% three fifths (3/5 = 0.6)
13010   \string<\glshex 215D% five eighths (5/8 = 0.625)
13011   \string<\glshex 2154% two thirds (2/3 ~ 0.667)
13012   \string<\glshex 00BE% three quarters (3/4 = 0.75)
13013   \string<\glshex 2158% four fifths (4/5 = 0.8)
13014   \string<\glshex 215A% five sixths (5/6 ~ 0.833)
13015   \string<\glshex 215E% seven eighths (7/8 = 0.875)
13016 }
```

sxtrdialecthook Check for scripts associated with the document dialects.

```
13017 \renewcommand{\@glxtrdialecthook}{%
13018   \ifundef\CurrentTrackedScript
13019   {%
13020     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13021     {%
13022       \edef\CurrentTrackedScript{%
13023         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13024       }%
13025     }%
13026   }%
13027   {}%
13028   \ifdef\CurrentTrackedScript
13029   {%
13030     \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
13031     \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
13032     \let\CurrentTrackedTag\CurrentTrackedScript
13033     \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
13034     {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13035     {}%
13036     \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13037   }%
13038   {}%
13039 }
```

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.

```

13040 \ifdef\glsxtr@loaddialect
13041 {%
13042   \@ifpackageloaded{tracklang}
13043   {%
13044     \AnyTrackedLanguages
13045     {%
13046       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13047     }%
13048   }%
13049 }
13050 {}
13051 }
13052 {}

```

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:

```
13053 \NeedsTeXFormat{LaTeX2e}
13054 \ProvidesPackage{glossaries-extra-stylemods}[2018/08/13 v1.35 (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`

```
13055 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13056 \DeclareOption{all}{%
13057   \appto\@glxtr@loadstyles{%
13058     \RequirePackage{glossary-inline}%
13059     \RequirePackage{glossary-list}%
13060     \RequirePackage{glossary-tree}%
13061     \RequirePackage{glossary-mcols}%
13062     \RequirePackage{glossary-long}%
13063     \RequirePackage{glossary-longragged}%
13064     \RequirePackage{glossary-longbooktabs}%
13065     \RequirePackage{glossary-super}%
13066     \RequirePackage{glossary-superragged}%
13067     \RequirePackage{glossary-bookindex}%
13068   }
13069 }

13070 \DeclareOption*{%
13071   \IfFileExists{glossary-\CurrentOption.sty}
13072   {\eappto\@glxtr@loadstyles{%
13073     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13074   }%
13075   {%
13076     \PackageError{glossaries-extra-styles}%

```

```

13077     {Unknown option ‘\CurrentOption’}{}%
13078   }%
13079 }

```

Process the package options:

```
13080 \ProcessOptions
```

Load the required packages:

```
13081 \@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`.

```
13082 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

13083 \providecommand{\renewglossarystyle}[2]{%
13084   \ifcsundef{@glsstyle@#1}%
13085   {%
13086     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13087   }%
13088   {%
13089     \csdef{@glsstyle@#1}{#2}%
13090   }%
13091 }

```

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.

```

13092 \ifdef{@glsstyle@listdotted}
13093 {%
13094   \renewglossarystyle{listdotted}{%
13095     \setglossarystyle{list}%
13096     \renewcommand*{\glossentry}[2]{%
13097       \item[]\makebox[\glslistdottedwidth][l]{%
13098         \glsentryitem{##1}%
13099         \glstarget{##1}{\glossentryname{##1}}%
13100         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13101         \glossentrydesc{##1}\glspostdescription}%
13102     \renewcommand*{\subglossentry}[3]{%
13103       \item[]\makebox[\glslistdottedwidth][l]{%
13104         \glssubentryitem{##2}%
13105         \glstarget{##2}{\glossentryname{##2}}%
13106         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13107         \glossentrydesc{##2}\glspostdescription}%
13108   }

```



```
13109 }
13110 {%
```

Assume the style isn't required if it hasn't already been defined.

```
13111 }
```

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.

```
13112 \ifdef{\@glsstyle@list}
13113 {%
```

listprelocation Space before number list for top-level entries.

```
13114 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13115 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13116 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13117 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
13118 \renewglossarystyle{list}{%
13119 \renewenvironment{theglossary}%
13120 {\begin{description}}{\end{description}}%
13121 \renewcommand*\{glossaryheader}{}%
13122 \renewcommand*\{glsgroupheading}[1]{}%
13123 \renewcommand*\{glossentry}[2]{%
13124 \item[\glssentryitem{##1}%
13125 \glstarget{##1}{\glossentryname{##1}}]
13126 \glslistdesc{##1}\glslistprelocation ##2}%
13127 \renewcommand*\{subglossentry}[3]{%
13128 \glssubentryitem{##2}%
13129 \glstarget{##2}{\strut}\space
13130 \glslistdesc{##2}%
13131 \glslistchildprelocation ##3\glslistchildpostlocation}%
13132 \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13133 }
13134 }
13135 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13136 \ifdef{\@glsstyle@altlist}
13137 {%
13138 \renewglossarystyle{altlist}{%
```

```

13139 \setglossarystyle{list}%
13140 \renewcommand*{\glossentry}[2]{%
13141   \item[\glentryitem{##1}%
13142     \glstarget{##1}{\glossentryname{##1}}]%
13143   \mbox{}\par\nobreak\@afterheading
13144   \glslistdesc{##1}\glslistprelocation ##2}%
13145 \renewcommand{\subglossentry}[3]{%
13146   \par
13147   \glssubentryitem{##2}%
13148   \glstarget{##2}{\strut}\glslistdesc{##2}%
13149   \glslistchildprelocation ##3}%
13150 }
13151 }
13152 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13153 \ifdef{\@glsstyle@listgroup}
13154 {%
13155   \renewglossarystyle{listgroup}{%
13156     \setglossarystyle{list}%
13157     \renewcommand*{\glsgroupheading}[1]{%
13158       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13159       \mbox{}\par\nobreak\@afterheading
13160     }%
13161   }
13162 }
13163 {}

```

Similarly for listhypergroup.

```

13164 \ifdef{\@glsstyle@listhypergroup}
13165 {%
13166   \renewglossarystyle{listhypergroup}{%
13167     \setglossarystyle{list}%
13168     \renewcommand*{\glossaryheader}{%
13169       \glslistnavigationitem{\glsnavigation}}%
13170     \renewcommand*{\glsgroupheading}[1]{%
13171       \item[\glslistgroupheaderfmt
13172         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13173       \mbox{}\par\nobreak\@afterheading
13174     }%
13175   }
13176 }
13177 {}

```

Similarly for altlistgroup.

```

13178 \ifdef{\@glsstyle@altlistgroup}
13179 {%
13180   \renewglossarystyle{altlistgroup}{%
13181     \setglossarystyle{altlist}%
13182     \renewcommand*{\glsgroupheading}[1]{%
13183       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```

```

13184     \mbox{}\par\nobreak\@afterheading
13185   }%
13186 }
13187 }
13188 {}

```

Similarly for altlisthypergroup.

```

13189 \ifdef{\@glsstyle@altlisthypergroup}
13190 {%
13191   \renewglossarystyle{altlisthypergroup}{%
13192     \setglossarystyle{altlist}%
13193     \renewcommand*\glossaryheader{%
13194       \glslistnavigationitem{\glsnavigation}}%
13195     \renewcommand*\glsgroupheading[1]{%
13196       \item[\glslistgroupheaderfmt
13197         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13198     \mbox{}\par\nobreak\@afterheading
13199   }%
13200 }
13201 }
13202 {}

```

2.3 Longtable Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glxtrprelocation`.

```

13203 \ifcsdef{@glsstyle@long}
13204 {%
13205   \renewglossarystyle{long}{%
13206     \renewenvironment{theglossary}%
13207       {\begin{longtable}[lp{\glsdescwidth}]}%
13208       {\end{longtable}}%
13209     \renewcommand*\glossaryheader{}%
13210     \renewcommand*\glsgroupheading[1]{}%
13211     \renewcommand{\glossentry}[2]{%
13212       \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13213       \glossentrydesc{##1}\glspostdescription
13214       \glxtrprelocation ##2\tabularnewline
13215     }%
13216     \renewcommand{\subglossentry}[3]{%
13217       &
13218       \glssubentryitem{##2}%
13219       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13220       \glxtrprelocation ##3\tabularnewline
13221     }%
13222     \ifglsnogroupskip
13223       \renewcommand*\glsgroupskip{}%
13224     \else

```

```

13225     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13226     \fi
13227 }
13228 }
13229 {}

```

Three column style:

```

13230 \ifcsdef{@glsstyle@long3col}
13231 {%
13232     \renewglossarystyle{long3col}{%
13233         \renewenvironment{theglossary}%
13234             {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13235             {\end{longtable}}}%
13236     \renewcommand*{\glossaryheader}{}%
13237     \renewcommand*{\glsgroupheading}[1]{}%
13238     \renewcommand{\glossentry}[2]{%
13239         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13240         \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13241     }%
13242     \renewcommand{\subglossentry}[3]{%
13243         &
13244         \glssubentryitem{##2}%
13245         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13246         ##3\tabularnewline
13247     }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

13248     \ifglsnogroupskip
13249         \renewcommand*{\glsgroupskip}{}%
13250     \else
13251         \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13252     \fi
13253 }
13254 }
13255 {}

```

Four column style:

```

13256 \ifcsdef{@glsstyle@long4col}
13257 {%
13258     \renewglossarystyle{long4col}{%
13259         \renewenvironment{theglossary}%
13260             {\begin{longtable}{llll}}}%
13261             {\end{longtable}}}%
13262     \renewcommand*{\glossaryheader}{}%
13263     \renewcommand*{\glsgroupheading}[1]{}%
13264     \renewcommand{\glossentry}[2]{%
13265         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13266         \glossentrydesc{##1}\glspostdescription &
13267         \glossentrysymbol{##1} &
13268         ##2\tabularnewline

```

```

13269 }%
13270 \renewcommand{\subglossentry}[3]{%
13271     &
13272     \glssubentryitem{##2}%
13273     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13274     \glossentrysymbol{##2} & ##3\tabularnewline
13275 }%

13276 \ifglsgroupskip
13277     \renewcommand*\{glsgroupskip}{}%
13278 \else
13279     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13280 \fi
13281 }
13282 }
13283 {}

```

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`.

```

13284 \ifcsdef{@glstyle@longragged}
13285 {%
13286     \renewglossarystyle{longragged}{%
13287         \renewenvironment{theglossary}%
13288             {\begin{longtable}[l>\raggedright]p{\glstdescwidth}}}%
13289             {\end{longtable}}}%
13290     \renewcommand*\{glossaryheader}{}%
13291     \renewcommand*\{glsgroupheading}[1]{}%
13292     \renewcommand{\glossentry}[2]{%
13293         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13294         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13295         \tabularnewline
13296     }%
13297     \renewcommand{\subglossentry}[3]{%
13298         &
13299         \glssubentryitem{##2}%
13300         \glstarget{##2}{\strut}\glossentrydesc{##2}%
13301         \glspostdescription\glxtrprelocation ##3%
13302         \tabularnewline
13303     }%
13304     \ifglsgroupskip
13305         \renewcommand*\{glsgroupskip}{}%
13306     \else
13307         \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13308   \fi
13309 }
13310 }
13311 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13312 \ifcsdef{@glsstyle@longragged3col}
13313 {%
13314   \renewglossarystyle{longragged3col}{%
13315     \renewenvironment{theglossary}%
13316       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13317         >{\raggedright}p{\glspagelistwidth}}}%
13318       {\end{longtable}}}%
13319     \renewcommand*{\glossaryheader}{}%
13320     \renewcommand*{\glsgroupheading}[1]{}%
13321     \renewcommand{\glossentry}[2]{%
13322       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13323       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13324     }%
13325     \renewcommand{\subglossentry}[3]{%
13326       &
13327       \glssubentryitem{##2}%
13328       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13329       ##3\tabularnewline
13330     }%
13331   \ifglsnogroupskip
13332     \renewcommand*{\glsgroupskip}{}%
13333   \else
13334     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13335   \fi
13336 }
13337 }
13338 {}

```

Four column style:

```

13339 \ifcsdef{@glsstyle@altlongragged4col}
13340 {%
13341   \renewglossarystyle{altlongragged4col}{%
13342     \renewenvironment{theglossary}%
13343       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13344         >{\raggedright}p{\glspagelistwidth}}}%
13345       {\end{longtable}}}%
13346     \renewcommand*{\glossaryheader}{}%
13347     \renewcommand*{\glsgroupheading}[1]{}%
13348     \renewcommand{\glossentry}[2]{%
13349       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13350       \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13351       ##2\tabularnewline

```

```

13352 }%
13353 \renewcommand{\subglossentry}[3]{%
13354     &
13355     \glssubentryitem{##2}%
13356     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13357     \glossentrysymbol{##2} & ##3\tabularnewline
13358 }%

13359 \ifglsgnogroupskip
13360     \renewcommand*{\glsgroupskip}{}%
13361 \else
13362     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13363 \fi
13364 }
13365 }
13366 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glstrprelocation`.

```

13367 \ifcsdef{@glstyle@super}
13368 {%
13369     \renewglossarystyle{super}{%
13370         \renewenvironment{theglossary}%
13371             {\tablehead{}}\tabletail{}}%
13372         \begin{supertabular}{lp{\glsgdescwidth}}%
13373             {\end{supertabular}}%
13374         \renewcommand*{\glossaryheader}{}%
13375         \renewcommand*{\glsgroupheading}[1]{}%
13376         \renewcommand{\glossentry}[2]{%
13377             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13378             \glossentrydesc{##1}\glspostdescription
13379             \glstrprelocation ##2\tabularnewline
13380         }%
13381         \renewcommand{\subglossentry}[3]{%
13382             &
13383             \glssubentryitem{##2}%
13384             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13385             \glstrprelocation ##3\tabularnewline
13386         }%
13387         \ifglsgnogroupskip
13388             \renewcommand*{\glsgroupskip}{}%
13389         \else
13390             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13391         \fi
13392     }
13393 }
13394 {}

```

Three column style:

```

13395 \ifcsdef{@glsstyle@super3col}
13396 {%
13397   \renewglossarystyle{super3col}{%
13398     \renewenvironment{theglossary}%
13399       {\tablehead{}\tabletail{}}%
13400       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13401     {\end{supertabular}}}%
13402   \renewcommand*{\glossaryheader}{}%
13403   \renewcommand*{\glsgroupheading}[1]{}%
13404   \renewcommand{\glossentry}[2]{%
13405     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13406     \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
13407   }%
13408   \renewcommand{\subglossentry}[3]{%
13409     &
13410     \glssubentryitem{##2}%
13411     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13412     ##3\tabularnewline
13413   }%
13414   \ifglsgroupskip
13415     \renewcommand*{\glsgroupskip}{}%
13416   \else
13417     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13418   \fi
13419 }
13420 }
13421 {}

```

Four column styles:

```

13422 \ifcsdef{@glsstyle@super4col}
13423 {%
13424   \renewglossarystyle{super4col}{%
13425     \renewenvironment{theglossary}%
13426       {\tablehead{}\tabletail{}}%
13427       \begin{supertabular}{llll}}}%
13428     \end{supertabular}}}%
13429   \renewcommand*{\glossaryheader}{}%
13430   \renewcommand*{\glsgroupheading}[1]{}%
13431   \renewcommand{\glossentry}[2]{%
13432     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13433     \glsentrydesc{##1}\glspostdescription &
13434     \glsentrysymbol{##1} & ##2\tabularnewline
13435   }%
13436   \renewcommand{\subglossentry}[3]{%
13437     &
13438     \glssubentryitem{##2}%
13439     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13440     \glsentrysymbol{##2} & ##3\tabularnewline

```



```

13441 }%
13442 \ifglsgroupskip
13443 \renewcommand*{\glsgroupskip}{}%
13444 \else
13445 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13446 \fi
13447 }
13448 }
13449 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glstrprelocation`.

```

13450 \ifcsdef{@glstyle@superragged}
13451 {%
13452 \renewglossarystyle{superragged}{%
13453 \renewenvironment{theglossary}%
13454 {\tablehead{}\tabletail}%
13455 \begin{supertabular}{l>\raggedright}p{\glsgdescwidth}}%
13456 {\end{supertabular}}%
13457 \renewcommand*{\glossaryheader}{}%
13458 \renewcommand*{\glsgroupheading}[1]{}%
13459 \renewcommand{\glossentry}[2]{%
13460 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13461 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
13462 \tabularnewline
13463 }%
13464 \renewcommand{\subglossentry}[3]{%
13465 &
13466 \glssubentryitem{##2}%
13467 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13468 \glstrprelocation ##3%
13469 \tabularnewline
13470 }%
13471 \ifglsgroupskip
13472 \renewcommand*{\glsgroupskip}{}%
13473 \else
13474 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13475 \fi
13476 }
13477 }
13478 {}

```

Three column style:

```

13479 \ifcsdef{@glstyle@superragged3col}
13480 {%

```

```

13481 \renewglossarystyle{superragged3col}{%
13482   \renewenvironment{theglossary}%
13483     {\tablehead{}\tabletail{}}%
13484     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
13485       >{\raggedright}p{\glspagelistwidth}}}%
13486     {\end{supertabular}}%
13487   \renewcommand*{\glossaryheader}{}%
13488   \renewcommand*{\glsgroupheading}[1]{}%
13489   \renewcommand{\glossentry}[2]{%
13490     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13491     \glossentrydesc{##1}\glspostdescription &
13492     ##2\tabularnewline
13493   }%
13494   \renewcommand{\subglossentry}[3]{%
13495     &
13496     \glssubentryitem{##2}%
13497     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13498     ##3\tabularnewline
13499   }%

13500   \ifglsnogroupskip
13501     \renewcommand*{\glsgroupskip}{}%
13502   \else
13503     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13504   \fi
13505 }
13506 }
13507 {}

```

Four columns:

```

13508 \ifcsdef{@glsstyle@altsuperragged4col}
13509 {%
13510   \renewglossarystyle{altsuperragged4col}{%
13511     \renewenvironment{theglossary}%
13512       {\tablehead{}\tabletail{}}%
13513       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
13514         >{\raggedright}p{\glspagelistwidth}}}%
13515       {\end{supertabular}}%
13516     \renewcommand*{\glossaryheader}{}%
13517     \renewcommand{\glossentry}[2]{%
13518       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13519       \glossentrydesc{##1}\glspostdescription &
13520       \glossentrysymbol{##1} & ##2\tabularnewline
13521     }%
13522     \renewcommand{\subglossentry}[3]{%
13523       &
13524       \glssubentryitem{##2}%
13525       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13526       \glossentrysymbol{##2} & ##3\tabularnewline
13527     }%

```

```

13528 \ifglsnogroupskip
13529 \renewcommand*{\glsgroupskip}{}%
13530 \else
13531 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13532 \fi
13533 }
13534 }
13535 {}

```

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`.

```

13536 \ifdef{\@glsstyle@inline}
13537 {%
13538 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13539 \renewcommand*{\glsinlinedescformat}[3]{%
13540 \space#1\glxtrpostdescription}
13541 \renewcommand*{\glsinlinesubdescformat}[3]{%
13542 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13543 }
13544 {}

```

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.

```

13545 \ifdef\glstreenamfmt
13546 {

```

```

\glstreedefaultnamfmt

```

```

13547 \newcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}

```

```

\glstreenamfmt

```

```

13548 \renewcommand{\glstreenamfmt}[1]{\glstreedefaultnamfmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13549 \def\glstreegroupheaderfmt#1{\glstreedefaultnamfmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13550 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13551 }
13552 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13553 \ifdef{\@glsstyle@index}
13554 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13555 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13556 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13557 \renewglossarystyle{index}{%
13558   \renewenvironment{theglossary}%
13559     {\setlength{\parindent}{0pt}%
13560      \setlength{\parskip}{0pt plus 0.3pt}%
13561      \let\item\glstreeitem
13562      \let\subitem\glstreesubitem
13563      \let\subsubitem\glstreesubsubitem
13564     }%
13565   {\par}%
13566   \renewcommand*{\glossaryheader}{}%
13567   \renewcommand*{\glsgroupheading}[1]{}%
13568   \renewcommand*{\glossentry}[2]{}%
13569     \item\glssentryitem{##1}%
13570     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13571     \glstreesymbol{##1}%
13572     \glstreedesc{##1}%
13573     \glstreeprelocation ##2%
13574   }%
13575   \renewcommand{\subglossentry}[3]{}%
13576     \ifcase##1\relax
13577       \item
13578     \or
13579       \subitem
13580       \glssubentryitem{##2}%
13581     \else
13582       \subsubitem
13583     \fi
13584     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13585     \glstreechildsymbol{##2}%
13586     \glstreechilddesc{##2}%
13587     \glstreechildprelocation ##3%
13588   }%
```

```

13589 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
13590 }
13591 }
13592 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13593 \ifdef{\@glsstyle@indexgroup}
13594 {%
13595 \renewglossarystyle{indexgroup}{%
13596 \setglossarystyle{index}%
13597 \renewcommand*{\glsgroupheading}[1]{%
13598 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13599 \nopagebreak\indexspace
13600 \nobreak\@afterheading
13601 }%
13602 }
13603 }
13604 {}

```

Similarly for indexhypergroup.

```

13605 \ifdef{\@glsstyle@indexhypergroup}
13606 {%
13607 \renewglossarystyle{indexhypergroup}{%
13608 \setglossarystyle{index}%
13609 \renewcommand*{\glossaryheader}{%
13610 \item\glstreenavigationfmt{\glsnavigation}%
13611 \nobreak\@afterheading\indexspace}%
13612 \renewcommand*{\glsgroupheading}[1]{%
13613 \item\glstreegroupheaderfmt
13614 {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13615 \nopagebreak\indexspace
13616 \nobreak\@afterheading}%
13617 }%
13618 }
13619 {}

```

Adjust tree style to remove hard coded space before number list.

```

13620 \ifdef{\@glsstyle@tree}
13621 {%
13622 %Provide a command for use with the \glostyle{tree} styles that displays
13623 %the pre-description separator, the
13624 %description and post-description hook.
13625 %\begin{macro}{\glstreedesc}
13626 %\changes{1.31}{2018-05-09}{new}
13627 % \begin{macrocode}
13628 \newcommand{\glstreedesc}[1]{%
13629 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13630 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13631 \newcommand{\glstreesymbol}[1]{%
13632   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13633 }%
```

And for the child entries:

`lstreechilddesc`

```
13634 \newcommand{\glstreechilddesc}[1]{%
13635   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13636 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13637 \newcommand{\glstreechildsymbol}[1]{%
13638   \glstreesymbol{#1}%
13639 }%

13640 \renewglossarystyle{tree}{%
13641   \renewenvironment{theglossary}%
13642     {\setlength{\parindent}{0pt}%
13643     \setlength{\parskip}{0pt plus 0.3pt}}%
13644   {%
13645     \renewcommand*{\glossaryheader}{}%
13646     \renewcommand*{\glsgroupheading}[1]{}%
13647     \renewcommand{\glossentry}[2]{%
13648       \hangindent0pt\relax
13649       \parindent0pt\relax
13650       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13651       \glstreesymbol{##1}%
13652       \glstreedesc{##1}%
13653       \glstreeprelocation##2\par
13654     }%
13655     \renewcommand{\subglossentry}[3]{%
13656       \hangindent##1\glstreeindent\relax
13657       \parindent##1\glstreeindent\relax
13658       \ifnum##1=1\relax
13659         \glssubentryitem{##2}%
13660       \fi
13661       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13662       \glstreechildsymbol{##2}%
13663       \glstreechilddesc{##2}%
13664       \glstreechildprelocation ##3\par
13665     }%
13666     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13667   }%
13668 }
13669 }
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13670 \ifdef{\@glstyle@treegroup}
```

```

13671 {%
13672   \renewglossarystyle{treegroup}{%
13673     \setglossarystyle{tree}%
13674     \renewcommand{\glsgroupheading}[1]{\par
13675       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13676       \nopagebreak\indexspace\nobreak\@afterheading}%
13677   }
13678 }
13679 {}

```

Similarly for treehypergroup

```

13680 \ifdef{\@glsstyle@treehypergroup}
13681 {%
13682   \renewglossarystyle{treehypergroup}{%
13683     \setglossarystyle{tree}%
13684     \renewcommand*\{\glossaryheader}{%
13685       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13686       \nobreak\@afterheading\indexspace}%
13687     \renewcommand*\{\glsgroupheading}[1]{%
13688       \par\noindent
13689       \glstreegroupheaderfmt
13690       {\glstravigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
13691       \nopagebreak\indexspace\nobreak\@afterheading}%
13692   }
13693 }
13694 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13695 \ifdef{\@glsstyle@treenoname}
13696 {%
13697 %Provide a command for use with the \glsstyle{treenoname} styles that displays
13698 %the pre-description separator, the
13699 %description and post-description hook.
13700 %\begin{macro}{\glstreenonamedesc}
13701 %\changes{1.31}{2018-05-09}{new}
13702 %   \begin{macrocode}
13703   \newcommand{\glstreenonamedesc}[1]{%
13704     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13705   }%

```

Similarly for the symbol.

treenonamesymbol

```

13706   \newcommand{\glstreenonamesymbol}[1]{%
13707     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13708   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13709   \newcommand{\glstreenonamechilddesc}[1]{%
13710     \glossentrydesc{#1}\glspostdescription
13711   }%

```

```

13712 \renewglossarystyle{treenoname}{%
13713   \renewenvironment{theglossary}%
13714     {\setlength{\parindent}{0pt}%
13715      \setlength{\parskip}{0pt plus 0.3pt}}%
13716     {}%
13717   \renewcommand*{\glossaryheader}{}%
13718   \renewcommand*{\glsgroupheading}[1]{}%
13719   \renewcommand{\glossentry}[2]{%
13720     \hangindent0pt\relax
13721     \parindent0pt\relax
13722     \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13723     \glstreenonamesymbol{##1}%
13724     \glstreenonamedesc{##1}%
13725     \glstreeprelocation##2\par
13726   }%
13727   \renewcommand{\subglossentry}[3]{%
13728     \hangindent##1\glstreeindent\relax
13729     \parindent##1\glstreeindent\relax
13730     \ifnum##1=1\relax
13731       \glssubentryitem{##2}%
13732     \fi
13733     \glstarget{##2}{\strut}%
13734     \glstreenonamechilddesc{##2}%
13735     \glstreechildprelocation##3\par
13736   }%
13737   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13738 }
13739 }
13740 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13741 \ifdef{\@glstyle@treenonamegroup}
13742 {%
13743   \renewglossarystyle{treenonamegroup}{%
13744     \setglossarystyle{treenoname}%
13745     \renewcommand{\glsgroupheading}[1]{\par
13746       \noindent\glstreegroupheaderfmt
13747       {\glsggetgrouptitle{##1}}}%
13748     \nopagebreak\indexspace\nobreak\@afterheading
13749   }%
13750 }
13751 }
13752 {}

```

Similarly for `treenonamehypergroup`

```

13753 \ifdef{\@glstyle@treenonamehypergroup}
13754 {%
13755   \renewglossarystyle{treenonamehypergroup}{%
13756     \setglossarystyle{treenoname}%
13757     \renewcommand*{\glossaryheader}{%

```



```

13758      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13759      \nobreak\@afterheading\indexspace}%
13760      \renewcommand*{\glsgroupheading}[1]{%
13761      \par\noindent
13762      \glstreegroupheaderfmt
13763      {\glsnavhypertarget{##1}{\glsgroupgetgrouptitle{##1}}}%
13764      \nopagebreak\indexspace\nobreak\@afterheading}%
13765  }
13766 }
13767 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13768 \ifdef{\@glstyle@almtree}
13769 {}

```

Only redefine this if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

13770 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13771   {%
13772     \let\par\glxtrAltTreePar
13773     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13774     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13775   }%
13776 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13777 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13778 \newcommand{\glxtrAltTreePar}{%
13779   \@par
13780   \glxtrAltTreeSetHangIndent
13781   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13782 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13783 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13784   \glxtralttreeSymbolDescLocation{#2}{#3}%
13785 }

```

trtreetopindent The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```
13786 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13787 \newcommand*{\glxtralttreeInit}{%
13788   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
13789   \glxstrAltTreeIndent=\parindent
13790 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13791 \newcommand*{\gglsetwidest}[2][0]{%
13792   \csgdef{@glswidestname\romannumeral#1}{#2}%
13793 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13794 \newcommand*{\eglssetwidest}[2][0]{%
13795   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13796 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13797 \newcommand*{\xglsetwidest}[2][0]{%
13798   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13799 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13800 \newcommand*{\glupdatewidest}[2][0]{%
13801   \ifcsundef{@glswidestname\romannumeral#1}%
13802   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13803   {%
13804     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13805     \settowidth{\dimen@ii}{#2}%
13806     \ifdim\dimen@ii>\dimen@
13807       \csdef{@glswidestname\romannumeral#1}{#2}%
13808     \fi
13809   }%
13810 }
```

glupdatewidest As above but global definition.

```
13811 \newcommand*{\gglupdatewidest}[2][0]{%
13812   \ifcsundef{@glswidestname\romannumeral#1}%
13813   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13814   {%
13815     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13816     \settowidth{\dimen@ii}{#2}%
13817     \ifdim\dimen@ii>\dimen@
13818       \csgdef{@glswidestname\romannumeral#1}{#2}%
13819     \fi
```

```

13820 }%
13821 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13822 \newcommand*{\eglsupdatewidest}[2][0]{%
13823   \ifcsundef{@glswidestname\romannumeral#1}%
13824     {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13825     {%
13826       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13827       \settowidth{\dimen@ii}{#2}%
13828       \ifdim\dimen@ii>\dimen0
13829         \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13830       \fi
13831     }%
13832 }

```

`glsupdatewidest` As above but global.

```

13833 \newcommand*{\xglsupdatewidest}[2][0]{%
13834   \ifcsundef{@glswidestname\romannumeral#1}%
13835     {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13836     {%
13837       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13838       \settowidth{\dimen@ii}{#2}%
13839       \ifdim\dimen@ii>\dimen0
13840         \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13841       \fi
13842     }%
13843 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13844 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glsgetwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13845 \newcommand*{\glsgetwidestsubname}[1]{%
13846   \ifcsundef{@glswidestname\romannumeral#1}%
13847     {\@glswidestname}%
13848     {\csuse{@glswidestname\romannumeral#1}}%
13849 }

```

`glsfindwidesttoplevelname` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`.

```

13850 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname

```

`glsfindwidestusedtoplevelname` Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```

13851 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
13852   \dimen@=0pt\relax

```

```

13853 \gls@tmplen=Opt\relax
13854 \foralllglossaries[#1]{\@gls@type}%
13855 {%
13856   \forglsentries[\@gls@type]{\@glo@label}%
13857   {%
13858     \ifglsused{\@glo@label}%
13859     {%
13860       \ifglshasparent{\@glo@label}%
13861       {}%
13862       {%
13863         \settowidth{\dimen@}%
13864           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13865         \ifdim\dimen@>\gls@tmplen
13866           \gls@tmplen=\dimen@
13867           \eglssetwidest{\glsentryname{\@glo@label}}%
13868         \fi
13869       }%
13870     }%
13871   }%
13872 }%
13873 }%
13874 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13875 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13876   \dimen@=Opt\relax
13877   \gls@tmplen=Opt\relax
13878   \foralllglossaries[#1]{\@gls@type}%
13879   {%
13880     \forglsentries[\@gls@type]{\@glo@label}%
13881     {%
13882       \ifglsused{\@glo@label}%
13883       {%
13884         \settowidth{\dimen@}%
13885           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13886         \ifdim\dimen@>\gls@tmplen
13887           \gls@tmplen=\dimen@
13888           \eglssetwidest{\glsentryname{\@glo@label}}%
13889         \fi
13890       }%
13891     }%
13892   }%
13893 }%
13894 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

13895 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13896   \dimen@=Opt\relax

```

```

13897 \gls@tmplen=0pt\relax
13898 \foralllglossaries[#1]{\@gls@type}%
13899 {%
13900 \forglsentries[\@gls@type]{\@glo@label}%
13901 {%
13902 \settowidth{\dimen@}%
13903 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13904 \ifdim\dimen@>\gls@tmplen
13905 \gls@tmplen=\dimen@
13906 \eglssetwidest{\glsentryname{\@glo@label}}%
13907 \fi
13908 }%
13909 }%
13910 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

13911 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13912 \dimen@=0pt\relax
13913 \dimen@i=0pt\relax
13914 \dimen@ii=0pt\relax
13915 \foralllglossaries[#1]{\@gls@type}%
13916 {%
13917 \forglsentries[\@gls@type]{\@glo@label}%
13918 {%
13919 \ifglsused{\@glo@label}%
13920 {%
13921 \ifglshasparent{\@glo@label}%
13922 {%
13923 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
13924 \ifglshasparent{\@glo@parent}%
13925 {%
13926 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
13927 \ifglshasparent{\@glo@parent}%
13928 }%
13929 {%
13930 \settowidth{\gls@tmplen}%
13931 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13932 \ifdim\gls@tmplen>\dimen@ii
13933 \dimen@ii=\gls@tmplen
13934 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13935 \fi
13936 }%
13937 }%
13938 {%
13939 \settowidth{\gls@tmplen}%
13940 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13941 \ifdim\gls@tmplen>\dimen@i
13942 \dimen@i=\gls@tmplen

```

```

13943         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13944     \fi
13945 }%
13946 }%
13947 {%
13948     \settowidth{\gls@tmplen}%
13949         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13950     \ifdim\gls@tmplen>\dimen@
13951         \dimen@=\gls@tmplen
13952         \eglssetwidest{\glsentryname{\@glo@label}}%
13953     \fi
13954 }%
13955 }%
13956 {}%
13957 }%
13958 }%
13959 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13960 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13961     \dimen@=0pt\relax
13962     \dimen@i=0pt\relax
13963     \dimen@ii=0pt\relax
13964     \foralllglossaries[#1]{\@gls@type}%
13965     {%
13966         \forglsentries[\@gls@type]{\@glo@label}%
13967         {%
13968             \ifglshasparent{\@glo@label}%
13969             {%
13970                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13971                 \ifglshasparent{\@glo@parent}%
13972                 {%
13973                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13974                     \ifglshasparent{\@glo@parent}%
13975                     {}%
13976                 }%
13977                 \settowidth{\gls@tmplen}%
13978                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13979                 \ifdim\gls@tmplen>\dimen@ii
13980                     \dimen@ii=\gls@tmplen
13981                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13982                 \fi
13983             }%
13984         }%
13985     }%
13986     \settowidth{\gls@tmplen}%
13987         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13988     \ifdim\gls@tmplen>\dimen@i
13989         \dimen@i=\gls@tmplen

```

```

13990         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13991     \fi
13992 }%
13993 }%
13994 {%
13995     \settowidth{\gls@tmplen}%
13996         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13997     \ifdim\gls@tmplen>\dimen@
13998         \dimen@=\gls@tmplen
13999         \eglssetwidest{\glsentryname{\@glo@label}}%
14000     \fi
14001 }%
14002 }%
14003 }%
14004 }

```

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.

```

14005 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
14006     \dimen@=0pt\relax
14007     \gls@tmplen=0pt\relax
14008     #2=0pt\relax
14009     \foralllglossaries[#1]{\@gls@type}%
14010     {%
14011         \forglsentries[\@gls@type]{\@glo@label}%
14012         {%
14013             \ifglsused{\@glo@label}%
14014             {%
14015                 \settowidth{\dimen@}%
14016                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14017                 \ifdim\dimen@>\gls@tmplen
14018                     \gls@tmplen=\dimen@
14019                     \eglssetwidest{\glsentryname{\@glo@label}}%
14020                 \fi
14021                 \settowidth{\dimen@}%
14022                     {\glsentrysymbol{\@glo@label}}%
14023                 \ifdim\dimen@>#2\relax
14024                     #2=\dimen@
14025                 \fi
14026             }%
14027         }%
14028     }%
14029 }%
14030 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

14031 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14032     \dimen@=0pt\relax
14033     \gls@tmplen=0pt\relax

```

```

14034 #2=Opt\relax
14035 \foralllglossaries[#1]{\@gls@type}%
14036 {%
14037   \forallglsentries[\@gls@type]{\@glo@label}%
14038   {%
14039     \settowidth{\dimen@}%
14040     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14041     \ifdim\dimen@>\gls@tmplen
14042       \gls@tmplen=\dimen@
14043       \eglssetwidest{\glsentryname{\@glo@label}}%
14044     \fi
14045     \settowidth{\dimen@}%
14046     {\glsentrysymbol{\@glo@label}}%
14047     \ifdim\dimen@>#2\relax
14048       #2=\dimen@
14049     \fi
14050   }%
14051 }%
14052 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but also measures the location list. This requires `\glsentrynumberlist`. The length of the widest symbol is stored in the second argument should be a length register. The length of the widest location list is stored in the third argument, which should also be a length register.

```

14053 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14054   \dimen@=Opt\relax
14055   \gls@tmplen=Opt\relax
14056   #2=Opt\relax
14057   #3=Opt\relax
14058   \foralllglossaries[#1]{\@gls@type}%
14059   {%
14060     \forallglsentries[\@gls@type]{\@glo@label}%
14061     {%
14062       \ifglsused{\@glo@label}%
14063       {%
14064         \settowidth{\dimen@}%
14065         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14066         \ifdim\dimen@>\gls@tmplen
14067           \gls@tmplen=\dimen@
14068           \eglssetwidest{\glsentryname{\@glo@label}}%
14069         \fi
14070         \settowidth{\dimen@}%
14071         {\glsentrysymbol{\@glo@label}}%
14072         \ifdim\dimen@>#2\relax
14073           #2=\dimen@
14074         \fi
14075         \settowidth{\dimen@}%
14076         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14077         \ifdim\dimen@>#3\relax

```



```

14078         #3=\dimen@
14079         \fi
14080     }%
14081     {}%
14082 }%
14083 }%
14084 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14085 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14086     \dimen@=0pt\relax
14087     \gls@tmplen=0pt\relax
14088     #2=0pt\relax
14089     #3=0pt\relax
14090     \forallglossaries[#1]{\@gls@type}%
14091     {%
14092         \forglsentries[\@gls@type]{\@glo@label}%
14093         {%
14094             \settowidth{\dimen@}%
14095                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14096             \ifdim\dimen@>\gls@tmplen
14097                 \gls@tmplen=\dimen@
14098                 \eglssetwidest{\glsentryname{\@glo@label}}%
14099             \fi
14100             \settowidth{\dimen@}%
14101                 {\glsentrysymbol{\@glo@label}}%
14102             \ifdim\dimen@>#2\relax
14103                 #2=\dimen@
14104             \fi
14105             \settowidth{\dimen@}%
14106                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14107             \ifdim\dimen@>#3\relax
14108                 #3=\dimen@
14109             \fi
14110         }%
14111     }%
14112 }

```

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.

```

14113 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14114     \dimen@=0pt\relax
14115     \gls@tmplen=0pt\relax
14116     #2=0pt\relax
14117     \forallglossaries[#1]{\@gls@type}%
14118     {%
14119         \forglsentries[\@gls@type]{\@glo@label}%
14120         {%

```

```

14121     \ifglsused{\@glo@label}%
14122     {%
14123         \settowidth{\dimen@}%
14124         {\glstreenamfmt{\glentryname{\@glo@label}}}%
14125         \ifdim\dimen@>\gls@tmplen
14126             \gls@tmplen=\dimen@
14127             \eglssetwidest{\glentryname{\@glo@label}}%
14128         \fi
14129         \settowidth{\dimen@}%
14130         {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14131         \ifdim\dimen@>#2\relax
14132             #2=\dimen@
14133         \fi
14134     }%
14135     {}%
14136 }%
14137 }%
14138 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14139 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14140     \dimen@=0pt\relax
14141     \gls@tmplen=0pt\relax
14142     #2=0pt\relax
14143     \forallglossaries[#1]{\@gls@type}%
14144     {%
14145         \forglsentries[\@gls@type]{\@glo@label}%
14146         {%
14147             \settowidth{\dimen@}%
14148             {\glstreenamfmt{\glentryname{\@glo@label}}}%
14149             \ifdim\dimen@>\gls@tmplen
14150                 \gls@tmplen=\dimen@
14151                 \eglssetwidest{\glentryname{\@glo@label}}%
14152             \fi
14153             \settowidth{\dimen@}%
14154             {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14155             \ifdim\dimen@>#2\relax
14156                 #2=\dimen@
14157             \fi
14158         }%
14159     }%
14160 }

```

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`.

```

14161 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
14162     \glstreeindent=\glstxtrtreetopindent\relax
14163 }

```

uteTreeSubIndent

```
\glxstrComputeTreeSubIndent{<level>}{<label>}{<register>}
```

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
14164 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14165 \ifcsundef{@glswidestname\romannumeral#1}%
14166 {%
14167 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14168 }%
14169 {%
14170 \settowidth{#3}{\glstreenamefmt{
14171 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14172 }%
14173 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14174 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14175 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14176 \renewglossarystyle{alttree}{%
14177 \renewenvironment{theglossary}%
14178 {%
14179 \glxstralttreeInit
14180 \def\@gls@prevlevel{-1}%
14181 \mbox{}\par}%
14182 {\par}%
14183 \renewcommand*{\glossaryheader}{}%
14184 \renewcommand*{\glsgroupheading}[1]{}%
14185 \renewcommand{\glossentry}[2]{%
14186 \ifnum\@gls@prevlevel=0\relax
14187 \else
14188 \glxstrComputeTreeIndent{##1}%
14189 \fi
14190 \parindent\glstreeindent
14191 \glxstrAltTreeSetHangIndent
14192 \makebox[Opt][r]%
14193 {%
14194 \glstreenamebox{\glstreeindent}%
14195 {%
14196 \glsentryitem{##1}%
14197 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14198 }%
14199 }%
```

```

14200 \glxtralttreeSymbolDescLocation{##1}{##2}%
14201 \def\@gls@prevlevel{0}%
14202 }
14203 \renewcommand{\subglossentry}[3]{%
14204 \ifnum##1=1\relax
14205 \glssubentryitem{##2}%
14206 \fi
14207 \ifnum\@gls@prevlevel=##1\relax
14208 \else
14209 \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14210 \ifnum\@gls@prevlevel<##1\relax
14211 \setlength\glstreeindent\gls@tmplen
14212 \addtolength\glstreeindent\parindent
14213 \parindent\glstreeindent
14214 \else
14215 \ifnum\@gls@prevlevel=0\relax
14216 \glxtrComputeTreeIndent{##2}%
14217 \else
14218 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14219 \fi
14220 \addtolength\parindent{-\glstreeindent}%
14221 \setlength\glstreeindent\parindent
14222 \fi
14223 \fi
14224 \glxtrAltTreeSetSubHangIndent{##1}%
14225 \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14226 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14227 \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14228 \def\@gls@prevlevel{##1}%
14229 }%
14230 \renewcommand*\@glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
14231 }
14232 }%
14233 {%
14234 }

```

Redefine alttreegroup so that it discourages a break after group headings. Can't use \@afterheading here as it messes with the first item of the group.

```

14235 \ifdef{\@glsstyle@alttreegroup}
14236 {%
14237 \renewglossarystyle{alttreegroup}{%
14238 \setglossarystyle{alttree}%
14239 \renewcommand{\glsgroupheading}[1]{\par
14240 \def\@gls@prevlevel{-1}%
14241 \hangindent0pt\relax
14242 \parindent0pt\relax
14243 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14244 \nopagebreak\indexspace\nopagebreak
14245 }%
14246 }%

```

```

14247 }%
14248 {%
14249 }

```

Similarly for `alttreehypergroup`.

```

14250 \ifdef{\@glsstyle@alttreehypergroup}
14251 {%
14252   \renewglossarystyle{alttreehypergroup}{%
14253     \setglossarystyle{alttree}%
14254     \renewcommand*{\glossaryheader}{%
14255       \par
14256       \def\@gls@prevlevel{-1}%
14257       \hangindent0pt\relax
14258       \parindent0pt\relax
14259       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14260     }%
14261     \renewcommand*{\glsgroupheading}[1]{%
14262       \par
14263       \def\@gls@prevlevel{-1}%
14264       \hangindent0pt\relax
14265       \parindent0pt\relax
14266       \glstreegroupheaderfmt
14267       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14268       \nopagebreak\indexspace\nopagebreak
14269     }%
14270   }
14271 }%
14272 {%
14273 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14274 \ifdef{\@glsstyle@mcolindexgroup}
14275 {%
14276   \renewglossarystyle{mcolindexgroup}{%
14277     \setglossarystyle{mcolindex}%
14278     \renewcommand*{\glsgroupheading}[1]{%
14279       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14280       \nopagebreak\indexspace\nobreak\@afterheading
14281     }%
14282   }
14283 }%
14284 {%
14285 }

```

Similarly for `mcolindexhypergroup`.

```

14286 \ifdef{\@glsstyle@mcolindexhypergroup}
14287 {%

```

```

14288 \renewglossarystyle{mcolindexhypergroup}{%
14289   \setglossarystyle{mcolindex}%
14290   \renewcommand*{\glossaryheader}{%
14291     \item\glstreenavigationfmt{\glsnavigation}%
14292     \indexspace
14293   }%
14294   \renewcommand*{\glsgroupheading}[1]{%
14295     \item\glstreegroupheaderfmt
14296       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14297     \nopagebreak\indexspace\nobreak\@afterheading
14298   }%
14299 }
14300}%
14301{%
14302}

```

Similarly for mcolindexspannav.

```

14303 \ifdef{\@glsstyle@mcolindexspannav}
14304 {%
14305   \renewglossarystyle{mcolindexspannav}{%
14306     \setglossarystyle{index}%
14307     \renewenvironment{theglossary}%
14308     {%
14309       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14310       \setlength{\parindent}{0pt}%
14311       \setlength{\parskip}{0pt plus 0.3pt}%
14312       \let\item\glstreeitem}%
14313     {\end{multicols}}}%
14314     \renewcommand*{\glsgroupheading}[1]{%
14315       \item\glstreegroupheaderfmt
14316         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14317       \nopagebreak\indexspace\nobreak\@afterheading
14318     }%
14319   }
14320}%
14321{%
14322}

```

Similarly for mcoltreegroup.

```

14323 \ifdef{\@glsstyle@mcoltreegroup}
14324 {%
14325   \renewglossarystyle{mcoltreegroup}{%
14326     \setglossarystyle{mcoltree}%
14327     \renewcommand{\glsgroupheading}[1]{\par
14328       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14329     \nopagebreak\indexspace\nobreak\@afterheading
14330   }%
14331 }
14332}%
14333{%

```

14334 }

Similarly for mcoltreehypergroup.

```
14335 \ifdef{\@glsstyle@mcoltreehypergroup}
14336 {%
14337   \renewglossarystyle{mcoltreehypergroup}{%
14338     \setglossarystyle{mcoltree}%
14339     \renewcommand*{\glossaryheader}{%
14340       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14341     }%
14342     \renewcommand*{\glsgroupheading}[1]{%
14343       \par\noindent
14344       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14345       \nopagebreak\indexspace\nobreak\@afterheading
14346     }%
14347   }
14348 }%
14349 {%
14350 }
```

Similarly for mcoltreespannav.

```
14351 \ifdef{\@glsstyle@mcoltreespannav}
14352 {%
14353   \renewglossarystyle{mcoltreespannav}{%
14354     \setglossarystyle{tree}%
14355     \renewenvironment{theglossary}%
14356     {%
14357       \begin{multicols}{\glsmcols}%
14358       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14359       \setlength{\parindent}{0pt}%
14360       \setlength{\parskip}{0pt plus 0.3pt}%
14361     }%
14362     {\end{multicols}}%
14363     \renewcommand*{\glsgroupheading}[1]{%
14364       \par\noindent
14365       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14366       \nopagebreak\indexspace\nobreak\@afterheading
14367     }%
14368   }
14369 }%
14370 {%
14371 }
```

Similarly for mcoltreenonamegroup.

```
14372 \ifdef{\@glsstyle@mcoltreenonamegroup}
14373 {%
14374   \renewglossarystyle{mcoltreenonamegroup}{%
14375     \setglossarystyle{mcoltreenoname}%
14376     \renewcommand{\glsgroupheading}[1]{\par
14377       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14378     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14379 }%
14380 }
14381 }%
14382 {%
14383 }

```

Similarly for mcoltreenonamehypergroup.

```

14384 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14385 {%
14386   \renewglossarystyle{mcoltreenonamehypergroup}{%
14387     \setglossarystyle{mcoltreenoname}%
14388     \renewcommand*{\glossaryheader}{%
14389       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14390     \renewcommand*{\glsgroupheading}[1]{%
14391       \par\noindent
14392       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14393       \nopagebreak\indexspace\nobreak\@afterheading}%
14394   }
14395 }%
14396 {%
14397 }

```

Similarly for mcoltreenonamespannav.

```

14398 \ifdef{\@glsstyle@mcoltreenonamespannav}
14399 {%
14400   \renewglossarystyle{mcoltreenonamespannav}{%
14401     \setglossarystyle{treenoname}%
14402     \renewenvironment{theglossary}%
14403     {%
14404       \begin{multicols}{\glsmcols}%
14405       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14406       \setlength{\parindent}{0pt}%
14407       \setlength{\parskip}{0pt plus 0.3pt}%
14408     }%
14409     {\end{multicols}}}%
14410   \renewcommand*{\glsgroupheading}[1]{%
14411     \par\noindent
14412     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14413     \nopagebreak\indexspace\nobreak\@afterheading}%
14414   }
14415 }%
14416 {%
14417 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{\par which would unbalance the top of the columns.

```

14418 \ifdef{\@glsstyle@mcolalttree}
14419 {%
14420   \renewglossarystyle{mcolalttree}{%
14421     \setglossarystyle{alttree}%
14422     \renewenvironment{theglossary}%

```



```

14423   {%
14424       \glstralttreeInit
14425       \def\@gls@prevlevel{-1}%
14426       \begin{multicols}{\glsmcols}%
14427   }%
14428   {\par\end{multicols}}%
14429   }
14430 }%
14431 {%
14432 }

```

Redefine mcolalttreegroup to discourage page breaks after the group headings.

```

14433 \ifdef{\@glsstyle@mcolalttreegroup}
14434 {%
14435   \renewglossarystyle{mcolalttreegroup}{%
14436       \setglossarystyle{mcolalttree}%
14437       \renewcommand{\glsgroupheading}[1]{\par
14438           \def\@gls@prevlevel{-1}%
14439           \hangindent0pt\relax
14440           \parindent0pt\relax
14441           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14442       \nopagebreak\indexspace\nopagebreak
14443   }%
14444   }
14445 }%
14446 {%
14447 }

```

Similarly for mcolalttreehypergroup.

```

14448 \ifdef{\@glsstyle@mcolalttreehypergroup}
14449 {%
14450   \renewglossarystyle{mcolalttreehypergroup}{%
14451       \setglossarystyle{mcolalttree}%
14452       \renewcommand*\{\glossaryheader}{%
14453           \par
14454           \def\@gls@prevlevel{-1}%
14455           \hangindent0pt\relax
14456           \parindent0pt\relax
14457           \glstreenavigationfmt{\glsnavigation}%
14458           \par\indexspace
14459       }%
14460       \renewcommand*\{\glsgroupheading}[1]{%
14461           \par
14462           \def\@gls@prevlevel{-1}%
14463           \hangindent0pt\relax
14464           \parindent0pt\relax
14465           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14466           \nopagebreak\indexspace\nopagebreak
14467       }%
14468   }

```

```

14469 }%
14470 {%
14471 }

```

Similarly for mcolalttreespannav.

```

14472 \ifdef{\@glsstyle@mcolalttreespannav}
14473 {%
14474   \renewglossarystyle{mcolalttreespannav}{%
14475     \setglossarystyle{alttree}%
14476     \renewenvironment{theglossary}%
14477     {%
14478       \glstralttreeInit
14479       \def\@gls@prevlevel{-1}%
14480       \begin{multicols}{\@gls@col}%
14481         [\noindent\glstreenavigationfmt{\@gls@navigation}]%
14482     }%
14483     {\par\end{multicols}}%
14484     \renewcommand*\@gls@groupheading[1]{%
14485       \par
14486       \def\@gls@prevlevel{-1}%
14487       \hangindent0pt\relax
14488       \parindent0pt\relax
14489       \glstreegroupheaderfmt{\@gls@navhypertarget{##1}{\@gls@getgrouptitle{##1}}}%
14490       \nopagebreak\indexspace\nopagebreak
14491     }%
14492   }
14493 }%
14494 {%
14495 }

```

Reset the default style

```

14496 \ifx\@glossary@default@style\relax
14497 \else
14498   \setglossarystyle{\@gls@current@style}
14499 \fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
14500 \NeedsTeXFormat{LaTeX2e}
14501 \ProvidesPackage{glossary-bookindex}[2018/08/18 v1.36 (NLCT)]
```

Load required packages.

```
14502 \RequirePackage{multicol}
14503 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14504 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14505 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14506 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14507 \providecommand*{\glstrprelocation}{\space}
```

`indexprelocation` Separator used before location list for top-level entries. Version 1.22 has removed the `\ifglsnopostdot` check since this style doesn't display the description.

```
14508 \newcommand*{\glstrbookindexprelocation}[1]{%
14509   \glstrifhasfield{location}{#1}%
14510   {,\glstrprelocation}%
14511   {\glstrprelocation}%
14512 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14513 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14514   \glstrbookindexprelocation{#1}%
14515 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14516 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14517 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
14518 \newcommand{\glxstrbookindexbetween}[2]{}

indexsubbetween Between two level 1 entries identified by the labels in the arguments.
14519 \newcommand{\glxstrbookindexsubbetween}[2]{}

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.
14520 \newcommand{\glxstrbookindexsubsubbetween}[2]{}

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.
14521 \newcommand{\glxstrbookindexatendgroup}[1]{}

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.
14522 \newcommand{\glxstrbookindexsubatendgroup}[1]{}

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.
14523 \newcommand{\glxstrbookindexsubsubatendgroup}[1]{}

kindexgroupskip Group separator.
14524 \newcommand{\glxstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

Format group title.

dexformatheader Group separator.
14525 \newcommand*{\glxstrbookindexformatheader}[1]{%
14526 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14527 }

okindexbookmark Book mark group heading if supported.
14528 \ifdef\pdfbookmark
14529 {%
14530 \newcommand*{\glxstrbookindexbookmark}[2]{%
14531 \ifdefstring{\@@glossarysec}{chapter}%
14532 {\pdfbookmark[1]{#1}{#2}}%
14533 {\pdfbookmark[2]{#1}{#2}}%
14534 }
14535 }
14536 {%
14537 \newcommand*{\glxstrbookindexbookmark}[2]{}
14538 }

kindexcolspread
14539 \newcommand*{\glxstrbookindexcolspread}{}

dexmulticolenv
14540 \newcommand*{\glxstrbookindexmulticolenv}{multicols}
```

Define the style.

```

14541 \newglossarystyle{bookindex}{%
14542   \setglossarystyle{index}%
14543   \renewenvironment{theglossary}%
14544   {%
14545     \ifdefempty{glxstrbookindexcolspread}
14546     {%
14547       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14548       {\glxstrbookindexcols}%
14549     }%
14550     {%
14551       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14552       {\glxstrbookindexcols}[\glxstrbookindexcolspread]%
14553     }%
14554     \setlength{\parindent}{0pt}%
14555     \setlength{\parskip}{0pt plus 0.3pt}%
14556     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14557     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14558     \let\@glxstr@bookindex@between\@gobble
14559     \let\@glxstr@bookindex@subbetween\@gobble
14560     \let\@glxstr@bookindex@subsubbetween\@gobble
14561     \let\@glxstr@bookindex@atendgroup\relax
14562     \let\@glxstr@bookindex@subatendgroup\relax
14563     \let\@glxstr@bookindex@subsubatendgroup\relax
14564     \let\@glxstr@bookindex@groupskip\relax
14565   }%
14566   {%

```

Do end group hooks.

```

14567     \@glxstr@bookindex@subsubatendgroup
14568     \@glxstr@bookindex@subatendgroup
14569     \@glxstr@bookindex@atendgroup

```

End multicol environment.

```

14570     \expandafter\end\expandafter{\glxstrbookindexmulticolseenv}%
14571   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

14572   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

14573   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

14574     \@glxstr@bookindex@between{##1}%

```

Update separators.

```

14575     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14576     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14577     \let\@glxstr@bookindex@subbetween\@gobble
14578     \let\@glxstr@bookindex@subsubbetween\@gobble
14579     \edef\@glxstr@bookindex@between{%

```

```

14580     \noexpand\glxstrbookindexbetween{##1}%
14581 }%
14582 \edef\@glxstr@bookindex@atendgroup{%
14583     \noexpand\glxstrbookindexatendgroup{##1}%
14584 }%
14585 \let\@glxstr@bookindex@subatendgroup\relax
14586 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14587 \glstreeitem
14588     \glstryitem{##1}%
14589     \glstarget{##1}{\glxstrbookindexname{##1}}%
14590 \glxstrbookindexprelocation{##1}##2%
14591 }%
14592 \renewcommand{\subglossentry}[3]{%
14593     \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14594     \glstreeitem
14595     \or

```

Level 1.

```

14596     \@glxstr@bookindex@sep
14597     \@glxstr@bookindex@subbetween{##2}%
14598     \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14599     \let\@glxstr@bookindex@subsubbetween\@gobble
14600     \let\@glxstr@bookindex@subsep\glxstrbookindexparentsubchildsep
14601     \edef\@glxstr@bookindex@subbetween{%
14602         \noexpand\glxstrbookindexsubbetween{##2}%
14603     }%
14604     \edef\@glxstr@bookindex@atsubendgroup{%
14605         \noexpand\glxstrbookindexatsubendgroup{##1}%
14606     }%

```

Start sub-item.

```

14607     \glstreesubitem
14608     \glssubentryitem{##2}%
14609     \else

```

All other levels.

```

14610     \@glxstr@bookindex@subsep
14611     \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14612     \let\@glxstr@bookindex@subsep\relax
14613     \edef\@glxstr@bookindex@subsubbetween{%
14614         \noexpand\glxstrbookindexsubsubbetween{##2}%
14615     }%
14616     \edef\@glxstr@bookindex@atsubsubendgroup{%
14617         \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14618     }%

```

Start sub-sub-item.

```
14619 \glstreesubsubitem
14620 \fi
```

Format entry.

```
14621 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14622 \glxtrbookindexsubprelocation{##2}##3%
14623 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14624 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14625 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14626 \@glxtr@bookindex@subsubatendgroup
14627 \@glxtr@bookindex@subatendgroup
14628 \@glxtr@bookindex@atendgroup
14629 \@glxtr@bookindexgroupskip
```

Update separators.

```
14630 \let\@glxtr@bookindexgroupskip\glxtrbookindexgroupskip
14631 \let\@glxtr@bookindex@between\@gobble
14632 \let\@glxtr@bookindex@atendgroup\relax
14633 \let\@glxtr@bookindex@subatendgroup\relax
14634 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14635 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14636 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14637 \glxtrbookindexformatheader{\thisgrptitle}%
14638 \nopagebreak\indexspace\nopagebreak\@afterheading
14639 }%
14640 }
```

Some supplementary commands that may be useful. These store the entry label for the current page. Since the page number is needed in the control sequence, this uses `\glxtrbookindexthepage` instead of `\thepage` in case the page numbering has been set to something that contains formatting commands.

`\glxtrbookindexthepage` The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```
14641 \newcommand{\glxtrbookindexthepage}{%
14642 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}}%
14643 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14644 \newcommand*\glxtrbookindexmarkentry}[1]{%
14645   \protected@write\@auxout
14646   {\let\glxtrbookindexthepage\relax}%
14647   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14648 }
```

etbookindexmark

```
14649 \newcommand*\glxtr@setbookindexmark}[2]{%
14650   \ifcsundef\glxtr@idxfirstmark@#1}%
14651   {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
14652   {}%
14653   \csgdef\glxtr@idxlastmark@#1}{#2}%
14654 }
```

indexfirstmarkfmt

```
14655 \newcommand*\glxtrbookindexfirstmarkfmt}[1]{%
14656   \glxtr@entryname{#1}%
14657 }
```

indexfirstmark

```
14658 \newcommand*\glxtrbookindexfirstmark{%
14659   \letcs\glxtr@label\glxtr@idxfirstmark@\glxtrbookindexthepage}%
14660   \ifdef\glxtr@label
14661   {\glxtrbookindexfirstmarkfmt\glxtr@label}}%
14662   {}%
14663 }
```

indexlastmarkfmt

```
14664 \newcommand*\glxtrbookindexlastmarkfmt}[1]{%
14665   \glxtr@entryname{#1}%
14666 }
```

okindexlastmark

```
14667 \newcommand*\glxtrbookindexlastmark{%
14668   \letcs\glxtr@label\glxtr@idxlastmark@\glxtrbookindexthepage}%
14669   \ifdef\glxtr@label
14670   {\glxtrbookindexlastmarkfmt\glxtr@label}}%
14671   {}%
14672 }
```


Glossary

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

First use flag A conditional that determines whether or not the entry has been used according to the rules of **first use**.

First use text The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.

Change History

0.1 (2015-11-22)		\@Glsymbol@: added redefinition	74
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	74
\Glsfmtshort: new	324	\@Glstext@: added redefinition	70
\glsfmtshort: new	323	\@Glsuseri@: added redefinition	75
\Glsfmtshortpl: new	324	\@Glsuserii@: added redefinition	75
\glsfmtshortpl: new	324	\@Glsuseriii@: added redefinition	75
short: switched inline full form to short		\@Glsuseriv@: added redefinition	75
(long)	227	\@Glsuseriv@: added redefinition	76
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	76
\@ACRlong: added redefinition	79	\@acrlong: added redefinition	79
\@ACRlongpl: added redefinition	80	\@acrlongpl: added redefinition	80
\@ACRshort: added redefinition	77	\@acrshort: added redefinition	76
\@ACRshortpl: added redefinition	78	\@acrshortpl: added redefinition	77
\@Acrlong: added redefinition	79	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	80	argument	62
\@Acrshort: added redefinition	77	\@glsdescplural@: added redefinition	73
\@Acrshortpl: added redefinition	78	\@glsfirst@: added redefinition	70
\@GLSdesc@: added redefinition	73	\@glsfirstplural@: added redefinition	71
\@GLSdescplural@: added redefinition	73	\@glsplural@: added redefinition	71
\@GLSfirst@: added redefinition	70	\@glssymbolplural@: added	
\@GLSfirstplural@: added redefinition	72	redefinition	74
\@GLSname@: added redefinition	72	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	71	new	133
\@GLSSymbol@: added redefinition	74	\@glsxtr@field@linkdefs: new	69
\@GLSSymbolplural@: added		\@glsxtr@insertdots: new	195
redefinition	74	\@print@glossary: added redefinition	129
\@GLStext@: added redefinition	69	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	75	\abbrvdefaultfont	200
\@GLSuserii@: added redefinition	75	\glsaccessdesc: new	158
\@GLSuseriii@: added redefinition	75	\glsaccessdescplural: new	159
\@GLSuseriv@: added redefinition	76	\glsaccessfirst: new	156
\@GLSuserv@: added redefinition	76	\glsaccessfirstplural: new	156
\@GLSuservi@: added redefinition	76	\Glsaccesslong: new	161
\@Glsdesc@: added redefinition	73	\glsaccesslong: new	160
\@Glsdescplural@: added redefinition	73	\glsaccessname: new	154
\@Glsfirst@: added redefinition	70	\glsaccessplural: new	155
\@Glsfirstplural@: added redefinition	72	\Glsaccessshort: new	159
\@Glsname@: added redefinition	72	\glsaccessshort: new	159
\@Glsplural@: added redefinition	71	\Glsaccessshortpl: new	160

\glsaccesssshortpl: new	160	\cGLSpl: new	106
\glsaccessssymbol: new	157	\cGLSpl@: new	106
\glsaccessssymbolplural: new	157	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	155	new	101
\glstentryfmt: added check for short ..	61	\cGLS: new	106
\glslongpltok: new	194	\cGLSformat: new	106
\glsshortpltok: new	194	\cGLSpl: new	106
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	106
name in \setkeys	196	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	16
for plural	191	\glsenableentrycount: new	101
\GLSxtrlongpl: new	211	\glsfirstabbrvdefaultfont: new ..	200
\Glsxtrlongpl: new	210	\glsfirstlongdefaultfont: new ...	200
\glsxtrlongpl: new	210	\Glsfmtfirst: new	327
\glsxtrNoGlossaryWarning: new	21	\glsfmtfirst: new	326
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	327
new	191	\glsfmtfirstpl: new	327
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	326
new	191	\glsfmtplural: new	326
\glsxtrpostlinkendsentence: new ..	190	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	209	\Glsxtrtitleshort	324
\Glsxtrshortpl: new	208	renamed from \Glstentryfmtshort ..	324
\glsxtrshortpl: new	208	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	323
\glslabeltok	221	renamed from \glstentryfmtshort ..	323
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	219	\Glsxtrtitleshortpl	324
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	324
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	324
\Glsxtrshort	324	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	324
\glsxtrshort	323	\Glsfmttext: new	325
\Glsfmtshortpl: changed to use		\glsfmttext: new	325
\glsxtrshortpl	324	\glshasattribute: new	169
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	168
\glsxtrshortpl	324	\glsxtremsuffix: new	263
\glsxtrifemptyglossary: new	28	\GlsXtrEnableEntryCounting: new ..	101
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	103
argument	172	\glsxtrscfont: new	234
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	235
argument	172	\glsxtrsmfont: new	249
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	249
default type to \acronymtype	114	short-em: new	270
\newterm: fixed name argument	171	short-em-desc: new	271
0.5 (2015-12-07)		short-em-footnote: new	281
\cGLS: new	106	short-em-long: new	266
\cGLS@: new	106	short-em-long-desc: new	268

short-em-postfootnote: new	282	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	245	attribute	315
short-sc-postfootnote: new	247	\Glsxtrheadtext: now uses headuc	
short-sm: new	252	attribute	317
short-sm-desc: new	254	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	259	attribute	317
short-sm-long: new	251	short-em-footnote: switch off regular	
short-sm-long-desc: new	252	attribute if set	281
short-sm-postfootnote: new	261	short-long: switch off regular attribute	
long-noshort-em: new	273	if set	220
long-noshort-em-desc: new	277	short-long-desc: switch off regular	
long-noshort-sm: new	256	attribute if set	222
long-noshort-sm-desc: new	258	short-sc-footnote: switch off regular	
long-short-em: new	263	attribute if set	245
long-short-em-desc: new	264	short-sm-footnote: switch off regular	
long-short-sm: new	249	attribute if set	259
long-short-sm-desc: new	250	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	218
\Glsaccessstext: new	155	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	220
\@gls@setup@default@short@access:		long-short-sc-desc: switch off regular	
removed \ifglxtruseuchead ...	314	attribute if set	236
\@glxtr@doaccsupp: new	21	footnote: switch off regular attribute if	
\Glsaccessdesc: new	158	set	223
\Glsaccessdescplural: new	159	postfootnote: switch off regular	
\Glsaccessfirst: new	156	attribute if set	224
\Glsaccessfirstplural: new	157	0.5.2 (2015-12-08)	
\Glsaccessname: new	154	\@GLSdesc@: added accessibility support	73
\Glsaccessplural: new	155	\@GLSdescplural@: added accessibility	
\Glsaccesssymbol: new	157	support	73
\Glsaccesssymbolplural: new	158	\@GLSfirst@: added accessibility	
\Glsxtrheadfirst: now uses headuc		support	70
attribute	319	\@GLSfirstplural@: added accessibility	
\glxtrheadfirst: now uses headuc		support	72
attribute	318	\@GLSname@: added accessibility support	72
\Glsxtrheadfirstplural: now uses		\@GLSplural@: added accessibility	
headuc attribute	319	support	71
\glxtrheadfirstplural: now uses		\@GLSsymbol@: added accessibility	
headuc attribute	319	support	74
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute	318	accessibility support	74
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	69
attribute	317	\@GLSdesc@: added accessibility support	73
\Glsxtrheadshort: now uses headuc		\@GLSdescplural@: added accessibility	
attribute	315	support	73
\glxtrheadshort: now uses headuc		\@GLSfirst@: added accessibility	
attribute	314	support	70
\Glsxtrheadshortpl: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	315	support	72

\@Glsname@: add accessibility support ..	72	\glsdohypertarget: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	154
\@Glsplural@: added accessibility support	71	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here	61
\@Glsymbol@: added accessibility support	74	\GlsXtrEnableInitialTagging: new	187
\@Glsymbolplural@: added accessibility support	74	\glsxtrfieldtitlecase: new	173
\@Glstext@: added accessibility support	70	\GlsXtrFormatLocationList: new ...	59
\@gls@setup@default@short@access: removed \glsxtrabbrvfmt	212	\glsxtrnewabbrevpresetkeyhook: new	198
\@glsdesc@: added accessibility support	73	\glsxtrtagfont: new	188
\@glsdescplural@: added accessibility support	73	\KV@printgloss@nonumberlist: added	61
\@glsfirst@: added accessibility support	70	\mfu@checkword@do: added	187
\@glsfirstplural@: added accessibility support	71	\setabbreviationstyle: added check for post-definition style switch	214
\@Glsname@: added accessibility support	72	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	71	\@glsxtr@autoindex@at: new	184
\@Glsymbol@: added accessibility support	74	\@glsxtr@autoindex@encap: new ...	184
\@Glsymbolplural@: added accessibility support	74	\@glsxtr@autoindex@esc: new	185
\@Glstext@: added accessibility support	69	\@glsxtr@autoindex@level: new ...	184
\@glsxtr@activate@initialtagging: new	188	\@glsxtr@autoindex@setname: new .	182
\@glsxtr@do@titlecaps@warn: new .	188	\@glsxtr@doabbreviationsdef: new .	17
\@glsxtr@tag: new	188	\glsdescwidth: added	58
\glossaryentrynumbers: added	59	\glsdohypertarget: removed \GlsXtrNoGlsWarningNoAutoMakeMain	132
\Glossentrydesc: added	186	\glspagelistwidth: added	58
\Glossentryname: added	178	\glsxtrdoautoindexname: new	182
\Glossentrysymbol: added	186	\glsxtrpostnamehook: new	179
\glossentrysymbol: added	186	\if@glsxtr@format@override: new .	181
\GLSaccessdesc: new	158, 166	\ProvidesGlossariesExtraLang: new	330
\GLSaccessdescplural: new ...	159, 166	\RequireGlossariesExtraLang: new	330
\GLSaccessfirst: new	156, 165	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	157, 165	\@newglossaryentry@defunitcounters: new	107
\GLSaccesslong: new	161, 167	\@GLSxtr@p@acrlong@: new	92
\GLSaccesslongpl: new	161, 167	\@GLSxtr@p@acrlongpl@: new	93
\Glsaccesslongpl: new	161	\@GLSxtr@p@acrshort@: new	92
\glsaccesslongpl: new	161	\@GLSxtr@p@acrshortpl@: new	92
\GLSaccessname: new	154, 164	\@GLSxtr@p@long@: new	91
\GLSaccessplural: new	156, 165	\@GLSxtr@p@longpl@: new	92
\GLSaccessshort: new	160, 167	\@GLSxtr@p@plural@: new	90
\GLSaccessshortpl: new	160, 167	\@GLSxtr@p@short@: new	91
\GLSaccesssymbol: new	157, 166	\@GLSxtr@p@shortpl@: new	91
\GLSaccesssymbolplural: new .	158, 166	\@GLSxtr@p@text@: new	90
\GLSaccessstext: new	155, 165	\@GlsXtrEnableOnTheFly: new	54
		\@Glsxtr: new	55
		\@Glsxtr@p@acrlong@: new	92

\@Glsxtr@p@acrlongpl@: new	92	\glstdonohyperlink: added	89
\@Glsxtr@p@acrshort@: new	92	\glsenableentryunitcount: new	109
\@Glsxtr@p@acrshortpl@: new	92	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	91	entry's existence	169
\@Glsxtr@p@longpl@: new	92	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	90	entry's existence	169
\@Glsxtr@p@short@: new	90	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	91	check	190
\@Glsxtr@p@text@: new	90	\Glsxtr: new	55
\@Glsxtrpl: new	56	\glxtr: new	55
\@alt@glshyp@opt: new	86	\glxtrcat: new	55
\@glscalt@hyp@opt: new	86	\glxtrdowrglossaryhook: new	86
\@glscalt@hyp@opt@char: new	86	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	86	new	112
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	54
new	108	\Glsxtrpl: new	56
\@glsc@local@increment@currunitcount:		\glxtrpl: new	55
new	108	\glxtrpostlocalreset: new	100
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	100
new	83	\glxtrpostreset: new	100
\@glxtr: new	55	\glxtrpostunset: new	98
\@glxtr@addunitcounter: new	107	\glxtrprotectlinks: new	89
\@glxtr@currunitcount: new	109	\GlsXtrSetAltModifier: new	86
\@glxtr@ifunitcounter: new	108	\GlsXtrSetDefaultGlsOpts: new	85
\@glxtr@p@acrlong@: new	92	\glxtrstarflywarn: new	54
\@glxtr@p@acrlongpl@: new	92	\GlsXtrWarning: new	56
\@glxtr@p@acrshort@: new	92	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new	92	disables \setacronymstyle	114
\@glxtr@p@long@: new	91	1.0 (2016-01-24)	
\@glxtr@p@longpl@: new	92	\@glxtr@autoindexcrossrefs: new	16
\@glxtr@p@plural@: new	90	\@glxtr@idx@displaynumberlist:	
\@glxtr@p@short@: new	90	new	123
\@glxtr@p@shortpl@: new	91	\@glxtr@idx@entrynumberlist: new	125
\@glxtr@p@text@: new	90	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new	109	new	123
\@glxtr@setentryunitcountunsetattr:		\@glxtr@noidx@entrynumberlist:	
new	113	new	124
\@glxtr@unitcountlist: new	107	\@glxtr@noidx@numberlistloop:	
\@glxtrpl: new	55	new	124
\@newglossaryentryposthook: added		\@glxtr@reg@glosslist: new	116
empty see value if not set and added		\makeglossaries: new	116
'see' to field key map	45	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	54	\glxtrdiscardperiod: added check	
\cGlsformat: added	107	for first use	191
\cGlsformat: added	106	short-desc: fixed typo in	
\cGlsplformat: added	107	\glxtrinlinefullformat and	
\cGlsplformat: added	107	added missing second argument	228
\glstdisablehyper: added	88	1.02 (2016-04-25)	
\glstdohyperlink: added	87	\@glxtr@current@style: new	57

\@glsfirst@: set abbreviation and regular format	70	\glsxtrregularfont: new	61
\@glsfirstplural@: set abbreviation and regular format	71	\glsxtruserfield: new	284
\@glsname@: set abbreviation and regular format	72	\glsxtruserparen: new	284
\@glsplural@: set abbreviation and regular format	71	\glsxtrusersuffix: new	285
\@glsymbol@: set regular format	74	\GlsXtrWarnDeprecatedAbbrStyle:	
\@glsymbolplural@: set regular format	74	new	217
\@glstext@: set abbreviation and regular format	69	short-em-long-em: new	268
\@glsxtr@deprecated@abbrstyle:		short-em-long-em-desc: new	269
new	216	short-em-nolong: new	271
\@glsxtr@do@style: new	22	short-em-nolong-desc: new	273
\@glsxtr@doloctag: new	61	short-em-postfootnote: renamed from “postfootnote-em”	282
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand	125	short-footnote: new	224
\@glsxtr@pagetag: new	60	short-long-user: new	292
\@glsxtr@pagetag: new	60	short-long-user-desc: new	293
\@glsxtr@preloctag: new	61	short-nolong: new	228
\@glsxtr@postloctag: new	60	short-nolong-desc: new	229
\@glsxtr@preloctag: new	60	short-postfootnote: new	226
\glossentrydesc: added glossdescfont attribute check	174	short-sc-footnote: renamed from “footnote-sc”	245
\Glossentryname: added glossnamefont attribute check	178	short-sc-nolong: new	239
\glossentryname: added glossnamefont attribute check	175	short-sc-nolong-desc: new	241
moved post name hook inside condition	177	short-sc-postfootnote: renamed from “postfootnote-sc”	247
\glsabbrvmfont: new	263	short-sm-footnote: renamed from “footnote-sm”	259
\glsabbrvuserfont: new	285	short-sm-nolong: new	254
\glsfirstabbrvmfont: new	263	short-sm-nolong-desc: new	255
\glsfirstabbrvuserfont: new	285	short-sm-postfootnote: renamed from “postfootnote-sm”	261
\glsfirstlongemfont: new	263	\letabbreviationstyle: new	216
\glsfirstlonguserfont: new	285	\newabbreviationstyle: bug fix: corrected test for existence	215
\glsifnotregularcategory: new ...	170	long-em-noshort-em: new	275
\glslongdefaultfont: new	200	long-em-noshort-em-desc: new	279
\glslongemfont: new	263	long-em-short-em: new	265
\glslongfont: new	200	long-em-short-em-desc: new	266
\glslonguserfont: new	285	long-noshort: new	234
\glsxtrassignfieldfont: new	69	long-noshort-desc: new	233
\GlsXtrEnablePreLocationTag: new .	59	long-noshort-em: renamed from “long-em”	273
\glsxtrfirstscfont: new	234	long-noshort-em-desc: renamed from “long-desc-em”	277
\glsxtrfirstsmfont: new	249	long-noshort-sc: renamed from “long-sc”	242
\glsxtrlongshortdescsort: new ...	219	long-noshort-sc-desc: renamed from “long-desc-sc”	243
\glsxtrpostnamehook: added category check	179	long-noshort-sm: renamed from “long-sm”	256

long-noshort-sm-desc: renamed from	General: disabled docdef key at the start
\long-desc-sm 258	of the document 27
long-short-user: new 285	docdef option changed to choice 14
long-short-user-desc: new 291	\glstr@usesee: new 46
\renewabbreviationstyle: new 216	\glstrusesee: new 45
style: new 22	\glstruseseeformat: new 46
1.05 (2016-06-10)	\if@glstrdocdefrestricted: new .. 15
\eglssetwidest: new 386	1.07 (2016-08-15)
\glFindWidestAnyName: new 388	@@glstrp: new 93
\glFindWidestAnyNameLocation:	\GLSfirst@: added check for
new 394	nohyperfirst attribute 71
\glFindWidestAnyNameSymbol: new 391	\GLSfirstplural@: added check for
\glFindWidestAnyNameSymbolLocation:	nohyperfirst attribute 72
new 393	\GLSxtrp: new 94
\glFindWidestLevelTwo: new 390	\@Glsfirst@: added check for
\glFindWidestUsedAnyName: new .. 388	nohyperfirst attribute 70
\glFindWidestUsedAnyNameLocation:	\@Glsfirstplural@: added check for
new 393	nohyperfirst attribute 72
\glFindWidestUsedAnyNameSymbol:	\@Glsxtrp: new 93
new 391	\@glspreglossaryhook: added
\glFindWidestUsedAnyNameSymbolLocation:	\glossxtrsetpopts 189
new 392	\@glfirst@: added check for
\glFindWidestUsedLevelTwo: new . 389	nohyperfirst attribute 70
\glFindWidestUsedTopLevelName:	\@glfirstplural@: added check for
new 387	nohyperfirst attribute 71
\glfirstlongfootnotefont: new .. 222	\@glxtrinmark: new 312
\glsgetwidestname: new 387	\@glxtrnotinmark: new 312
\glsgetwidestsubname: new 387	\@glxtrp: new 93
\glslongfootnotefont: new 222	\@glxtrp@opt: new 93
\glxtrAltTreeIndent: new 385	\glossxtrsetpopts: new 93
\glxtralttreeInit: new 386	\glsp: new 95
\glxtrAltTreePar: new 385	\glsp: new 95
\glxtrAltTreeSetHangIndent: new 395	\glxtr@entry@p: new 94
\glxtrAltTreeSetSubHangIndent:	\glxtrabbrvfootnote: new 222
new 395	\glxtrchecknohyperfirst: new 70
\glxtralttreeSubSymbolDescLocation:	\glxtrfieldtitlecasecs: new 173
new 385	\glxtrifinmark: new 312
\glxtralttreeSymbolDescLocation:	\GLSxtrp: new 97
new 385	\Glsxtrp: new 96
\glxtrComputeTreeIndent: new ... 394	\glxtrp: new 94
\glxtrComputeTreeSubIndent: new 395	\glxtrsetpopts: new 93
\glxtrtreetopindent: new 386	short-long-desc: added text key 222
short-em-long: fixed incorrect font used	fixed misspelling of \glabbrvfont in
by long form 267	plural key 222
\xglsetwidest: new 386	long-short-desc: added missing text
1.06 (2016-06-18)	key 219
\@glsoifexistsorwarn: new 15	fixed misspelling of \glabbrvfont . 220
\@glxtr@docdefval: new 15	footnote: changed first forms to use
\@glxtr@usesee: new 46	\glfirstlongfootnotefont ... 223

postfootnote: removed \footnote		\@printglossary: redefined to save	
from first keys	224	options	122
switched from \glsfirstlongfont to		\glsxtr@makeglossaries: new	122
\glsfirstlongfootnotefont ...	225	1.10 (2016-12-17)	
\RestoreAcronyms: modified		\@GLSp1@: fixed bug caused by typo in	
\@gls@link@checkfirsthyper to		command name	63
set \glsxtrifwasfirstuse	115	1.11 (2017-01-19)	
1.08 (2016-12-13)		\@glsxtr@do@redef@forglsentries:	
\@glsxtr@record: new	8	new	6
\@GLS@: added \@glsxtr@record	63	\@glsxtr@noidx@do: new	143
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@redef@forglsentries: new .	6
\@GLS@: added \@glsxtr@record	63	\@glsxtr@shortcutsval: new	20
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@unsrt@getgrouptitle: new	142
\@gls@: added \@glsxtr@record	62	\@print@noidx@glossary: added	
\@gls@link@: added		redefinition	126
\@glsxtr@record	64	\glsxtr@addloclistfield: added	
\@gls@field@link: added		group key	13
\@glsxtr@record	62	added location key	12
\@gls@saveentrycounter: new	27	\glsxtr@fields: new	135
\@glsdisp: added \@glsxtr@record ..	63	\glsxtr@linkprefix: new	135
\@GLSp1@: added \@glsxtr@record ...	62	\glsxtr@org@newignoredglossary:	
\@glsxtr@dorecord: new	10	new	40
\@glsxtr@err@undefaction: new	6	\glsxtr@s@newignoredglossary: new	41
\@glsxtr@record: new	7	\glsxtr@shortcutsval: new	135
\@glsxtr@warn@onexistsordo: new ...	6	\glsxtr@texencoding: new	135
\@glsxtr@warn@undefaction: new	6	\glsxtr@writefields: new	135
\@print@unsrt@glossary: new	140	\GlsXtrLoadResources: new	135
record: added record package option ...	13	\glsxtrpageref: new	38
\glsadd: added \@glsxtr@record	68	\glsxtrresourcefile: changed	
\glsdoifexists: now defines		extension to .gls tex	134
\glslabel	44	\newignoredglossary: added starred	
\glsxtr@do@wrglossary: new	27	version	40
\glsxtr@addloclistfield: new	12	1.12 (2017-02-03)	
\glsxtr@indexonly@saveentrycounter:		\@glsxtr@recordcounter: new	11
new	12	\@gls@preglossaryhook: check for	
\glsxtr@record: new	137	definition	188
\glsxtr@resource: new	135	\@glsxtr@counterrecordhook: new .	137
\glsxtr@saveentrycounter: new	27	\@glsxtr@display@loc: new	127
\glsxtr@setup@record: new	12	\@glsxtr@docounterrecord: new ...	137
\glsxtrassignfieldfont: added check		\@glsxtr@longnewglossaryentry:	
for existence	69	new	40
\glsxtrresourcefile: new	134	\@glsxtr@noop@recordcounter: new .	11
\printunsrtglossaries: new	140	\@glsxtr@op@recordcounter: new ...	11
\printunsrtglossary: new	139	\@glsxtr@provide@storagekey: new .	28
1.09 (2016-12-16)		\@glsxtr@s@longnewglossaryentry:	
\@glsxtr@gettype: new	123	new	39
\@glsxtr@mixed@assign@sortkey:		\@glsxtrentryfmt: new	31
new	123	\@glsxtrindexaliased: new	84
		\@glsxtrsetaliasnoindex: new	84

\@newglossaryentryposthook: added check for alias key	49	\glxtrindexaliased: new	85
\@no@glxtrindexaliased: new	84	\GlsXtrLetField: new	34
\@printunsrtglossary: new	139	\GlsXtrLetFieldToField: new	34
General: added target key to printgloss family	122	\GlsXtrLoadResources: removed restriction on only one per document	135
\apptoglossarypreamble: new	38	\glxtrlocrangefmt: new	128
\csGlsXtrLetField: new	34	\glxtrpostlongdescription: new ..	40
\eGlsXtrSetField: new	35	\glxtrprovidestoragekey: new	28
\gGlsXtrSetField: new	35	\GlsXtrRecordCounter: new	137
\glsdohyperlink: added check for alias field	88	\glxtrresourcecount: new	135
\glsnoidxdisplayloc: added redefinition	127	\glxtrresourcefile: added catcode change for @	134
\glissettoctitle: added patch	41	\glxtrsetaliasnoindex: new	84
\glxtr@counterrecord: new	137	\GlsXtrSetField: new	34
\glxtr@langtag: new	135	\glxtrsetfieldifexists: new	34
\glxtr@newabbreviation: new	196	\glxtrunsrtdo: new	143
\glxtr@org@newignoredglossary: Added check for existence	40	\glxtrusefield: new	34
\glxtr@pluralsuffixes: new	135	\glxtrusefield: new	34
\glxtr@provideignoredglossary: new	42	short-postlong-user: new	289
\glxtr@s@newignoredglossary: Added check for existence	41	short-postlong-user-desc: new ...	290
\glxtr@s@provideignoredglossary: new	43	\longnewglossaryentry: added starred version	39
\glxtrabbrvpluralsuffix: new ...	200	long-postshort-user: new	286
\glxtralias: new	49	long-postshort-user-desc: new ...	288
\glxtrcopytoglossary: new	43	postdot: new	16
\glxtrdeffield: new	34	\pretoglossarypreamble: new	38
\glxtrdisplayendloc: new	128	\print@noop@unsrtglossaryunit: new	142
\glxtrdisplayendlochook: new ...	128	\print@op@unsrtglossaryunit: new	142
\glxtrdisplaysingleloc: new	127	\printunsrtglossary: added starred form	139
\glxtrdisplaystartloc: new	127	\printunsrtglossaryhandler: new .	141
\glxtrredefield: new	34	\printunsrtglossaryunit: new	12
\glxtrentryfmt: new	30	\printunsrtglossaryunitsetup: new	142
\glxtrfieldddolistloop: new	31	\provideignoredglossary: new	42
\glxtrfieldforlistloop: new	32	\s@glxtr@provide@storagekey: new	29
\glxtrfielddininlist: new	32	\s@printunsrtglossary: new	140
\glxtrfieldlistadd: new	31	\xGlsXtrSetField: new	35
\glxtrfieldlistadd: new	31	1.13 (2017-02-07)	
\glxtrfieldlistgadd: new	31	\@glsdisp: removed	
\glxtrfieldlistxadd: new	31	\@glxtr@org@glsdisp	63
\glxtrfieldxifinlist: new	32	\glxtrsetaliasnoindex: switched to \providecommand	84
\glxtrfmt: new	29	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	29	\@gls@link: added redefinition	65
\GlsXtrFmtField: new	29	\@gls@noidx@getgrouptitle: new ..	125
\glxtrifkeydefined: new	28	\@gls@removespaces: new	128
		\@glxtr@do@automake@err: new ...	137
		\@glxtr@org@gloautosee: new	25

\@glxtr@record: added third arg	7	postfootnote: fixed spelling of	
\@glxtr@recordsee: new	11	\glsabbrvfont	224
General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	68	\@glo@autosee: added redefinition	26
added \glsadd option thevalue	68	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	88	bug	125
\glsenableentrycount: fixed		\@glxtr@addunusedxrefs: added	
assignment of \@cGls@	102	check for seealso field	50
\glsenableentryunitcount: fixed		\@glxtr@checkgroup: use \csuse	
assignment of \@cGls@	111	instead of \csname	143
\glsnavigation: new	126	\@glxtr@dorecordnodefer: new	11
\glxtr@org@getgrouptitle: new ..	125	\@print@unsrt@glossary: corrected	
\glxtr@recordsee: new	7	misspelt command	140
\glxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	136	new	141
\glxtrdisplayendloc: added check		record: added check for	
for empty format	128	\@gls@setupsort@none	14
\glxtrgetgrouptitle: new	125	\gls@checkseeallowed: added	
\glxtrinitwrgloss: new	64	redefinition	26
\glxtrlocationhyperlink: new ...	129	\glxtr@writefields: added	
\glxtrsetgrouptitle: new	125	\providecommand lines	135
\glxtrsupphypernumber: new	129	\glxtrautoindex: new	182
\ifglxtrwrglossbefore: new	64	\glxtrautoindexassignsort: new .	183
1.15 (2017-05-10)		\glxtrautoindexentry: new	183
\@glxtr@dorecord: corrected		\glxtrindexseealso: new	47
premature expansion of \@glslocref	10	\glxtrseealsolabels: new	49
short-em-long-em: fixed spelling of		\glxtrseelist: new	47
\glsabbrvfont	268	\glxtruseseealso: new	46
short-long: fixed spelling of		\glxtruseseealsoformat: new	46
\glsabbrvfont	220	\seealso name: new	47
short-long-user: fixed spelling of		autoseeindex: new	16
\glsabbrvfont	292	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@gls@setup@default@short@access:	
\glsabbrvfont	289	removed some inconsistencies in the	
short-postlong-user-desc: fixed		abbreviation styles	218
spelling of \glsabbrvfont	291	\@glxtr@mark@wordseps: new	195
long-em-short-em: fixed spelling of		\@glxtr@markwordseps: new	195
\glsabbrvfont	265	\@glxtr@noidx@displaynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	287	\glxtrundeftag	123
long-postshort-user-desc: fixed		\@glxtr@noidx@entrynumberlist:	
spelling of \glsabbrvfont	288	replace hard-coded ?? with	
long-short: fixed spelling of		\glxtrundeftag	124
\glsabbrvfont	218	\@glxtr@noidx@numberlistloop:	
long-short-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	285	\glxtrundeftag	124
footnote: fixed spelling of		\@glxtrifhyphenstart: new	294
\glsabbrvfont	223	\glsabbrvhyphenfont: new	295
		\glsabbrvonlyfont: new	308

<code>\glsabbrvscfont:new</code>	234	<code>short-hyphen-postlong-hyphen-desc:</code>	
<code>\glsabbrvsmfont:new</code>	249	<code>new</code>	307
<code>\glsabbrvuserfont: initialised to</code>		<code>short-long-user-desc: corrected first</code>	
<code>default font</code>	285	<code>forms</code>	293
<code>\glsfirstabbrvhyphenfont:new</code>	295	<code>short-nolong-desc-noreg:new</code>	230
<code>\glsfirstabbrvonlyfont:new</code>	308	<code>short-nolong-noreg:new</code>	228
<code>\glsfirstabbrvscfont:new</code>	234	<code>long-em-noshort-em-desc-noreg:</code>	
<code>\glsfirstabbrvsmfont:new</code>	249	<code>new</code>	280
<code>\glsfirstlonghyphenfont:new</code>	295	<code>long-em-noshort-em-noreg:new</code>	277
<code>\glsfirstlongonlyfont:new</code>	308	<code>long-hyphen-noshort-desc-noreg:</code>	
<code>\glslonghyphenfont:new</code>	295	<code>new</code>	297
<code>\glslongonlyfont:new</code>	308	<code>long-hyphen-postshort-hyphen:new</code>	300
<code>\glslonguserfont: initialised to default</code>		<code>long-hyphen-postshort-hyphen-desc:</code>	
<code>font</code>	285	<code>new</code>	302
<code>\glxtr@newabbreviation: added</code>		<code>long-hyphen-short-hyphen:new</code>	295
<code>\glxtrorgshort and</code>		<code>long-hyphen-short-hyphen-desc:</code>	
<code>\glxtrorglong</code>	196	<code>new</code>	296
<code>\GlsXtrDefineAcShortcuts:new</code>	18	<code>long-noshort-desc-noreg:new</code>	233
<code>\glxtrgenabbrvfmt: added check for</code>		<code>long-noshort-noreg:new</code>	234
<code>\ifglxtrininsertinside</code>	212	<code>long-only-short-only:new</code>	308
<code>\glxtrhyphensuffix:new</code>	295	<code>long-only-short-only-desc:new</code>	310
<code>\glxtrifhyphenstart:new</code>	294	<code>long-short-user-desc: corrected first</code>	
<code>\glxtrlonghyphen:new</code>	299	<code>forms</code>	291
<code>\glxtrlonghyphennoshort:new</code>	296	1.18 (2017-08-10)	
<code>\glxtrlonghyphenshort:new</code>	294	<code>stylemods: changed default value to</code>	
<code>\glxtrlongshortdescname:new</code>	219	<code>"default"</code>	22
<code>\glxtronlydescname:new</code>	310	1.19 (2017-09-09)	
<code>\glxtronlydescsort:new</code>	310	<code>\@glxtr@defaultnumberformat:new</code>	7
<code>\glxtronlysuffix:new</code>	308	<code>\@glxtr@dorecord: Use</code>	
<code>\glxtrparen:new</code>	199	<code>\@glxtr@recordloc instead of</code>	
<code>\glxtrposthyphenlong:new</code>	305	<code>\@glxtr@recordloc</code>	10
<code>\glxtrposthyphenshort:new</code>	299	<code>\@glxtr@dorecordnodefer: Use</code>	
<code>\glxtrposthyphensubsequent:new</code>	300	<code>\theglentrycounter for the</code>	
<code>\glxtrshortdescname:new</code>	228	<code>location rather than \@glxtr@recordloc</code>	11
<code>\glxtrshorthyphen:new</code>	304	<code>\@glxtr@record@setting:new</code>	13
<code>\glxtrshorthyphenlong:new</code>	302	<code>\@glxtr@record@setting@alsoindex:</code>	
<code>\glxtrshortlongdescname:new</code>	221	<code>new</code>	13
<code>\glxtrshortlongdescsort:new</code>	221	<code>\@glxtrifhasfield:new</code>	32
<code>\Glsxtrsubsequentfmt:new</code>	214	General: added <code>\glslink</code> option	
<code>\glxtrsubsequentfmt:new</code>	214	<code>theHvalue</code>	65
<code>\Glsxtrsubsequentplfmt:new</code>	214	added <code>\glslink</code> option <code>thevalue</code>	65
<code>\glxtrsubsequentplfmt:new</code>	214	<code>\glxtr@writefields: removed</code>	
<code>\glxtrword:new</code>	195	<code>double-quotes around \@jobname</code>	136
<code>\glxtrwordsep:new</code>	195	<code>\glxtrdoautoindexname: changed</code>	
<code>short-hyphen-long-hyphen:new</code>	303	<code>format test</code>	182
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink:new</code>	88
<code>new</code>	304	<code>\glxtrifhasfield:new</code>	32
<code>short-hyphen-postlong-hyphen:new</code>	305	<code>\GlsXtrSetDefaultNumberFormat:</code>	
		<code>new</code>	7

\s@glxstrifhasfield: new	32	\@glxstrsetaliasnoindex: changed to use \glxstrifhasfield instead of \ifglshasfield	84
1.20 (2017-09-11)		\@glxstrwrglossmark: new	24
\@glxstrhypernameprefix: new	122	\@rGLS: new	151
\glsdohypertarget: added redefinition	122	\@rGLS@: new	151
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glxstrhypernameprefix	142	\@rGLSpl: new	151
1.21 (2017-11-03)		\@rGLSpl@: new	151
\@@glxstr@record: added check for default options	9	\@rGLs: new	150
\@@glxstrwrglossmark: new	24	\@rGLspl: new	150
\@gls@setup@default@short@access: modified index to remove hard coded \space	380	\@rGLspl@: new	150
modified list to remove hard coded \space	369	\@rgls: new	149
moved conditional outside of \glsgroupskip	372–379	\@rgls@: new	149
redefined altlistgroup to discourage breaks after group headings	370	\@rglspl: new	149
redefined altlisthypergroup to discourage breaks after group headings	371	\@rglspl@: new	149
redefined indexgroup to discourage breaks after group headings	381	General: adjusted mcolalmtree	400
redefined indexhypergroup to discourage breaks after group headings	381	new	403
redefined listgroup to discourage breaks after group headings	370	redefined alttreegroup to discourage breaks after group headings	396
redefined listhypergroup to discourage breaks after group headings	370	redefined alttreehypergroup to discourage breaks after group headings	397
\@glslink: changed \let to \def	89	redefined mcolalmtreegroup to discourage breaks after group headings	401
\@glxstr@checkgroup: new	143	redefined mcolalmtreehypergroup to discourage breaks after group headings	401
\@glxstr@defpostpunc: new	16	redefined mcolalmtreespannav to discourage breaks after group headings	402
\@glxstr@do@record@wrglossary: new	8	redefined mcolindexgroup to discourage breaks after group headings	397
\@glxstr@dosee@alsoindex@glossary: new	25	redefined mcolindexhypergroup to discourage breaks after group headings	397
\@glxstr@doseeglossary: new	25	redefined mcolindexspannav to discourage breaks after group headings	398
\@glxstr@noidx@do: removed code dealing with the group	144	redefined mcoltreegroup to discourage breaks after group headings	398
\@glxstr@record@setting@off: new .	13	redefined mcoltreehypergroup to discourage breaks after group headings	399
\@glxstr@record@setting@only: new	13	redefined mcoltreenamegroup to discourage breaks after group	
\@glxstr@rglstrigger@record: new	148		
\@glxstrglossentry: new	138		
\@glxstrnewglsl: new	145		

headings	399	\glstrbookindexatendgroup: new ..	404
redefined		\glstrbookindexbetween: new	404
mcoltreenonamehypergroup to		\glstrbookindexbookmark: new ...	404
discourage breaks after group		\glstrbookindexcols: new	403
headings	400	\glstrbookindexcolspread: new ..	404
redefined mcoltreenonamespannav to		\glstrbookindexfirstmark: new ..	408
discourage breaks after group		\glstrbookindexfirstmarkfmt: new	408
headings	400	\glstrbookindexformatheader: new	404
redefined mcoltreepannav to		\glstrbookindexgroupskip: new ..	404
discourage breaks after group		\glstrbookindexlastmark: new ...	408
headings	399	\glstrbookindexlastmarkfmt: new	408
redefined treenonamegroup to		\glstrbookindexmarkentry: new ..	408
discourage breaks after group		\glstrbookindexname: new	403
headings	384	\glstrbookindexparentchildsep:	
redefined treenonamehypergroup to		new	403
discourage breaks after group		\glstrbookindexparentschildsep:	
headings	384	new	403
debug: new	24	\glstrbookindexprelocation: new	403
\gglsssetwidest: new	386	\glstrbookindexsubatendgroup:	
\gl:disablehyper: added check for		new	404
existence	88	\glstrbookindexsubbetween: new ..	404
changed to use \def rather than \let ..	88	\glstrbookindexsubname: new	403
\gl:dohypertarget: redefined		\glstrbookindexsubprelocation:	
treegroup to discourage breaks after		new	403
group headings	382	\glstrbookindexsubsubatendgroup:	
redefined treehypergroup to		new	404
discourage breaks after group		\glstrbookindexsubsubbetween:	
headings	383	new	404
\gl:enablehyper: changed to use \def		\glstrbookindexthepage: new	407
rather than \let	89	\glstrdetoklocation: new	147
\gl:fmtname: new	325	\glxtrenablerecordcount: new ...	147
\gl:fmtname: new	325	\glstrglossentry: new	137
\gl:shex: new	331	\glstrgroupfield: new	143
\gl:listchildpostlocation: new ..	369	\GlsXtrheadname: new	316
\gl:listchildprelocation: new ...	369	\GlsXtrheadname: new	316
\gl:listprelocation: new	369	\GlsXtrIfFieldEqStr: new	35
\gl:navhyperlink: patched	87	\glstriflabelinlist: new	142
\gl:seeitemformat: new	46	\glstrifrecordtrigger: new	148
\gl:sshowtarget: new	25	\glstrindexseealso: added check	
\gl:treechildprelocation: new ...	380	that the entry exists	47
\gl:treeprelocation: new	380	\glstrinithyperoutside: new	65
\gl:triggerrecordformat: new	149	\GlsXtrLocationRecordCount: new ..	147
\gl:useabbrvfont: new	211	\glstrnewgls: new	145, 146
\gl:suselongfont: new	212	\glstrnewGLSlike: new	146
\gl:xttr@do@alsoindex@wrglossary:		\glstrnewglslike: new	146
new	8	\glstrnewrgls: new	146
\gl:xttr@org@@do@wrglossary: new ..	27	\glstrnewrGLSlike: new	146
\gl:xttr@org@dohyperlink: new	87	\glstrnewrglslike: new	146
\gl:xttr@setbookindexmark: new ...	408	\glstrprelocation: new	368, 403

\GlsXtrRecordCount: new	147	\@glxtr@orgprintglossary: changed	
\glxtrrecordtriggervalue: new ..	147	explicit \let for \nopostdesc to	
\glxtrresourcefile: now disables		\glxtractivatenopost	120
record key	134	\@glxtrglossentryother: new	139
\glxtrresourceinit: new	135	\glossentrynameother: new	180
\GlsXtrSetRecordCountAttribute:		\glseeitemformat: switched check	
new	147	from regular to short	46
\glxtrtitlename: new	316	\glxtr@setaccessdisplay: new ...	179
\glxtrtitleorpdforheading: new ..	312	\glxtr@writefields: provide	
\GlsXtrTotalRecordCount: new	147	\glxtr@record in aux file	136
\glxtrwrglossmark: new	24	\glxtractivatenopost: new	121
short-em: new	271	\glxtrbookindexprelocation:	
short-sc: corrected first letter		removed check for no post dot	403
uppercasing	239	\glxtrglossentryother: new	138
short-sm: corrected first letter		\glxtrnopostpunc: new	121
uppercasing	253	1.23 (2017-11-12)	
shortcuts: ac	21	\@@glxtrfmt: added check for indexing	30
\ifglxtr@hyperoutside: new	65	added grouping	30
all: new	367	new	30
nolong-short: new	230	\@glxtr@nopostpunc@postdesc: new	121
nolong-short-em: new	273	\@glxtr@restore@postpunc: new ..	121
nolong-short-noreg: new	231	\@glxtrentryfmt: fixed missing label	
nolong-short-sc: new	241	argument	31
nolong-short-sm: new	255	\@glxtrfmt: new	29
nopostdot: new	16	\eglupdatewidest: new	387
postpunc: new	16	\gglupdatewidest: new	386
\printunsrtglossaryentryprocesshook:		\glupdatewidest: new	386
new	141	\GlsXtrDefineAbbreviationShortcuts:	
\printunsrtglossarypredoglossary:		changed \newabbr definition to use	
new	141	\providecommand	18
\printunsrtglossaryskipentry: new	141	\GlsXtrDefineAcShortcuts: changed	
\rGLS: new	150	\newabbr definition to use	
\rGls: new	150	\providecommand	19
\rgls: new	149	\glxtrfmtdisplay: new	30
\rGLSformat: new	152	\glxtrifcustomdiscardperiod: new	190
\rGlsformat: new	152	\GlsXtrIfFieldUndef: new	33
\rglsformat: new	151	\glxtrrestorepostpunc: new	122
\rGLSpl: new	151	\s@glxtrfmt: new	30
\rGlspl: new	150	\s@glxtrfmt: new	30
\rglspl: new	149	\xglupdatewidest: new	387
\rGLSplformat: new	152	1.24 (2017-11-14)	
\rglsplformat: new	152	\glxtr@setsort	68
\s@glxtrifhasfield: switched from		\glxtrforcsvfield: new	32
\ifdef to \ifundef	32	\glxtrlocalsetgrouptitle: new ..	126
1.22 (2017-11-08)		1.25 (2017-11-14)	
\@glxtr@nopostpunc: new	121	\glxtrbookindexmulticolsest: new	404
		1.25 (2017-11-24)	
		\glsextrapostnamehook: new	179
		\glxtrfootnotename: new	222

\glxtrlongnoshortdescname: new .	231	\glxtrGeneralLatinIIrules: new .	342
\glxtrlongnoshortname: new	233	\glxtrGeneralLatinIrules: new ..	342
\glxtrlongshortname: new	218	\glxtrGeneralLatinIVrules: new .	344
\glxtrlongshortuserdescname: new	288	\glxtrGeneralLatinVIIrules: new	346
\glxtronlyname: new	308	\glxtrGeneralLatinVIIrules: new	346
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVrules: new .	345
changed to use \glxtrparen	191	\glxtrGeneralLatinVrules: new ..	344
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules: new ..	341
changed to use \glxtrparen	191	\glxtrgeneralpuncIrules: new ...	340
\glxtrshortlongname: new	220	\glxtrgeneralpuncrules: new	340
\glxtrshortlonguserdescname: new	290	\glxtrhyphenrules: new	340
\glxtrshortnolongname: new	226	\glxtrLatinA: new	347
1.26 (2018-01-05)		\glxtrLatinAA: new	349
\@glxtr@do@inc@linkcount: new ..	152	\glxtrLatinAELigature: new	349
\glslinkpresetkeys: new	65	\glxtrLatinE: new	347
\glxtr@inc@linkcount: new	65	\glxtrLatinEszettSs: new	349
\GlsXtrEnableLinkCounting: new ..	153	\glxtrLatinEszettSz: new	349
\GlsXtrIfLinkCounterDef: new	153	\glxtrLatinEth: new	349
\glxtrinclinkcounter: new	153	\glxtrLatinH: new	347
\GlsXtrLinkCounterName: new	153	\glxtrLatinI: new	348
\GlsXtrLinkCounterValue: new	153	\glxtrLatinInsularG: new	350
\GlsXtrTheLinkCounter: new	153	\glxtrLatinK: new	348
1.27 (2018-02-26)		\glxtrLatinL: new	348
\@glset@setup@default@short@access:		\glxtrLatinLslash: new	350
added glossaries-extra-bib2gls.sty .	331	\glxtrLatinM: new	348
\@glxtrdialecthook: new	27	\glxtrLatinN: new	348
\Alpha: new	334	\glxtrLatinO: new	348
\Beta: new	334	\glxtrLatinOELigature: new	349
\Chi: new	334	\glxtrLatinOslash: new	350
\Digamma: new	334	\glxtrLatinP: new	348
\Epsilon: new	334	\glxtrLatinS: new	348
\Eta: new	334	\glxtrLatinSchwa: new	349
\glxtr@loaddialect: new	330	\glxtrLatinT: new	348
\glxtrBasicDigitrules: new	364	\glxtrLatinThorn: new	349
\glxtrcombiningdiacriticIIrules:		\glxtrLatinWynn: new	349
new	338	\glxtrLatinX: new	348
\glxtrcombiningdiacriticIrules:		\glxtrMathGreekIIrules: new	356
new	338	\glxtrMathGreekIrules: new	355
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new	360
new	337	\glxtrMathItalicBeta: new	360
\glxtrcombiningdiacriticIVrules:		\glxtrMathItalicChi: new	363
new	339	\glxtrMathItalicDelta: new	361
\glxtrcombiningdiacriticrules:		\glxtrMathItalicEpsilon: new ...	361
new	337	\glxtrMathItalicEta: new	361
\glxtrcontrolrules: new	336	\glxtrMathItalicGamma: new	360
\glxtrcurrencyrules: new	341	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new	364	new	352
\glxtrfractionrules: new	364	\glxtrMathItalicGreekIrules: new	351
\glxtrGeneralLatinIIrules: new	343	\glxtrMathItalicIota: new	361

<code>\glxtrMathItalicKappa:new</code>	361	<code>\glxtrUpPi:new</code>	359
<code>\glxtrMathItalicLambda:new</code>	362	<code>\glxtrUpPsi:new</code>	360
<code>\glxtrMathItalicLowerGreekIIrules:</code>		<code>\glxtrUpRho:new</code>	359
<code>new</code>	354	<code>\glxtrUpSigma:new</code>	359
<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau:new</code>	359
<code>new</code>	353	<code>\glxtrUpTheta:new</code>	358
<code>\glxtrMathItalicMu:new</code>	362	<code>\glxtrUpUpsilon:new</code>	360
<code>\glxtrMathItalicNabla:new</code>	363	<code>\glxtrUpXi:new</code>	359
<code>\glxtrMathItalicNu:new</code>	362	<code>\glxtrUpZeta:new</code>	358
<code>\glxtrMathItalicOmega:new</code>	363	<code>\Iota:new</code>	334
<code>\glxtrMathItalicOmicron:new</code>	362	<code>\Kappa:new</code>	334
<code>\glxtrMathItalicPartial:new</code>	363	<code>\Mu:new</code>	334
<code>\glxtrMathItalicPhi:new</code>	363	<code>\Nu:new</code>	334
<code>\glxtrMathItalicPi:new</code>	362	<code>\Omicron:new</code>	334
<code>\glxtrMathItalicPsi:new</code>	363	<code>\omicron:new</code>	335
<code>\glxtrMathItalicRho:new</code>	362	<code>\Rho:new</code>	334
<code>\glxtrMathItalicSigma:new</code>	362	<code>\Tau:new</code>	334
<code>\glxtrMathItalicTau:new</code>	363	<code>\Upalpha:new</code>	335
<code>\glxtrMathItalicTheta:new</code>	361	<code>\Upbeta:new</code>	335
<code>\glxtrMathItalicUpperGreekIIrules:</code>		<code>\Upchi:new</code>	335
<code>new</code>	353	<code>\Upsilon:new</code>	335
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta:new</code>	335
<code>new</code>	352	<code>\Upiota:new</code>	335
<code>\glxtrMathItalicUpsilon:new</code>	363	<code>\Upkappa:new</code>	335
<code>\glxtrMathItalicXi:new</code>	362	<code>\Upmu:new</code>	335
<code>\glxtrMathItalicZeta:new</code>	361	<code>\Upnu:new</code>	335
<code>\glxtrMathUpGreekIIrules:new</code>	350	<code>\Upomicron:new</code>	335
<code>\glxtrMathUpGreekIrules:new</code>	350	<code>\upomicron:new</code>	336
<code>\glxtrnonprintablerules:new</code>	337	<code>\Uprho:new</code>	335
<code>\glxtrprovidecommand:new</code>	331	<code>\Uptau:new</code>	335
<code>\glxtrspacerules:new</code>	337	<code>\Upzeta:new</code>	335
<code>\glxtrSubScriptDigitrules:new</code>	364	<code>\Zeta:new</code>	334
<code>\glxtrSuperScriptDigitrules:new</code>	364		
<code>\glxtrUpAlpha:new</code>	357	1.28 (2018-03-06)	
<code>\glxtrUpBeta:new</code>	357	<code>\@glxtr@docdefval</code> : changed from	
<code>\glxtrUpChi:new</code>	360	count register to macro	15
<code>\glxtrUpDelta:new</code>	357	<code>\@glxtrdialecthook</code> : save and restore	
<code>\glxtrUpDigamma:new</code>	358	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon:new</code>	357	365
<code>\glxtrUpEta:new</code>	358	<code>\glxtrredeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code>	357	<code>\protected@csedef</code>	34
<code>\glxtrUpIota:new</code>	358	<code>\glxtrlocalsetgrouptitle</code> : changed	
<code>\glxtrUpKappa:new</code>	358	<code>\csedef\protected@csedef</code>	126
<code>\glxtrUpLambda:new</code>	358	<code>\glxtrsetgrouptitle</code> : changed	
<code>\glxtrUpMu:new</code>	359	<code>\csxdef\protected@csxdef</code>	125
<code>\glxtrUpNu:new</code>	359	1.29 (2018-04-09)	
<code>\glxtrUpOmega:new</code>	360	<code>\@glxtr@removespaces</code> : added expansion	128
<code>\glxtrUpOmicron:new</code>	359	<code>\@glxtr@dorecord</code> : don't suppress	
<code>\glxtrUpPhi:new</code>	360	expansion of <code>\@glxtrrecordloc</code> if	
		counter isn't page	11

\@glxstr@wrglossary@locationhyperlink: new	23	\Glsxtrlongpl: added \@glxstr@record	210
\glxstr@inc@wrglossaryctr: new ...	23	\glxtrlongpl: added \@glxstr@record	210
\glxstr@wrglossarylocation: new ..	331	\GLSxtrshort: added \@glxstr@record	205
\GlsXtrBibTeXEntryAliases: new ..	332	\Glsxtrshort: added \@glxstr@record	205
\glxtrfieldforlistloop: corrected argument order in \forlistcsloop	32	\glxtrshort: added \@glxstr@record	205
\GlsXtrIndexCounterLink: new	331	\glxtrshort: added \@glxstr@record	204
\GlsXtrInternalLocationHyperlink: new	23	\GLSxtrshortpl: added \@glxstr@record	209
\GlsXtrProvideBibTeXFields: new ..	333	\Glsxtrshortpl: added \@glxstr@record	208
indexcounter: new	23	\glxtrshortpl: added \@glxstr@record	208
\setentrycounter: new	128	\GlsXtrStartUnsetBuffering: new ..	98
1.30 (2018-04-25)		\GlsXtrStopUnsetBuffering: new ...	99
\@@glxstr@record: added check for post-key hook	9	indexcounter: added check for wrglossary counter	23
added check for pre-key hook	9	\s@GlsXtrStopUnsetBuffering: new ..	99
\@GLSxtr@fullpl: added \@glxstr@record	204	1.31 (2018-05-09)	
\@GlsXtrStopUnsetBuffering: new ..	99	\@GlsXtrStartUnsetBuffering: new ..	98
\@Glsxtr@fullpl: added \@glxstr@record	203	\@gl@ifaccessattribute@set: new	162
\@glxstr@dorecord: don't suppress expansion of \@glarecordlocoref ..	11	\@gl@initaccesskeys: new ...	162, 167
\@glxstr@full: added \@glxstr@record	201	\@gl@setup@default@short@access: new	162, 167
\@glxstr@fullpl: added \@glxstr@record	202	\@glxstr@record@noglossarywarning: new	134
\@glxstr@glossadd@postkeys: new ..	10	\@glxstrbuffer@nodup@unset: new ..	99
\@glxstr@glossadd@prekeys: new ...	10	General: added prefix key for glslink	65
\@glxstr@glslink@postkeys: new ...	10	added prefix key for printgloss ..	122
\@glxstr@glslink@prekeys: new	10	changed \let to \def	122
\@glxstr@local@textformat: new ...	65	\glsaddeach: new	69
\@glxstr@unset: new	98	\glsapturedgroup: new	331
\@glxstrbuffer@unset: new	99	\glsdefpostdesc: new	189
\glsadd: added \glsaddpostsetkeys	68	\glsdefpostlink: new	190
added \glsaddpresetkeys	68	\glsdefpostname: new	179
\glsaddpostsetkeys: new	68	\glsdohypertarget: bug fix: ensure that new version is picked up	122
\glsaddpresetkeys: new	68	\glslistdesc: new	369
\glsuserdescription: new	285	\glslocalreseteach: new	100
\glxtrabbreviationfont: new	62	\glslocalunseteach: new	100
\GlsXtrDualBackLink: new	332	\glstreechilddesc: new	382
\GlsXtrDualField: new	332	\glstreechildsymbol: new	382
\GlsXtrExpandedFmt: new	65	\glstreedefaultnamefmt: new	379
\GLSxtrlong: added \@glxstr@record	207	\glstreegroupheaderfmt: added redefinition	379
\Glsxtrlong: added \@glxstr@record	207	\glstreenamefmt: added redefinition	379
\glxtrlong: added \@glxstr@record	206		
\GLSxtrlongpl: added \@glxstr@record	211		

\glstreenavigationfmt: added		\s@GlsXtrStartUnsetBuffering: new	99
redefinition	380	1.32 (2018-05-24)	
\glstreenonamechilddesc: new 383	\GlsXtrForeignText: new 36
\glstreenonamesymbol: new 383	\GlsXtrForeignTextField: new 37
\glstreesymbol: new 382	\GlsXtrUnknownDialectWarning: new	37
\glsxtr@newabbreviation: added		1.33 (2018-07-26)	
\ExtraCustomAbbreviationFields		\ifglused: added redefinition 39
.....	196	1.34 (2018-07-29)	
\GlsXtrForUnsetBufferedList: new	. 99	\glx@begindocdefs: atom 51
\GlsXtrIfFieldCmpNum: new 33	\GlsXtrIfUnusedOrUndefined: new	.. 27
\GlsXtrIfFieldEqNum: new 33	\glxtrNoGlossaryWarning: added	
\GlsXtrIfFieldEqXpStr: new 35	package warning	21
\GlsXtrIfFieldNonZero: new 33	\if@glxtrdocdefrestricted:	
\GlsXtrIfHasNonZeroChildCount:		changed to allow for atom as well	... 15
new	331	docdef: atom 15
\GlsXtrIfXpFieldEqXpStr: new 35	1.35 (2018-08-13)	
\glxtrpostlinkAddSymbolDescOnFirstUse:		\@glx@@link@: initialise post-link hook	
new	191	commands	64
\GlsXtrRecordWarning: new 132	1.36 (2018-08-18)	
\glxtrRevertTocMarks: new 312	\glxtrautoindexesc: new 182
\GlsXtrStandaloneGlossaryType:		\glxtrdisplaysupplc: new 333
new	138	\glxtrmultisupplcation: new	... 333
\GlsXtrStandaloneSubEntryItem:			
new	138		

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	
\\$	331
\.	16, 17, 190, 191, 379
\@	51, 134
\@cGLS@	102, 111
\@cGLSpl@	103, 111
\@cGls@	102, 111
\@cGlspl@	102, 111
\@cgls@	102, 111
\@cglspl@	102, 111
\@do@wrglossary	8, 10, 117
\@do@wrglossary	12–14, 27, 68, 84
\@glo@assign@sortkey	123
\@glo@list	6
\@glo@type	140
\@glossarysec	404
\@gls@expand@field	29
\@glslocalreset	100
\@glslocalunset	100
\@glsreset	100
\@glsunset	98
\@glsxtr@autoindex@escspch	184–186
\@glsxtr@checkspch	183, 184, 186
\@glsxtr@disabledflycommand	57
\@glsxtr@org@postdescription	121
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 137
\@glsxtrfmt	29, 30
\@glsxtrp	93, 94
\@glsxtrp@postloctag	59
\@glsxtrp@preloctag	59, 60
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 47, 52, 117
\@newglossaryentry@defcounters	101
\@newglossaryentry@defunitcounters	109
\@par	385
\@ACRlong	90
\@ACRlongpl	90
\@ACRshort	90
\@ACRshortpl	90
\@Acrlong	90
\@Acrlongpl	90
\@Acrshort	90
\@Acrshortpl	90
\@GLS@	89, 105, 106, 151
\@GLSdesc@	73
\@GLSpl@	89, 105, 106, 151
\@GLSplural@	90
\@GLSSymbol@	74
\@GLSxtr@	90
\@GLSxtr@full	202
\@GLSxtr@fullpl	203
\@GLSxtr@p@acrlong@	90
\@GLSxtr@p@acrlongpl@	90
\@GLSxtr@p@acrshort@	90
\@GLSxtr@p@acrshortpl@	90
\@GLSxtr@p@long@	89
\@GLSxtr@p@longpl@	90
\@GLSxtr@p@plural@	89
\@GLSxtr@p@short@	89
\@GLSxtr@p@shortpl@	90
\@GLSxtr@p@text@	89
\@GLSxtrlong	89, 207
\@GLSxtrlongpl	90, 211
\@GLSxtrp	97, 98
\@GLSxtrshort	89, 205
\@GLSxtrshortpl	90, 209
\@Gls@	89, 104, 105, 150
\@Gls@acentryname	113
\@Gls@entry@field	81, 96, 180
\@Gls@entryname	113
\@GlsXtrEnableOnTheFly	54
\@GlsXtrStartUnsetBuffering	98
\@GlsXtrStopUnsetBuffering	99

<code>\@Glspl@</code>	89, 105, 106, 150	<code>\@end@glstr@addunused</code>	50, 51
<code>\@Glsplural@</code>	90	<code>\@end@glstr@gettype</code>	119, 123
<code>\@Glstext@</code>	90	<code>\@end@glstr@usesee</code>	45, 46
<code>\@Glsxtr</code>	55, 56	<code>\@end@glstr@trifhyphenstart</code>	294
<code>\@Glsxtr@full</code>	201	<code>\@endfortrue</code>	32, 180, 215
<code>\@Glsxtr@fullpl</code>	203	<code>\@firstofone</code>	69, 140, 174, 175, 181, 187
<code>\@Glsxtr@p@acrlong@</code>	90	<code>\@firstofthree</code>	63,
<code>\@Glsxtr@p@acrlongpl@</code>	90		69, 77–80, 86, 201, 203, 204, 206, 208, 210
<code>\@Glsxtr@p@acrshort@</code>	90	<code>\@firstoftwo</code>	70–74, 78, 80, 83, 86, 115, 180,
<code>\@Glsxtr@p@acrshortpl@</code>	90		192, 193, 201, 203, 204, 208–211, 312, 313
<code>\@Glsxtr@p@long@</code>	89	<code>\@for</code>	6, 22, 32, 51, 69, 100, 101, 113,
<code>\@Glsxtr@p@longpl@</code>	90		116, 119, 126, 140, 147, 154, 173, 179, 187
<code>\@Glsxtr@p@plural@</code>	89	<code>\@glo@alias</code>	48, 49
<code>\@Glsxtr@p@short@</code>	89	<code>\@glo@assign@sortkey</code>	119
<code>\@Glsxtr@p@shortpl@</code>	89	<code>\@glo@autosee</code>	25
<code>\@Glsxtr@p@text@</code>	89	<code>\@glo@autoseehook</code>	49
<code>\@Glsxtrlong</code>	89, 206, 207	<code>\@glo@category</code>	107
<code>\@Glsxtrlongpl</code>	90, 210	<code>\@glo@check@sortallowed</code>	119
<code>\@Glsxtrp</code>	96	<code>\@glo@counterprefix</code>	10, 11, 128
<code>\@Glsxtrpl</code>	56	<code>\@glo@countunit</code>	107
<code>\@Glsxtrshort</code>	89, 205	<code>\@glo@default@sorttype</code>	119
<code>\@Glsxtrshortpl</code>	89, 208	<code>\@glo@desc</code>	40
<code>\@acrlong</code>	90	<code>\@glo@descplural</code>	40
<code>\@acrlongpl</code>	90	<code>\@glo@group</code>	13
<code>\@acrshort</code>	90	<code>\@glo@label</code>	
<code>\@acrshortpl</code>	90		12, 13, 28, 45, 48–50, 81, 88, 388–394
<code>\@addtoreset</code>	153	<code>\@glo@location</code>	12
<code>\@afterheading</code>		<code>\@glo@loclist</code>	12
	370, 371, 381, 383–385, 397–400, 407	<code>\@glo@name</code>	182, 183
<code>\@alt@glshyp@opt</code>	86	<code>\@glo@no@assign@sortkey</code>	123
<code>\@auxout</code>	11, 12, 52, 60, 61, 103,	<code>\@glo@parent</code>	389, 390
	111, 112, 116, 117, 129, 130, 134–137, 408	<code>\@glo@see</code>	45, 46, 49–51
<code>\@bibgls@restoreat</code>	134	<code>\@glo@seealso</code>	48, 49
<code>\@cGLS</code>	106	<code>\@glo@sort</code>	182, 183
<code>\@cGLS@</code>	102, 106, 111	<code>\@glo@sorttype</code>	119, 126
<code>\@cGLSpl</code>	106	<code>\@glo@text</code>	63
<code>\@cGLSpl@</code>	103, 106, 111	<code>\@glo@thislettergrp</code>	143
<code>\@cGls@</code>	102, 111	<code>\@glo@thisvalue</code>	285
<code>\@cGlspl@</code>	102, 111	<code>\@glo@tmp</code>	28, 47, 81
<code>\@cgls@</code>	102, 111	<code>\@glo@type</code>	50, 87, 113, 116,
<code>\@cglspl@</code>	102, 111		120, 123, 126, 127, 129, 130, 133, 134, 140
<code>\@disable@onlypremakeg</code>	117	<code>\@glo@types</code>	171, 387–394
<code>\@do@auxoutstuff</code>	129, 130	<code>\@glossary@default@style</code>	57, 58, 120, 402
<code>\@do@glsg@getcounterprefix</code>	10, 11	<code>\@glossarystyle</code>	120
<code>\@do@glsssee</code>	49, 50	<code>\@gls@</code>	89, 104, 105, 149
<code>\@do@newglossaryentry</code>	114, 198	<code>\@gls@@link</code>	64
<code>\@do@seeglossary</code>	13, 14, 25, 52, 117	<code>\@gls@ReturnAfterFi</code>	129
<code>\@do@wrglossary</code>	67, 148, 149	<code>\@gls@actualchar</code>	183
<code>\@empty</code>	69, 77–80, 121, 128, 183, 184, 201–211	<code>\@gls@adjustmode</code>	68

\@gls@alt@hyp@opt	86	\@gls@long	196, 197
\@gls@alt@hyp@opt@char	86	\@gls@longpl	194, 196–198
\@gls@alt@hyp@opt@keys	86	\@gls@map	179, 180
\@gls@automake	119	\@gls@nameaccess	161–163
\@gls@between	126	\@gls@nohyperlist	41–43
\@gls@checkedmkidx	183, 184, 186	\@gls@noidx@do	127
\@gls@checkmkidxchars	47, 182	\@gls@noidx@getgrouptitle	140
\@gls@codepage	130	\@gls@noidx@nosanitizesort	119
\@gls@counter	9, 11, 23, 66, 68, 84, 148	\@gls@noidx@sanitizesort	119
\@gls@currentlettergroup ...	126, 140, 143	\@gls@noidx@loclist@finalsep	123
\@gls@declareoption	5	\@gls@noidx@loclist@prev	123
\@gls@default@longpl	196, 197	\@gls@noidx@loclist@sep	123
\@gls@deffile	52	\@gls@noref@warn	118, 127
\@gls@doautomake	119, 137	\@gls@org@glsnoidx@displayloc	124
\@gls@doautomake@err	137	\@gls@org@gls@seeformat	124
\@gls@enablesavenonumberlist	51	\@gls@preglossaryhook	120, 187
\@gls@encapchar	184	\@gls@prevlevel	395–397, 401, 402
\@gls@entry@count	103	\@gls@quotechar	183
\@gls@entry@field		\@gls@reference	52, 53, 116, 117
.....	29, 34, 49, 81, 94–97, 102, 139	\@gls@restreat	51
\@gls@entry@unitcount	111, 112	\@gls@saveentrycounter	13, 14, 27, 66, 68, 148
\@gls@field@font	69–76	\@gls@see@noindex	26, 134
\@gls@field@link	69–76, 81, 82	\@gls@setdefault@glslink@opts	
\@gls@firstaccess	161, 162, 164	9, 30, 66, 85
\@gls@getcounterprefix	10, 11	\@gls@setsort	66, 68
\@gls@getgrouptitle	125, 140	\@gls@setupsort@none	14
\@gls@grptitle	87, 126	\@gls@short	197
\@gls@hyp@opt		\@gls@shortaccess	161–164
... ..	81, 82, 86, 106, 145, 149–151, 201–211	\@gls@shortaccesspl	162, 163
\@gls@hyp@opt@cs	86	\@gls@shortpl	194, 197, 198
\@gls@ifaccessattribute@set	163	\@gls@sort	143
\@gls@ifinlist	142	\@gls@textaccess	161–163
\@gls@increment@currcount	102	\@gls@thislabel	69, 100
\@gls@increment@currunitcount	110	\@gls@thisval	179, 180
\@gls@initaccesskeys	196	\@gls@tmp	35, 126
\@gls@keymap ..	12, 13, 28, 45, 48, 81, 136, 179	\@gls@tmpb	186
\@gls@label ..	8, 9, 11, 52, 85, 86, 117, 137, 215	\@gls@type	117–119, 215, 388–394
\@gls@levelchar	183	\@gls@write@entrycounts	103
\@gls@link	30, 62–64, 77–81, 201–211	\@gls@write@entryunitcounts	111
\@gls@link@checkfirsthyper	63, 115	\@gls@write@entryunitcounts@do	112
\@gls@link@label	66, 148	\@gls@writedef	52
\@gls@link@nocheckfirsthyper		\@gls@xref	12, 47
.....	62, 77–80, 201–211	\@gls@abbrv@current@abbreviation	196, 211
\@gls@link@opts	66	\@gls@sacronymlists	113
\@gls@list	126	\@gls@doifexistsorwarn	15, 175, 176, 178, 180
\@gls@local@increment@currcount ...	102	\@gls@entry	52, 103, 112
\@gls@local@increment@currunitcount	110	\@gls@link	67, 87–89
\@gls@location	144	\@gls@localreset	100
\@gls@loclist	123, 124, 144	\@gls@localunset	99, 100

<code>\@glsnextpages</code>	120	<code>\@glsxtr@abbreviationsdef</code>	18, 26
<code>\@glsnonnextpages</code>	120	<code>\@glsxtr@accessdisplay</code>	179–181
<code>\@glsnumberformat</code>		<code>\@glsxtr@activate@initialtagging</code> ..	
..... 9, 11, 66, 68, 84, 148, 179, 182		187, 188
<code>\@glsorder</code>	116	<code>\@glsxtr@addunitcounter</code>	107
<code>\@glspl@</code>	89, 104, 105, 150	<code>\@glsxtr@addunused</code>	51
<code>\@glsplural@</code>	90	<code>\@glsxtr@addunusedxrefs</code>	50, 51
<code>\@gls punc@token</code>	193	<code>\@glsxtr@attrval</code>	
<code>\@glsrecordlocref</code>	10, 11	67, 174, 175, 177, 178, 180, 182
<code>\@glsshowtarget</code>	88	<code>\@glsxtr@autoindex@at</code>	183, 184
<code>\@glsstyle@altlist</code>	369	<code>\@glsxtr@autoindex@doextra@esc</code>	182
<code>\@glsstyle@altlistgroup</code>	370	<code>\@glsxtr@autoindex@encap</code>	182, 184
<code>\@glsstyle@altlisthypergroup</code>	371	<code>\@glsxtr@autoindex@esc</code>	183, 185, 186
<code>\@glsstyle@alttree</code>	385	<code>\@glsxtr@autoindex@escat</code>	183, 184
<code>\@glsstyle@alttreegroup</code>	396	<code>\@glsxtr@autoindex@escencap</code>	184
<code>\@glsstyle@alttreehypergroup</code>	397	<code>\@glsxtr@autoindex@esclevel</code> ...	183, 185
<code>\@glsstyle@index</code>	380	<code>\@glsxtr@autoindex@escquote</code> ...	183, 185
<code>\@glsstyle@indexgroup</code>	381	<code>\@glsxtr@autoindex@level</code>	183, 185
<code>\@glsstyle@indexhypergroup</code>	381	<code>\@glsxtr@autoindex@setname</code>	182
<code>\@glsstyle@inline</code>	379	<code>\@glsxtr@autoindex@crossrefs</code> 14, 15, 45, 48	
<code>\@glsstyle@list</code>	369	<code>\@glsxtr@autoseeindexfalse</code>	14
<code>\@glsstyle@listdotted</code>	368	<code>\@glsxtr@autoseeindextrue</code>	16
<code>\@glsstyle@listgroup</code>	370	<code>\@glsxtr@bookindex@atendgroup</code> .	405–407
<code>\@glsstyle@listhypergroup</code>	370	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	406
<code>\@glsstyle@mc colalttree</code>	400	<code>\@glsxtr@bookindex@atsubsubendgroup</code>	406
<code>\@glsstyle@mc colalttreegroup</code>	401	<code>\@glsxtr@bookindex@between</code>	405, 407
<code>\@glsstyle@mc colalttreehypergroup</code> ..	401	<code>\@glsxtr@bookindex@sep</code>	405, 406
<code>\@glsstyle@mc colalttreesspannav</code>	402	<code>\@glsxtr@bookindex@subatendgroup</code> ..	
<code>\@glsstyle@mc colindexgroup</code>	397	405–407
<code>\@glsstyle@mc colindexhypergroup</code>	397	<code>\@glsxtr@bookindex@subbetween</code> .	405, 406
<code>\@glsstyle@mc colindexspannav</code>	398	<code>\@glsxtr@bookindex@subsep</code>	405, 406
<code>\@glsstyle@mc coltreegroup</code>	398	<code>\@glsxtr@bookindex@subsubatendgroup</code>	
<code>\@glsstyle@mc coltreehypergroup</code>	399	405–407
<code>\@glsstyle@mc coltreenonamegroup</code>	399	<code>\@glsxtr@bookindex@subsubbetween</code> ..	
<code>\@glsstyle@mc coltreenonamehypergroup</code>	400	405, 406
<code>\@glsstyle@mc coltreenonamespannav</code> ..	400	<code>\@glsxtr@bookindex@groupskip</code> ...	405, 407
<code>\@glsstyle@mc coltreesspannav</code>	399	<code>\@glsxtr@cat</code>	101, 113, 147, 148, 187
<code>\@glsstyle@tree</code>	381	<code>\@glsxtr@checkgroup</code>	141
<code>\@glsstyle@treegroup</code>	382	<code>\@glsxtr@counterrecordhook</code>	11
<code>\@glsstyle@treehypergroup</code>	383	<code>\@glsxtr@csname</code>	108–111
<code>\@glsstyle@treenoname</code>	383	<code>\@glsxtr@current@style</code>	57, 402
<code>\@glsstyle@treenonamegroup</code>	384	<code>\@glsxtr@currentunitcount</code>	108, 110
<code>\@glsstyle@treenonamehypergroup</code> ...	384	<code>\@glsxtr@currunitcount</code>	109, 112
<code>\@glstarget</code>	89, 122	<code>\@glsxtr@debugnr</code>	24
<code>\@glstext@</code>	90	<code>\@glsxtr@debugval</code>	24
<code>\@glsunset</code>	99	<code>\@glsxtr@declareoption</code> ...	5, 16, 18, 21, 23
<code>\@glswidestname</code>	387, 395	<code>\@glsxtr@defaultnoglossarywarning</code> ..	21
<code>\@glsxtr</code>	55, 56	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glsxtr@do@wrglossary</code>	117	7, 9, 66, 68, 84, 179, 182

\@glsxtr@defpostpunc	16, 17, 25	\@glsxtr@glossdescfont	174, 175
\@glsxtr@deprecated@abbrstyle	243, 245, 247, 248, 258, 259, 261, 263, 275, 279, 282, 284	\@glsxtr@glossnamefont	175–181
\@glsxtr@dialect	36, 37	\@glsxtr@gobbleto@endescspch	186
\@glsxtr@disabledflycommand	56	\@glsxtr@groupheading	141, 143
\@glsxtr@display@loc	127	\@glsxtr@idx@displaynumberlist	118
\@glsxtr@do@wrindex	85, 86	\@glsxtr@idx@entrynumberlist	118
\@glsxtr@do@gl:disablehyperinlist ..	83	\@glsxtr@ifcsstart	54
\@glsxtr@do@inc@linkcount	153	\@glsxtr@ifpunctoken	193
\@glsxtr@do@record@wrglossary	8, 14	\@glsxtr@ifunitcounter	107
\@glsxtr@do@redef@for@gl:sentries	7	\@glsxtr@insert@dots	195
\@glsxtr@do@style	22, 331	\@glsxtr@insert@dots@next	195
\@glsxtr@do@titlecaps@warn	174–177, 181, 188	\@glsxtr@insertdots	162, 197
\@glsxtr@doabbreviationsdef	18	\@glsxtr@label	32, 51, 154, 173
\@glsxtr@doaccsupp	21, 25	\@glsxtr@loadstyles	367, 368
\@glsxtr@docdefsetting	15, 52	\@glsxtr@local@textformat	66, 67
\@glsxtr@docdefval	15, 51–53	\@glsxtr@locale	36, 37
\@glsxtr@docounterrecord	11	\@glsxtr@longnewglossaryentry	39
\@glsxtr@doglossary	140, 141	\@glsxtr@mark@wordseps	195
\@glsxtr@doiflabelinlist	142	\@glsxtr@mark@wordseps@next	196
\@glsxtr@dolocktag	59, 60	\@glsxtr@markwordseps	196–198
\@glsxtr@dorecord	8, 10	\@glsxtr@mixed@assign@sortkey	119
\@glsxtr@dorecordnodefer	8, 10	\@glsxtr@noidx@displaynumberlist ..	118
\@glsxtr@dosee@alsoindex@glossary ..	14	\@glsxtr@noidx@do	143
\@glsxtr@doseeglossary	13, 25	\@glsxtr@noidx@entrynumberlist	118
\@glsxtr@dostylewarn	215	\@glsxtr@noidx@numberlistloop	118
\@glsxtr@enabletagging	187	\@glsxtr@nomissingglstextnr	21
\@glsxtr@end@	54	\@glsxtr@nomissingglstextval	21
\@glsxtr@endescspch	183–186	\@glsxtr@noop@recordcounter	11, 13
\@glsxtr@entrycount@org@localreset	102	\@glsxtr@nopostpunc	121
\@glsxtr@entrycount@org@localunset	102	\@glsxtr@nopostpunc@postdesc	121
\@glsxtr@entrycount@org@reset	102	\@glsxtr@notfoundinlist	193
\@glsxtr@entrycount@org@unset	102	\@glsxtr@op@recordcounter	14
\@glsxtr@entryunitcount@org@localreset	110	\@glsxtr@optlist	56
\@glsxtr@entryunitcount@org@localunset	110	\@glsxtr@org@@starttoc	311, 312
\@glsxtr@entryunitcount@org@reset ..	110	\@glsxtr@org@GLS@	63
\@glsxtr@entryunitcount@org@unset ..	110	\@glsxtr@org@GLSpl@	63
\@glsxtr@err@undefaction	7, 13	\@glsxtr@org@Gls@	63
\@glsxtr@field@linkdefs	62, 64	\@glsxtr@org@Glspl@	63
\@glsxtr@format@overridefalse	181	\@glsxtr@org@Glsxtrtitlefirst .	313, 314
\@glsxtr@format@overridefalse ..	181, 182	\@glsxtr@org@Glsxtrtitlefirstplural	313, 314
\@glsxtr@foundinlist	193	\@glsxtr@org@Glsxtrtitlefull ..	313, 314
\@glsxtr@full	201	\@glsxtr@org@Glsxtrtitlefullpl	313, 314
\@glsxtr@fullpl	202	\@glsxtr@org@Glsxtrtitlelong ..	313, 314
\@glsxtr@gettype	119	\@glsxtr@org@Glsxtrtitlelongpl	313, 314
		\@glsxtr@org@Glsxtrtitlename ..	312, 314
		\@glsxtr@org@Glsxtrtitleplural	313, 314
		\@glsxtr@org@Glsxtrtitleshort .	312, 314
		\@glsxtr@org@Glsxtrtitleshortpl	312, 314

<code>\@glsxtr@org@Glsxtrtitletext</code> ..	313, 314	<code>\@glsxtr@pagetag</code>	59, 60
<code>\@glsxtr@org@MakeUppercase</code>	312, 314	<code>\@glsxtr@pagetag</code>	59, 60
<code>\@glsxtr@org@checkfirsthyper</code> ...	83, 115	<code>\@glsxtr@prevunitcount</code>	110
<code>\@glsxtr@org@currentfieldvalue</code>	36	<code>\@glsxtr@printglossnr</code>	122
<code>\@glsxtr@org@delimN</code>	60	<code>\@glsxtr@printglossopts</code>	56, 119, 122
<code>\@glsxtr@org@delimR</code>	60	<code>\@glsxtr@printglossval</code>	122
<code>\@glsxtr@org@doseeglossary</code>	25, 117	<code>\@glsxtr@printunsrtglossaryskipentry</code>	140, 141
<code>\@glsxtr@org@gloautosee</code>	26	<code>\@glsxtr@provide@addstoragekey</code>	29
<code>\@glsxtr@org@glolinkprefix</code>	66, 68	<code>\@glsxtr@provide@storagekey</code>	28
<code>\@glsxtr@org@gls@</code>	62	<code>\@glsxtr@record</code> ..	13, 14, 62–64, 68, 201–211
<code>\@glsxtr@org@glsdohypertarget</code>	122	<code>\@glsxtr@record@noglossarywarning</code> ..	14
<code>\@glsxtr@org@glsignore</code>	60	<code>\@glsxtr@record@setting</code>	8, 10, 13, 47, 52, 53, 116
<code>\@glsxtr@org@glspl@</code>	62, 63	<code>\@glsxtr@record@setting@alsoindex</code> ..	8, 10, 47, 116
<code>\@glsxtr@org@glsxtrtitlefirst</code> ..	313, 314	<code>\@glsxtr@record@setting@off</code>	52
<code>\@glsxtr@org@glsxtrtitlefirstplural</code>	313, 314	<code>\@glsxtr@record@setting@only</code>	116
<code>\@glsxtr@org@glsxtrtitlefull</code> ..	313, 314	<code>\@glsxtr@recordsee</code>	14, 25, 47
<code>\@glsxtr@org@glsxtrtitlefullpl</code> ..	313, 314	<code>\@glsxtr@redef@forglsentries</code> ..	7, 26, 27
<code>\@glsxtr@org@glsxtrtitlelong</code> ..	313, 314	<code>\@glsxtr@redefstyles</code>	22, 331
<code>\@glsxtr@org@glsxtrtitlelongpl</code> ..	313, 314	<code>\@glsxtr@reg@glosslist</code>	116–119, 123
<code>\@glsxtr@org@glsxtrtitlename</code> ..	312, 314	<code>\@glsxtr@restore@postpunc</code>	121
<code>\@glsxtr@org@glsxtrtitleorpdforheading</code>	312, 314	<code>\@glsxtr@rglstrigger@record</code> ...	149–151
<code>\@glsxtr@org@glsxtrtitleplural</code> ..	313, 314	<code>\@glsxtr@s@longnewglossaryentry</code>	39
<code>\@glsxtr@org@glsxtrtitleshort</code> ..	312, 314	<code>\@glsxtr@savepreloctag</code>	59–61
<code>\@glsxtr@org@glsxtrtitleshortpl</code> ..	312, 314	<code>\@glsxtr@setentrycountunsetattr</code> ...	101
<code>\@glsxtr@org@glsxtrtitletext</code> ..	313, 314	<code>\@glsxtr@setentryunitcountunsetattr</code> ..	113
<code>\@glsxtr@org@makeglossaries</code>	116	<code>\@glsxtr@setupshortcuts</code>	20, 21, 26
<code>\@glsxtr@org@markboth</code>	311, 312	<code>\@glsxtr@shortcutsnr</code>	20
<code>\@glsxtr@org@markright</code>	311, 312	<code>\@glsxtr@shortcutsval</code>	20, 136
<code>\@glsxtr@org@newacronymstyle</code> ..	114, 115	<code>\@glsxtr@swaptwo</code>	194
<code>\@glsxtr@org@postdescription</code> ..	121, 189	<code>\@glsxtr@tag</code>	187
<code>\@glsxtr@org@see@noindex</code>	134	<code>\@glsxtr@taggingcs</code>	187
<code>\@glsxtr@org@setacronymstyle</code> ..	114, 115	<code>\@glsxtr@textformat</code>	67
<code>\@glsxtr@org@theHvalue</code>	8, 9	<code>\@glsxtr@theHvalue</code> ...	8–10, 65, 66, 68, 148
<code>\@glsxtr@org@unset@buffer</code>	98, 99	<code>\@glsxtr@thevalue</code>	8–10, 65, 66, 68, 148
<code>\@glsxtr@org@prefix</code>	10, 11	<code>\@glsxtr@thisloctag</code>	60
<code>\@glsxtr@org@printglossary</code>	56, 122	<code>\@glsxtr@titlelabel</code>	125, 126, 142, 143
<code>\@glsxtr@org@warndep</code>	194	<code>\@glsxtr@tmp</code>	22, 65, 128
<code>\@glsxtr@p@acrlong@</code>	90	<code>\@glsxtr@type</code>	173
<code>\@glsxtr@p@acrlongpl@</code>	90	<code>\@glsxtr@unitcountlist</code>	107, 108
<code>\@glsxtr@p@acrshort@</code>	90	<code>\@glsxtr@unset</code>	98, 99
<code>\@glsxtr@p@acrshortpl@</code>	90	<code>\@glsxtr@unset@buffer</code>	98, 99
<code>\@glsxtr@p@long@</code>	89	<code>\@glsxtr@unsrt@getgroupptitle</code>	140
<code>\@glsxtr@p@longpl@</code>	90	<code>\@glsxtr@usesee</code>	46
<code>\@glsxtr@p@plural@</code>	89	<code>\@glsxtr@warn@onexistsordo</code>	7, 14
<code>\@glsxtr@p@short@</code>	89	<code>\@glsxtr@warn@undefaction</code>	7, 14
<code>\@glsxtr@p@shortpl@</code>	89		
<code>\@glsxtr@p@text@</code>	89		

\@glxstr@wrglossary@locationhyperlink	\@mfu@nocaplist	188
..... 24	\@one	103, 112, 145
\@glxstr@wrglossnr	\@newglossaryentry@defcounters	101, 109
\@glxstr@wrglossval	\@newglossaryentryposthook
\@glxstrbuffer@nodup@unset 12, 13, 28, 48, 81	
\@glxstrbuffer@unset	\@newglossaryentryprehook
\@glxstrdialecthook 12, 13, 28, 39, 40, 48, 81	
\@glxstrdocdeffalse	\@enil	128, 129, 143
\@glxstrentryfmt	\@ennil	183, 184, 186, 193, 195
\@glxstrfmt	\@no@glxstrindexaliased	84, 85
\@glxstrglossentry	\@no@makeglossaries	133
\@glxstrglossentryother	\@nocounterr	154
\@glxstrhypernameprefix	\@nopostdesc	121
\@glxstrifhasfield	\@onelevel@sanitize	12, 47, 56, 125, 126, 143
\@glxstrifhyphenstart	\@onlypreamble	57,
\@glxstrindexaliased	60, 111, 135, 137, 154, 182, 184, 185, 187	
\@glxstrindexcrossrefsfalse	\@org@glossaryentrynumbers	120, 121
\@glxstrindexcrossrefstrue	\@org@newglossaryentryprehook	39, 40
\@glxstrinmark	\@print@unsrt@glossary	139, 140
\@glxstrlong	\@printgloss@setsort	119, 120
\@glxstrlongpl	\@printglossary	56, 139, 140
\@glxstrnewgls	\@printunsrt@glossary@handler	141
\@glxstrnewgls@inner	\@printunsrtglossary	139
\@glxstrnewgls@innercsname	\@rGLS	150
\@glxstrnotinmark	\@rGLS@	151
\@glxstrp	\@rGLSpl	151
\@glxstrp@opt	\@rGLSpl@	151
\@glxstrpl	\@rGls	150
\@glxstrpostloctag	\@rGls@	150
\@glxstrpreloctag	\@rGlspl	150
\@glxstrsetaliasnoindex	\@rGlspl@	150
\@glxstrshort	\@rgls	149
\@glxstrshortpl	\@rgls@	149
\@glxstrundeftag	\@rglspl	149
\@glxstrwrglossmark	\@rglspl@	149
\@gobble .. 7, 13, 16, 69, 141, 142, 195, 405–407	\@sGlsXtrEnableOnTheFly	54
\@gobbletwo	\@secondofthree	70–
\@ifnextchar	72, 77–80, 82, 202, 203, 205, 207, 209, 210	
\@ifpackageloaded	\@secondoftwo	63, 69,
..... 5, 17, 136,	72–80, 83, 89, 115, 122, 180, 193, 201,	
154, 173, 175, 178, 179, 181, 330, 335, 366	202, 204–211, 225, 247, 261, 283, 312, 314	
\@ifstar	\@sglsxtr@provide@storagekey	28
..... 28,	\@starttoc	312
29, 32, 39, 40, 42, 54, 86, 98, 99, 139, 187	\@thirdofthree	70–72,
\@ifundefined	77–80, 82, 202, 204, 206, 207, 209, 211, 313	
..... 330	\@thirdoftwo	72–76
\@ignored@glossaries	\@this@key	180
..... 41–43	\@tracklang@lang	37
\@input	\@warn@nomakeglossaries	130
..... 134		
\@input@		
..... 129		
\@istfilename		
..... 116		
\@makeglossary		
..... 116, 117		
\@mfu@domakefirsttuc		
..... 187, 188		

\@xdy@main@language	130	\acl	19
\@xdycrossrefhook	47	\ACLP	19
\@xdy@language	130	\Aclp	19
\@xdy@locationclassorder	47	\aclp	19
\\	128	\ACP	19
		\Acp	19
		\acp	19
_	53, 131, 132	\ACRfullfmt	114
		\Acrfullfmt	114
		\acrfullfmt	114
		\ACRfullplfmt	114
		\Acrfullplfmt	114
		\acrfullplfmt	114
		\acronymentry	114
		\acronymfont	77–80, 92, 115
		\acronymname	17
		\acronymsort	114
		\acronymtype	18, 113–115
		\acrpluralsuffix	114, 136
		\ACS	19
		\Acs	19
		\acs	19
		\ACSP	19
		\Acsp	19
		\acsp	19
		\actualchar	185
		\addtolength	396
		\advance	103, 112, 135, 145
		\AF	18
		\Af	18
		\af	18
		\AFP	18
		\Afp	18
		\afp	18
		\AL	18
		\Al	18
		\al	18
		\ALP	18
		\Alp	18
		\alp	18
		amsmath package	23
		\AnyTrackedLanguages	330, 366
		\appto	12, 13, 22, 28, 45, 47–49, 81, 85, 86, 101, 109, 141, 142, 181, 182, 192, 195, 367
		\arabic	23, 407
		\AS	18
		\As	18
		\as	18
		\ASP	18
A			
\AA	349		
\aa	349		
\AB	18		
\Ab	18		
\ab	18		
abbreviation styles:			
long-hyphen-postshort-hyphen	299, 302		
long-hyphen-short-hyphen	296, 300		
long-postshort-user	288		
long-short-user	286		
nolong-short	231		
short	228		
short-hyphen-long-hyphen	304, 305		
short-hyphen-postlong-hyphen	305, 307		
short-long-user	289		
short-nolong	228, 230		
short-nolong-desc	230		
short-postlong-user	290		
\abbreviationsname	17		
\abbrvpluralsuffix			
	136, 163, 197, 218, 220, 223, 225, 227, 229, 231, 235, 237, 238, 240, 242, 244, 246, 248, 249, 251, 253, 254, 256, 258, 260, 262, 263, 265, 267, 269, 270, 272, 274, 275, 277, 279, 281, 283, 286, 287, 289, 292, 295, 297, 301, 303, 306, 309		
\ABP	18		
\Abp	18		
\abp	18		
\AC	19		
\Ac	19		
\ac	19		
\ACF	19		
\Acf	19		
\acf	19		
\ACFP	19		
\Acfp	19		
\acfp	19		
\ACL	19		
\Acl	19		

<code>\Asp</code>	18	<code>\cGls</code>	18, 19, 101, 112
<code>\asp</code>	18	<code>\cglS</code>	18, 19, 101, 112
<code>\AtBeginDocument</code>	24, 27, 58, 136	<code>\cGLSformat</code>	105
<code>\AtEndDocument</code>	50, 103, 111, 129, 130	<code>\cGlsformat</code>	104
B			
babel package	182, 184, 192	<code>\cglSformat</code>	104, 106
<code>\begin</code>	126, 131, 132,	<code>\cGLSpl</code>	18, 19, 101, 112
	140, 369, 371–378, 381, 383, 398–402, 405	<code>\cGlspl</code>	18, 19, 101, 112
<code>\begingroup</code>	8, 9, 30, 84, 138–140, 153	<code>\cglSpl</code>	18, 19, 101, 112
<code>\bgroup</code>	39, 40, 120	<code>\cGLSplformat</code>	105
<code>bib2gls</code> ...	23, 30, 143, 148, 149, 330, 331, 333	<code>\cGLsplformat</code>	105
C			
<code>\c@wrglossary</code>	23	<code>\cglSplformat</code>	104, 106
<code>\catcode</code>	51, 134	<code>\changes</code>	317, 381, 383
category attributes:		<code>\char</code>	125
<code>accessinsertdots</code>	162	<code>\columnwidth</code>	58
<code>apospplural</code>	197	<code>\count@</code>	103, 112
<code>discardperiod</code>	191	<code>\csappto</code>	38
<code>entrycount</code>	98, 101, 103, 112, 113	<code>\csdef</code>	28, 34, 38,
<code>externallocation</code>	333		39, 81, 82, 102, 107, 108, 111, 168, 179,
<code>firstshortaccess</code>	164		189, 190, 215–217, 224, 247, 261, 283,
<code>firstuc</code>	177		287–289, 291, 300, 302, 305, 307, 368, 386
<code>glossdesc</code>	173	<code>\cseappto</code>	43
<code>glossdescfont</code>	174	<code>\csedef</code>	109
<code>glossname</code>	175	<code>\csgdef</code>	35, 41–
<code>glossnamefont</code>	175, 178		43, 53, 59, 102, 103, 108, 110, 111, 386, 408
<code>headuc</code>	314	<code>\cslet</code>	34, 40, 120
<code>indexname</code>	182	<code>\csletcs</code>	34, 216, 217
<code>indexonlyfirst</code>	85	<code>\csname</code>	6, 29, 42, 47, 52, 57,
<code>insertdots</code>	197		61, 63, 66, 68, 77–82, 84, 93, 108, 109,
<code>linkcount</code>	152		117, 126, 129, 130, 133, 134, 140, 145,
<code>linkcountmaster</code>	153		147, 148, 153, 173, 194, 201–211, 217, 395
<code>markshortwords</code>	197	<code>\cspreto</code>	39
<code>markwords</code>	196, 198, 294, 295, 303	<code>\csuse</code>	9, 30,
<code>nameshortaccess</code>	163		31, 42, 50, 60, 81, 82, 95, 96, 107–111,
<code>nohyper</code>	83		120, 125–127, 137, 139, 142–144, 168,
<code>nohyperfirst</code>	70–72		179, 189, 190, 215, 217, 386, 387, 389, 390
<code>noshortplural</code>	197	<code>\csxdef</code>	45, 48, 108, 111
<code>regular</code>	61, 106, 217, 218, 220, 222–	<code>\currentglossary</code>	120, 138, 139, 407
	224, 227–234, 236–240, 243–245, 247,	<code>\CurrentOption</code>	24, 367, 368
	250–253, 255, 257, 259, 261, 264–268,	<code>\CurrentTrackedLanguage</code>	365
	270–272, 275–278, 280, 281, 283, 286,	<code>\CurrentTrackedLanguageTag</code>	136
	291–293, 295–297, 299, 303, 304, 308, 310	<code>\CurrentTrackedScript</code>	365
<code>textformat</code>	67	<code>\CurrentTrackedTag</code>	330, 365
<code>textshortaccess</code>	163	<code>\CustomAbbreviationFields</code>	
<code>\cdot</code>	24		198, 218–222, 224,
<code>\centering</code>	404		226, 228, 231, 233, 235, 236, 238, 240,
<code>\cGLS</code>	18, 19, 101, 112		242, 245, 247, 249–252, 254, 256, 259,
			261, 263–266, 268–271, 273, 275, 279,
			281, 282, 285, 286, 288, 289, 291–293,
			295–297, 299, 300, 302–305, 307, 308, 310

D

`\DeclareAcronymList` 113
`\DeclareOption` 5, 367
`\DeclareOptionX` 5, 24
`\def` 9, 10, 12–15, 25, 27,
 30, 33, 40, 42, 46–48, 51, 54–56, 59, 61–
 66, 68–80, 86, 88–93, 99, 104–106, 113,
 117, 119–123, 125–129, 140, 143, 145,
 148–151, 161–163, 183–188, 192–197,
 201–211, 215, 294, 297, 299, 300, 303,
 305, 333, 365, 379, 380, 395–397, 401, 402
`\defglentryfmt` 41–43
`\define@boolkey` 15, 16, 65, 83
`\define@choicekey`
 7, 13, 15, 16, 20, 21, 24, 64, 122
`\define@key` 12, 13,
 16, 22, 28, 48, 65, 68, 81, 122, 161, 162, 194
`\DefineAcronymSynonyms` 20
`\delimN` 60
`\delimR` 60
`\detokenize` 54
`\dimen@` 115, 386–394
`\dimen@i` 389, 390
`\dimen@ii` 386, 387, 389, 390
`\dimexpr` 58, 385
`\disable@keys` 17, 27, 53, 134
`\do` 6, 22, 32, 51, 69, 100, 101, 113,
 116, 119, 126, 140, 147, 154, 173, 179, 187
`\do@gl@link@checkfirsthyper`
 30, 62–64, 66, 77–80, 201–211
`\do@gl@disablehyperinlist` 66, 84
`doc package` 185
`\dolistcsloop` 31
`\DTLifinlist` 117, 118, 123
`\DTLifint` 125

E

`\eappto`
 11, 22, 41–43, 141, 143, 162–164, 182, 367
`\edef` 6, 8–11, 36, 41–44, 47,
 49–52, 66–68, 83, 84, 87, 88, 107, 108,
 110, 116–118, 123, 125, 127–130, 134,
 138, 139, 148, 153, 174, 175, 177–180,
 183, 184, 186, 194, 365, 389, 390, 405, 406
`\eglssetwidest` 388–394
`\egroup` 40, 121
`\else` 8–12,
 15, 16, 18, 20, 21, 25, 26, 30, 33, 38, 51,
 53, 54, 59, 61, 63, 67, 84, 85, 104, 115,

 116, 119–122, 125, 127–129, 131, 133,
 135, 148, 149, 181–184, 186, 193, 195–
 197, 204–211, 214, 218, 219, 221, 223–
 227, 229–233, 235–237, 239–248, 250–
 262, 264–267, 269–284, 286–288, 290,
 293, 294, 297, 300–303, 305, 307, 309,
 369, 371–382, 384, 395, 396, 402, 404, 406
`\emph` 263
`\empty` 127, 128
`\encapchar` 185
`\end` 122, 127,
 131, 132, 141, 369, 371–378, 398–402, 405
`\end@gl@xtr@display@loc` 127
`\endcsname` 6, 29, 42, 47, 52, 57,
 61, 63, 66, 68, 77–82, 84, 93, 108, 109,
 117, 126, 129, 130, 133, 134, 140, 145,
 147, 148, 153, 173, 194, 201–211, 217, 395
`\endgroup` 8, 10, 30, 84, 138–140, 153
`\ensuremath` 24
`entry categories:`
 `abbreviation` 211
 `general` 168, 170
 `index` 171
`\epreto` 183
`\equal` 133, 190
`etoolbox package` 5
`\expandafter` 24, 29, 30, 32, 36, 45,
 46, 50, 51, 54–56, 65, 81, 82, 86, 93, 99,
 106, 107, 117–119, 123, 126, 128, 140,
 141, 143, 145, 152, 163, 173, 176, 177,
 180–182, 185, 186, 193, 196–198, 294, 405
`\expandonce`
 114, 143, 162–164, 183, 184, 219, 297
`\ExtraCustomAbbreviationFields`
 162–164, 196, 198

F

`\fi` 7–12, 14–16, 18, 20, 21, 24–26, 30,
 33, 38, 45, 47, 48, 50, 52–54, 57–59, 61,
 64, 67, 84, 85, 103, 104, 111, 112, 115,
 119–122, 125, 127–131, 133–135, 137,
 148, 149, 182–184, 186, 193, 195, 196,
 198, 204–211, 214, 218, 219, 221, 223–
 227, 229–233, 235–237, 239–248, 250–
 262, 264–267, 269–284, 286–288, 290,
 293, 294, 297, 300–303, 305, 307, 309,
 369, 372–382, 384, 386–396, 402, 404, 407
`first use` 409
`flag` 409
`text` 409

<code>\firstacronymfont</code>	115	<code>indexhypergroup</code>	381
<code>fontspec</code> package	136	<code>inline</code>	379
<code>\footnote</code>	222	<code>list</code>	369
<code>\forall glossaries</code> 50, 140, 171, 173, 388–394		<code>listdotted</code>	368
<code>\forall glossentries</code>	52, 103, 112	<code>listdottedstyle</code>	369
<code>\ForEachTrackedDialect</code>	331, 366	<code>listgroup</code>	370
<code>\foreignlanguage</code>	36, 37	<code>listhypergroup</code>	370
<code>\forall glossentries</code>	6, 50, 171, 173, 388–394	<code>mcolalttree</code>	400
<code>\forall listcsloop</code>	32, 112, 127	<code>mcolalttreegroup</code>	401
<code>\forall listloop</code>	99, 123, 124, 188	<code>mcolalttreehypergroup</code>	401
<code>\futurelet</code>	193	<code>mcolalttreespannav</code>	402
G			
<code>\gdef</code>	60, 184, 185	<code>mcolindexgroup</code>	397
<code>\Genacrfullformat</code>	114	<code>mcolindexhypergroup</code>	397
<code>\genacrfullformat</code>	114	<code>mcolindexspannav</code>	398
<code>\GenericAcronymFields</code>	114	<code>mcoltreegroup</code>	398
<code>\Genplacrfullformat</code>	114	<code>mcoltreehypergroup</code>	399
<code>\genplacrfullformat</code>	114	<code>mcoltreenonamegroup</code>	399
<code>\GetTrackedDialectFromLanguageTag</code> ..	36	<code>mcoltreenonamehypergroup</code>	400
<code>\GetTrackedDialectToMapping</code>	36	<code>mcoltreenonamespannav</code>	400
<code>\glo@grabfirst</code>	143	<code>mcoltreespannav</code>	399
<code>\glo@name</code>	176, 177, 181	<code>sublistdotted</code>	369
<code>\gloaliaslabel</code>	88	<code>tree</code>	381
<code>\global</code>	10, 40, 52, 121, 144	<code>treegroup</code>	382
<code>\glo linkprefix</code>	65–68, 88, 122, 136	<code>treehypergroup</code>	383
<code>glossaries</code> package		<code>treenoname</code>	383
..... 14, 25, 26, 38, 45, 47–49, 119, 368		<code>treenonamegroup</code>	384
<code>glossaries-accsupp</code> package	21, 25, 154, 198	<code>treenonamehypergroup</code>	384
<code>glossaries-extra</code> package	2, 366	<code>glossary-bookindex</code> package	368
<code>glossaries-extra-bib2gls</code> package 14, 27, 330, 366		<code>glossary-hypernav</code> package	87
<code>glossaries-extra-stylemods</code> package .	21, 189, 331	<code>glossary-long</code> package	373
<code>glossaries-stylemods</code> package	403	<code>glossary-longbooktabs</code> package	373
<code>glossaries.sty</code> package	40	<code>glossary-tree</code> package	379, 380
<code>\GlossariesExtraWarning</code> . 6, 16, 21, 36–		<code>\glossaryentrynumbers</code> ...	61, 120, 121, 144
39, 54, 56, 67, 115, 118, 128, 131, 134,		<code>\glossaryheader</code>	127,
140, 174, 175, 177, 178, 180, 187, 188, 217		140, 369–378, 380–384, 395, 397–401, 405	
<code>\GlossariesExtraWarningNoLine</code> 16, 103, 112		<code>\glossaryname</code>	120
<code>\GlossariesWarning</code> 59, 118, 120, 123, 124, 215		<code>\glossarypostamble</code>	127, 141, 142
<code>\GlossariesWarningNoLine</code>	117, 130	<code>\glossarypreamble</code>	126, 140
<code>glossary</code> styles:		<code>\glossarysection</code> 126, 127, 133, 134, 140, 142	
<code>altlist</code>	369	<code>\glossarytitle</code> 42, 120, 126, 127, 133, 134, 140	
<code>altlistgroup</code>	370	<code>\glossarytoctitle</code>	
<code>altlisthypergroup</code>	371 42, 120, 126, 127, 133, 134, 140	
<code>alttree</code>	385, 386, 395	<code>\glossentry</code>	
<code>alttreegroup</code>	396	120, 144, 368–378, 380, 382, 384, 395, 405	
<code>alttreehypergroup</code>	397	<code>\glossentrydesc</code>	
<code>index</code>	380 368, 369, 371–378, 381–383, 385	
<code>indexgroup</code>	381	<code>\glossentryname</code>	
		138, 368–378, 380, 382, 384, 395, 396, 403	
		<code>\glossentrynameother</code>	139

<code>\glossentrysymbol</code>	372–376, 378, 382, 383, 385	<code>\Glsaccessfirst</code>	70
<code>\glossxtrsetpopts</code>	189	<code>\glsaccessfirst</code>	70
<code>\glostyle</code>	381, 383	<code>\GLSaccessfirstplural</code>	72
<code>\GLS</code>	101, 112, 147	<code>\Glsaccessfirstplural</code>	72
<code>\Gls</code>	55, 101, 112, 147	<code>\glsaccessfirstplural</code>	71
<code>\gls</code>	38, 55, 57, 101, 112, 118, 131, 147	<code>\Glsaccesslong</code>	79, 199, 207, 219, 227, 232, 236, 241–244, 250, 256–258, 264, 266, 273, 274, 276, 278–280, 286–288, 296, 298, 301, 304, 309
<code>\gls@assign@desc</code>	40	<code>\glsaccesslong</code>	79, 199, 206, 207, 218, 221, 223, 224, 226, 227, 229, 230, 232, 233, 235, 237, 239–246, 248, 250–262, 264, 265, 267, 269–282, 284, 286, 287, 290, 293, 296–298, 301, 304, 309
<code>\gls@assign@field</code>	12, 13, 28, 81	<code>\Glsaccesslongpl</code>	80, 199, 211, 219, 227, 232, 236, 241–244, 250, 256–259, 264, 266, 273–276, 278–280, 286–288, 296, 298, 301, 302, 304, 309
<code>\gls@checkseeallowed</code>	52–54, 117	<code>\glsaccesslongpl</code>	80, 199, 210, 211, 219, 221, 223, 224, 226, 227, 229, 230, 232, 233, 235, 237, 239–248, 250, 252–262, 264, 266, 267, 269, 271–282, 284, 286–288, 290, 293, 296, 298, 301, 304, 309
<code>\gls@codepage</code>	130	<code>\GLSaccessname</code>	72
<code>\gls@defdocnewglossaryentry</code>	51, 102, 109	<code>\Glsaccessname</code>	72
<code>\gls@defglossaryentry</code>	40, 52, 55, 56	<code>\glsaccessname</code>	46, 72
<code>\gls@dotoctitle</code>	120	<code>\GLSaccessplural</code>	71
<code>\gls@glossary</code>	47	<code>\Glsaccessplural</code>	71
<code>\gls@grplabel</code>	87	<code>\glsaccessplural</code>	71
<code>\gls@ifnotmeasuring</code>	100	<code>\Glsaccessshort</code>	77, 205, 214, 221, 223–226, 229, 230, 237, 239, 240, 246, 248, 252, 253, 255, 260–262, 267, 269, 271, 272, 282–284, 290, 293, 301, 306, 307
<code>\gls@level</code>	144	<code>\glsaccessshort</code>	77, 199, 204, 206, 214, 219, 221, 223–227, 229, 230, 232, 235–237, 239–241, 243, 244, 246, 248, 250, 251, 253–258, 260, 262, 264–267, 269–274, 276, 278–284, 286, 288, 290, 293, 296, 298, 301, 304, 306, 307, 309
<code>\gls@noidxglossary</code>	117	<code>\Glsaccessshortpl</code>	78, 209, 214, 221, 223–226, 229, 231, 237, 239, 240, 246–248, 252, 253, 255, 260–262, 267, 269, 271, 272, 282–284, 290, 293, 301, 306, 307, 310
<code>\gls@org@glossaryentryfield</code>	120	<code>\glsaccessshortpl</code>	78, 199, 208, 209, 214, 219, 221, 223–227, 229, 230, 232, 235–237, 239–244, 246, 248, 250, 251, 253–260, 262, 264, 266, 267, 269–276, 278, 280–284, 286, 288, 290, 293, 296, 298, 301, 304, 306, 307, 309
<code>\gls@org@glossarysubentryfield</code>	120		
<code>\gls@orgTrackLangRequireDialectPrefix</code>	365		
<code>\gls@save@numberlist</code>	59, 61		
<code>\gls@set@xr@key</code>	48		
<code>\gls@tmplen</code>	388–394, 396		
<code>\gls@type</code>	117		
<code>\glsabbrvdefaultfont</code>	200, 218, 221, 223, 225, 227, 229, 232, 285, 295, 297, 308		
<code>\glsabbrvfont</code>	263–272, 274, 275, 277, 279, 281–283		
<code>\glsabbrvfont</code>	90, 91, 115, 200, 204–206, 208, 209, 211, 214, 218–229, 232, 233, 235, 237, 238, 240, 242, 244, 246, 248, 249, 251, 253, 254, 256, 258, 260, 262, 263, 265, 267, 269, 270, 272, 274, 275, 277, 279, 281, 283, 286, 287, 289, 291–293, 295, 297, 300, 301, 303, 306, 309		
<code>\glsabbrvhyphenfont</code>	295, 296, 300–307		
<code>\glsabbrvonlyfont</code>	308–310		
<code>\glsabbrvscfont</code>	234–238, 240, 242, 244–248		
<code>\glsabbrvsmfont</code>	249–254, 256, 258–262		
<code>\glsabbrvuserfont</code>	285–292		
<code>\GLSaccessdesc</code>	73		
<code>\Glsaccessdesc</code>	73, 174, 186		
<code>\glsaccessdesc</code>	73, 174, 191		
<code>\GLSaccessdescplural</code>	73		
<code>\Glsaccessdescplural</code>	73		
<code>\glsaccessdescplural</code>	73		
<code>\GLSaccessfirst</code>	71		

<code>\GLSaccesssymbol</code>	74	<code>\glsdoifexistsorwarn</code>	15, 173, 174, 186
<code>\Glsaccesssymbol</code>	74, 186	<code>\glsdoifnoexists</code>	39, 40
<code>\glsaccesssymbol</code>	74, 186, 191	<code>\glsdonohyperlink</code>	67, 89
<code>\GLSaccesssymbolplural</code>	74	<code>\glsdosanitizesort</code>	118
<code>\Glsaccesssymbolplural</code>	74	<code>\glsenableentrycount</code>	101, 103, 111
<code>\glsaccesssymbolplural</code>	74	<code>\glsenableentryunitcount</code>	103, 112
<code>\GLSaccessstext</code>	70	<code>\glsentrycounter</code>	24, 128
<code>\Glsaccessstext</code>	70	<code>\GlsEntryCounterLabelPrefix</code>	38
<code>\glsaccessstext</code>	46, 69	<code>\glsentrycurrcount</code>	102, 103, 109
<code>\glsacrshortcutstrue</code>	20, 21	<code>\Glsentrydesc</code>	158, 166, 175
<code>\glsacspacemax</code>	115	<code>\glsentrydesc</code>	158, 159, 166, 175
<code>\glsadd</code>	30, 51, 69, 131	<code>\Glsentrydescplural</code>	159, 166
<code>\glsadd options</code>		<code>\glsentrydescplural</code>	159, 166
<code>theHvalue</code>	9	<code>\Glsentryfirst</code>	107, 152, 156, 165
<code>thevalue</code>	9, 10	<code>\glsentryfirst</code> ..	106, 151, 156, 165, 326, 327
<code>\glsaddpostsetkeys</code>	10, 68	<code>\Glsentryfirstplural</code> ...	107, 152, 157, 165
<code>\glsaddpresetkeys</code>	10, 68	<code>\glsentryfirstplural</code>	
<code>\glsaddstoragekey</code>	49, 168, 333	107, 151, 156, 157, 165, 327
<code>\glsbackslash</code>	54	<code>\glsentryfmt</code>	41–43
<code>\glscapscase</code>	63, 69–80, 82, 201–213	<code>\Glsentryfull</code>	114
<code>\glscategory</code>		<code>\glsentryfull</code>	114
..	61, 69, 83, 90, 91, 169, 170, 173–175,	<code>\Glsentryfullpl</code>	114
	177–180, 186, 189, 190, 201–205, 208, 209	<code>\glsentryfullpl</code>	114
<code>\glscategorylabel</code>		<code>\glsentryitem</code>	
..	83, 162–164, 194, 196–198, 224, 247,	138, 139, 368–378, 380, 382, 384, 395, 406
	261, 283, 287–289, 291, 300, 302, 305, 307	<code>\Glsentrylong</code>	91, 92, 107, 152, 161, 167
<code>\glsclsebrace</code>	47, 132, 133	<code>\glsentrylong</code> .	91, 92, 106, 151, 160, 161,
<code>\glscounter</code>	9	167, 225, 247, 261, 283, 289, 291, 305, 328
<code>\glscurrententrylabel</code>		<code>\Glsentrylongpl</code>	92, 107, 161, 167
.....	59, 60, 120, 129, 138–141, 188, 189	<code>\glsentrylongpl</code> ...	92, 93, 107, 161, 167, 328
<code>\glscurrentfieldvalue</code>	30–33, 35, 36, 284, 332	<code>\Glsentrylongplural</code>	152
<code>\glscustomtext</code> ..	62–64, 77–80, 201–212, 214	<code>\glsentrylongplural</code>	151
<code>\glsdefaulttype</code>	6,	<code>\Glsentryname</code>	154, 164, 176–179
	17, 38, 39, 119, 120, 131, 139, 140, 142	<code>\glsentryname</code>	
<code>\glsdescriptionaccessdisplay</code>	158, 159, 174	138, 154, 164, 183, 325, 388–394, 408
<code>\glsdescriptionpluralaccessdisplay</code>	159	<code>\glsentrynumberlist</code>	118, 125, 392–394
<code>\glsdescwidth</code>	371–378	<code>\Glsentryplural</code>	155, 165
<code>\glsdetoklabel</code>		<code>\glsentryplural</code>	155, 156, 165, 326
.....	8, 9, 28, 31–35, 38–40, 44–46,	<code>\glsentryprevcount</code>	102, 104, 110
	50, 52–54, 66, 68, 84, 88, 102, 103, 108–	<code>\glsentryprevmaxcount</code>	110
	112, 117, 120, 123, 124, 138, 139, 143,	<code>\glsentryprevtotalcount</code>	110
	144, 147, 148, 173, 176, 177, 181, 389, 390	<code>\Glsentryshort</code>	91, 92, 160, 167
<code>\glsdisplaynumberlist</code>	118, 123	<code>\glsentryshort</code>	90–92, 115,
<code>\glsdohyperlink</code>	87, 89	159, 160, 166, 167, 287, 288, 300, 323, 324
<code>\glsdohypertarget</code>	89, 122	<code>\Glsentryshortpl</code>	91, 92, 160, 167
<code>\glsdoifexists</code>	15,	<code>\glsentryshortpl</code>	91, 92, 160, 167, 324
	25, 34, 39, 43, 45–47, 62, 63, 68, 77–80,	<code>\Glsentrysymbol</code>	157, 166
	88, 100, 117, 123, 124, 138, 139, 201–211	<code>\glsentrysymbol</code>	157, 165, 166, 391–393
<code>\glsdoifexistsordo</code>	30, 31, 64	<code>\Glsentrysymbolplural</code>	158, 166

<code>\glsentrysymbolplural</code>	158, 166	272, 274, 275, 277, 279, 281, 283, 286,
<code>\Glsentrytext</code>	155, 165	287, 289, 292, 295, 297, 301, 303, 306, 309
<code>\glsentrytext</code>	88, 155, 165, 325, 326	<code>\glsfirstlongfootnotefont</code>
<code>\glsentrytype</code>	138 223–226, 245–248, 259–262, 281–284
<code>\Glsentryuseri</code>	75	<code>\glsfirstlonghyphenfont</code> 294–297, 299–306
<code>\glsentryuseri</code>	75	<code>\glsfirstlongonlyfont</code>
<code>\Glsentryuserii</code>	75	308–310
<code>\glsentryuserii</code>	75	<code>\glsfirstlonguserfont</code>
<code>\Glsentryuseriii</code>	75	285–293
<code>\glsentryuseriii</code>	75	<code>\GLSfirstplural</code>
<code>\Glsentryuseriv</code>	76	319, 320
<code>\glsentryuseriv</code>	76	<code>\Glsfirstplural</code>
<code>\Glsentryuserv</code>	76	320
<code>\glsentryuserv</code>	76	<code>\glsfirstplural</code>
<code>\Glsentryuservi</code>	76	319
<code>\glsentryuservi</code>	76	<code>\glsfirstpluralaccessdisplay</code> ..
<code>\glsextrapostnamehook</code>	179	156, 157
<code>\glsfieldfetch</code>	88	<code>\glsforeachincategory</code>
<code>\glsfieldxdef</code>	173	215
<code>\glsfindwidesttoplevelname</code>	387	<code>\glsgenentryfmt</code>
<code>\GLSfirst</code>	318, 319	61
<code>\Glsfirst</code>	319	<code>\glsgetattribute</code> 67, 87, 88, 104, 108–110,
<code>\glsfirst</code>	318, 319	129, 148, 153, 174, 175, 177, 178, 180, 182
<code>\glsfirstabbrvdefaultfont</code>		<code>\glsgetcategoryattribute</code>
200, 218, 221, 223, 225, 227, 229, 232, 297		169
<code>\glsfirstabbrvmfont</code>	263–284	<code>\glsgetgroupitle</code>
<code>\glsfirstabbrvfont</code> ..	115, 199, 218–221, 370, 371, 381, 383–385, 396–402
223–232, 235, 237, 238, 240, 242, 244,		<code>\glsgetwidestname</code>
246, 248, 249, 251, 253, 254, 256, 258,		386
260, 262, 263, 265, 267, 269, 270, 272,		<code>\glsgroupheading</code>
274, 275, 277, 279, 281, 283, 286, 287,	 143, 369–378, 380–385, 395–402, 407
289, 292, 295, 297, 298, 301, 303, 306, 309		<code>\glsgroupskip</code>
<code>\glsfirstabbrvhyphenfont</code>		143, 369, 371–379, 381, 382, 384, 396, 407
..... 294–296, 300, 301, 303–307		<code>\glshasattribute</code>
<code>\glsfirstabbrvonlyfont</code>	309, 310	67, 87, 103,
<code>\glsfirstabbrvscfont</code>	235–248	104, 108, 110, 112, 129, 148, 153, 174,
<code>\glsfirstabbrvsmfont</code>	249–262	175, 177, 178, 180, 182, 218, 220, 222,
<code>\glsfirstabbrvuserfont</code>	285–293	223, 225, 228, 230, 231, 233–238, 245,
<code>\glsfirstaccessdisplay</code>	156	247, 249–252, 259, 261, 263–270, 277,
<code>\glsfirstlongdefaultfont</code>		280, 281, 283, 286, 287, 289, 291–293,
218, 221, 227, 229, 232, 235–245, 249–		295–297, 299, 300, 302–304, 306–308, 310
259, 263, 264, 267, 268, 270–275, 277, 278		<code>\glshascategoryattribute</code>
<code>\glsfirstlongemfont</code>		169
..... 265, 266, 268–270, 275–277, 279, 280		<code>\glshex</code> ..
<code>\glsfirstlongfont</code>	199, 218–223,	336–342, 344–350, 352–355, 357–365
225, 227, 229–235, 237, 239, 240, 242,		<code>\glshyperlink</code>
244, 246, 248, 250, 251, 253, 254, 256,		88, 332
258, 260, 262, 264, 265, 267, 269, 270,		<code>\glshypernavsep</code>
		126
		<code>\glshypernumber</code>
		129, 181, 182, 333
		<code>\glsifattribute</code>
	 64, 65, 70, 83, 85, 94, 152, 171,
		174–177, 180, 181, 188, 191, 192, 314–323
		<code>\glsifcategory</code>
		171
		<code>\glsifcategoryattribute</code>
	 83, 162–164, 169, 170, 196–198
		<code>\glsifnotregular</code>
		69
		<code>\glsifnotregularcategory</code>
		170
		<code>\glsifplural</code> 63, 69, 71–74, 77–80, 192, 201–213
		<code>\glsifregular</code>
		61, 69, 106, 107, 151, 152
		<code>\glsifregularcategory</code>
		170
		<code>\glsifusetranslator</code>
		42
		<code>\glsignore</code>
		60
		<code>\glsinlinedescformat</code>
		379

<code>\glsinlinesubdescformat</code>	379	274, 275, 277, 279, 281, 283, 286, 287,
<code>\glsinsert</code>	63,	289, 292, 295, 297, 301, 303, 306, 309, 310
	69, 77–80, 201–213, 294, 300, 302, 305–307	
<code>\glskeylisttok</code>	113, 114, 196, 198	<code>\glslongfootnotefont</code>
<code>\glslabel</code>	8, 9, 30, 44,	222, 223, 225, 246, 248, 260, 262, 281, 283
	46, 61, 64–67, 83, 84, 87, 88, 115, 148,	<code>\glslonghyphenfont</code>
	152, 153, 190, 191, 212, 213, 225, 247, 295, 297, 299–301, 303, 305, 306
	261, 283, 287–289, 291, 300, 302, 305–307	<code>\glslongonlyfont</code>
<code>\glslabeltok</code>	113,	308, 309
	114, 196, 198, 218–223, 225, 226, 228,	<code>\glslongpltok</code>
	230, 231, 233–238, 240, 242, 245, 247,	. 198, 218–220, 222, 223, 231, 234–238,
	249–254, 256, 259–261, 263–270, 272,	242, 245, 249–252, 256, 259, 263–268,
	274, 275, 277, 279–281, 283, 285–293,	270, 274, 275, 279, 281, 285, 286, 288–
	295–297, 299, 300, 302–304, 306–308, 310	293, 295–297, 299, 300, 302–304, 308, 310
<code>\glsletentryfield</code>	183	<code>\glslongpluralaccessdisplay</code>
<code>\glslink</code>	114	161
<code>\glslink options</code>		<code>\glslongtok</code> .. 113, 114, 196–198, 218–224,
counter	10	226, 228, 231, 233–238, 240, 242, 245,
format	181	247, 249–254, 256, 259, 261, 263–270,
hyper	311	272–275, 279, 281, 282, 285–289, 291–
hyperoutside	65	293, 295–297, 299, 300, 302–305, 308, 310
noindex	8, 9, 83, 311	<code>\glslonguserfont</code>
textformat	67	285–290, 292
theHvalue	66	<code>\glsmcols</code>
thevalue	66, 145	398–402
wrgloss	8, 64	<code>\GLSname</code>
<code>\glslinkcheckfirsthyperhook</code>	83	316
<code>\glslinkpostsetkeys</code>	10, 66, 148	<code>\Glsname</code>
<code>\glslinkpresetkeys</code>	10, 66, 148	316, 317
<code>\glslinkvar</code>	86	<code>\glsname</code>
<code>\glslistchildpostlocation</code>	369	316
<code>\glslistchildprelocation</code>	369, 370	<code>\glsnameaccessdisplay</code>
<code>\glslistdesc</code>	369, 370	154, 176, 178
<code>\glslistdottedwidth</code>	368	<code>\glsnamefont</code>
<code>\glslistgroupheaderfmt</code>	370, 371	175–180
<code>\glslistnavigationitem</code>	370, 371	<code>\glsnavhyperlink</code>
<code>\glslistprelocation</code>	369, 370	126
<code>\glslocalunset</code>	63, 149	<code>\glsnavhyperlinkname</code>
<code>\glslongaccessdisplay</code>	160, 161	87
<code>\glslongdefaultfont</code> .. 200, 218, 221, 222,		<code>\glsnavhypertarget</code>
227, 229, 232, 235, 237, 239, 240, 242,	 370, 371, 381, 383, 385, 397–402
244, 250, 251, 253, 254, 256–258, 264,		<code>\glsnavigation</code>
267, 270, 272, 274, 277, 278, 285, 295, 308	 370, 371, 381, 383, 385, 397–402
<code>\glslongemfont</code> 263, 265, 268, 269, 275, 276, 279		<code>\glsnextpages</code>
<code>\glslongfont</code> 91, 92, 200, 206, 207, 210–212,		120
218, 219, 221, 223, 225, 227, 229, 231,		<code>\glsnoidxdisplayloc</code>
232, 234, 235, 237, 239, 240, 242, 244,		124
246, 248, 250, 251, 253, 254, 256, 258,		<code>\glsnoidxdisplaylocloclisthandler</code>
260, 262, 264, 265, 267, 269, 270, 272,		123
		<code>\glsnoidxloclist</code>
		124, 144
		<code>\glsnoidxnumberlistloophandler</code>
		124
		<code>\glsnonextpages</code>
		120
		<code>\glsnonumberlistfalse</code>
		59
		<code>\glsnonumberlisttrue</code>
		59
		<code>\glsnopostdotfalse</code>
		121
		<code>\glsnopostdottrue</code>
		121
		<code>\glsnumberlistloop</code>
		118
		<code>\glsnumlistlastsep</code>
		123
		<code>\glsnumlistsep</code>
		123
		<code>\glsopenbrace</code>
		47, 132, 133
		<code>\glsorder</code>
		116
		<code>\glspagelistwidth</code>
		372, 374, 376, 378
		<code>\glspar</code>
		142
		<code>\GLSpl</code>
		101, 112, 147

<code>\Glspl</code>	56, 101, 112, 147	<code>\Glstext</code>	317
<code>\glspl</code>	56, 101, 112, 147	<code>\glstext</code>	317
<code>\GLSplural</code>	318	<code>\glstextaccessdisplay</code>	155
<code>\Glsplural</code>	318	<code>\glstextformat</code>	64, 67
<code>\glsplural</code>	318	<code>\glstextup</code>	235
<code>\glspluralaccessdisplay</code>	155, 156	<code>\glstreechilddesc</code>	380, 382
<code>\glspluralsuffix</code>	136, 196, 200	<code>\glstreechildpredesc</code>	382
<code>\glspostdescription</code>	16, 17, 121, 189, 368, 369, 371–378, 381–383, 385	<code>\glstreechildprelocation</code> ...	380, 382, 384
<code>\glspostinline</code>	379	<code>\glstreechildsymbol</code>	380, 382
<code>\glspostlinkhook</code> .	62, 64, 77–81, 93, 201–211	<code>\glstreedefaultnamefmt</code>	379, 380
<code>\glsprestandardsort</code>	118	<code>\glstreedesc</code>	380–382
<code>\glsresetentrylist</code>	127, 140	<code>\glstreegroupheaderfmt</code>	
<code>\glssee</code>	48–50	381, 383–385, 396–402, 404
<code>\glsseeformat</code>	46, 52, 117, 124	<code>\glstreeindent</code>	382, 384, 394–396
<code>\glsseelist</code>	47	<code>\glstreeitem</code>	380, 398, 406
<code>\glssetabbrvfmt</code>		<code>\glstreenamebox</code>	395, 396
.....	61, 69, 90, 91, 173–175, 177, 178, 180, 186, 201–205, 208, 209, 211, 212	<code>\glstreenamefmt</code>	
<code>\glssetattribute</code>	218, 220, 222, 223, 225, 226, 228, 230, 231, 233– 238, 240, 242, 245, 247, 249–254, 256, 259–261, 263–270, 272, 274, 275, 277, 279–281, 283, 286, 287, 289, 291–293, 295–297, 299, 300, 302–304, 306–308, 310	379, 380, 382, 384, 386, 388–396
<code>\glssetcategoryattribute</code>		<code>\glstreenavigationfmt</code>	381, 383, 385, 397–402
.....	101, 113, 115, 148, 154, 169, 170, 172, 187	<code>\glstreenonamechilddesc</code>	384
<code>\glssetnoexpandfield</code>	12, 13	<code>\glstreenonamedesc</code>	383, 384
<code>\glssettoctitle</code>	120	<code>\glstreenonamesymbol</code>	384
<code>\glsshortaccessdisplay</code>	159, 160	<code>\glstreepredesc</code>	381, 383
<code>\glsshortpltok</code>		<code>\glstreeprelocation</code> ...	380, 382, 384, 385
.....	198, 218–224, 226, 228, 235– 238, 240, 245, 247, 249–254, 259, 261, 263–272, 281, 282, 285, 287–289, 291– 293, 295, 296, 300, 302–305, 307, 308, 310	<code>\glstreesubitem</code>	380, 406
<code>\glsshortpluralaccessdisplay</code>	160	<code>\glstreesubsubitem</code>	380, 407
<code>\glsshorttok</code>		<code>\glstreesymbol</code>	380, 382
.....	113, 114, 196–198, 218–224, 226, 228, 233, 235, 236, 238, 240, 242, 245, 247, 249–254, 256, 259, 261, 263–271, 273, 275, 281, 282, 285, 286, 288–293, 295, 296, 299, 300, 302–305, 307, 308, 310	<code>\GlsrLetField</code>	34
<code>\glssubentryitem</code>		<code>\glstype</code>	63, 66, 77–81, 148, 201–211
.....	138, 368–378, 380, 382, 384, 396, 406	<code>\glunset</code>	51, 63, 104, 105, 149
<code>\glssymbolaccessdisplay</code>	157	<code>\gluserdescription</code> ...	285, 287, 289, 292
<code>\glssymbolpluralaccessdisplay</code>	158	<code>\glswrite</code>	47, 116
<code>\glstarget</code>	138, 139, 368–378, 380, 382, 384, 395, 396, 406, 407	<code>\glswriteentry</code>	8, 9
<code>\GLStext</code>	317	<code>\Glsxtr</code>	56
		<code>\glxtr</code>	56
		<code>\glxtr@@do@wrglossary</code>	8, 10, 12, 13
		<code>\glxtr@addloclistfield</code>	14
		<code>\glxtr@addunused</code>	50, 51
		<code>\glxtr@applyabbrvfmt</code>	211
		<code>\glxtr@applyabbrvstyle</code> ...	194, 196, 215
		<code>\glxtr@counterrecord</code>	137
		<code>\glxtr@do@alsoindex@wrglossary</code> ...	14
		<code>\glxtr@doooption</code>	5, 16, 17, 23, 24, 26
		<code>\glxtr@fields</code>	135, 136
		<code>\glxtr@headentry@p</code>	94, 95
		<code>\glxtr@hyperoutsidefalse</code>	65
		<code>\glxtr@hyperoutsidettrue</code>	65
		<code>\glxtr@ifnextpunc</code>	193
		<code>\glxtr@ifpunctoken</code>	193

<code>\glxtr@inc@linkcount</code>	66, 153	<code>\glxtralttreeInit</code>	395, 401, 402
<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 23, 27	<code>\glxtrAltTreePar</code>	385
<code>\glxtr@indexonly@saveentrycounter</code>	13, 14, 27	<code>\glxtrAltTreeSetHangIndent</code> ...	385, 395
<code>\glxtr@keylist</code>	55, 56	<code>\glxtrAltTreeSetSubHangIndent</code>	396
<code>\glxtr@label</code>	408	<code>\glxtralttreeSubSymbolDescLocation</code>	396
<code>\glxtr@langtag</code>	136	<code>\glxtralttreeSymbolDescLocation</code> ..	385, 396
<code>\glxtr@linkprefix</code>	136	<code>\glxtrassignfieldfont</code>	69–76
<code>\glxtr@loaddialect</code>	331, 366	<code>\glxtrautoindex</code>	182
<code>\glxtr@makeglossaries</code>	116	<code>\glxtrautoindexassignsort</code>	183
<code>\glxtr@newabbreviation</code>	115, 196	<code>\glxtrautoindexentry</code>	183
<code>\glxtr@next</code>	193	<code>\glxtrautoindexesc</code>	183
<code>\glxtr@org@odo@wrglossary</code>	27	<code>\glxtrbibaddress</code>	333
<code>\glxtr@org@dohyperlink</code>	87	<code>\glxtrbibauthor</code>	333
<code>\glxtr@org@getgrouptitle</code>	125	<code>\glxtrbibbooktitle</code>	333
<code>\glxtr@org@newignoredglossary</code>	40	<code>\glxtrbibchapter</code>	333
<code>\glxtr@org@makenoidxglossaries</code>	52	<code>\glxtrbibedition</code>	333
<code>\glxtr@pluralsuffixes</code>	135, 136	<code>\glxtrbibhowpublished</code>	333
<code>\glxtr@process</code>	140, 141	<code>\glxtrbibinstitution</code>	333
<code>\glxtr@provideignoredglossary</code>	42	<code>\glxtrbibjournal</code>	333
<code>\glxtr@punclist</code>	192, 193	<code>\glxtrbibmonth</code>	333
<code>\glxtr@record</code>	11, 136	<code>\glxtrbibnote</code>	333
<code>\glxtr@record@nr</code>	13	<code>\glxtrbibnumber</code>	333
<code>\glxtr@recordsee</code>	12	<code>\glxtrbiborganization</code>	333
<code>\glxtr@resource</code>	134, 135	<code>\glxtrbibpages</code>	333
<code>\glxtr@s@newignoredglossary</code>	40	<code>\glxtrbibpublisher</code>	333
<code>\glxtr@s@provideignoredglossary</code> ...	42	<code>\glxtrbibschooll</code>	333
<code>\glxtr@saveentrycounter</code>	8, 9, 12, 84	<code>\glxtrbibseries</code>	333
<code>\glxtr@setaccessdisplay</code>	180	<code>\glxtrbibtitle</code>	333
<code>\glxtr@setbookindexmark</code>	408	<code>\glxtrbibtype</code>	333
<code>\glxtr@setup@record</code>	13, 14, 26, 27	<code>\glxtrbibvolume</code>	333
<code>\glxtr@shortcutsval</code>	136	<code>\glxtrbookindexatendgroup</code>	406
<code>\glxtr@texencoding</code>	136	<code>\glxtrbookindexatsubendgroup</code>	406
<code>\glxtr@undefaction@nr</code>	7	<code>\glxtrbookindexatsubsubendgroup</code> ..	406
<code>\glxtr@undefaction@val</code>	7	<code>\glxtrbookindexbetween</code>	406
<code>\glxtr@usesee</code>	45	<code>\glxtrbookindexbookmark</code>	407
<code>\glxtr@warnonexistsordo</code> .	7, 13, 14, 44, 45	<code>\glxtrbookindexcols</code>	405
<code>\glxtr@writefields</code>	134	<code>\glxtrbookindexcolspread</code>	405
<code>\glxtrabbreviationfont</code>	61	<code>\glxtrbookindexfirstmarkfmt</code>	408
<code>\glxtrabbrvfootnote</code>	223–225, 245–247, 259–261, 281–283	<code>\glxtrbookindexformatheader</code>	407
<code>\glxtrabbrvpluralsuffix</code>	136, 200, 218, 220, 223, 225, 227, 229, 231, 235, 249, 263, 285, 295, 297, 301, 306, 308	<code>\glxtrbookindexgroupskip</code>	407
<code>\glxtrabbrvtype</code>	17, 18, 198	<code>\glxtrbookindexlastmarkfmt</code>	408
<code>\glxtractivatenopost</code>	120	<code>\glxtrbookindexmulticolenv</code>	405
<code>\glxtraddallcrossrefs</code>	50	<code>\glxtrbookindexname</code>	403, 406
<code>\glxtralias</code>	84	<code>\glxtrbookindexparentchildsep</code>	403, 405
<code>\glxtrAltTreeIndent</code>	385, 386	<code>\glxtrbookindexparentsubchildsep</code> .	405, 406
		<code>\glxtrbookindexprelocation</code> ...	403, 406
		<code>\glxtrbookindexsubbetween</code>	406

\glstrbookindexsubname	407	\GlsXtrFormatLocationList	59, 61, 392–394
\glstrbookindexsubprelocation	407	\GLSxtrfull	18, 19, 322, 323
\glstrbookindexsubsubbetween	406	\Glsxtrfull	18, 19, 323
\glstrbookindexthepage	408	\glstrfull	18, 19, 322
\glstrcat	55, 56	\Glsxtrfullformat	
\glstrchecknohyperfirst	70–72	199, 213, 216, 219, 221, 223, 225, 227, 229, 233, 235, 237, 239, 241, 243, 245, 246, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 267, 269, 271, 272, 275, 277, 278, 280, 281, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\glstrcombingdiacriticIIrules .	337	\glstrfullformat	
\glstrcombingdiacriticIrules ..	337	199, 213, 216, 218, 221, 223, 225, 227, 229, 232, 235, 237, 239, 241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 272, 275, 276, 278, 280, 281, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\glstrcombingdiacriticIVrules ..	337	\GLSxtrfullpl	18, 19, 322, 323
\glstrComputeTreeIndent	395, 396	\Glsxtrfullpl	18, 19, 323
\glstrComputeTreeSubIndent	396	\glstrfullpl	18, 19, 322
\glstrcounterprefix	128	\Glsxtrfullplformat	
\glstrcurrencyrules	340	199, 213, 216, 219, 221, 223, 225, 227, 229, 233, 236, 237, 239, 241, 243, 245, 246, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 267, 269, 271, 272, 275, 277, 278, 280, 282, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\Glsxtrdefaultsubsequentfmt ...	214, 216	\GLSxtrfullpl	18, 19, 322, 323
\glstrdefaultsubsequentfmt ...	214, 216	\Glsxtrfullpl	18, 19, 323
\Glsxtrdefaultsubsequentplfmt .	214, 216	\glstrfullpl	18, 19, 322
\glstrdefaultsubsequentplfmt .	214, 216	\Glsxtrfullplformat	
\GlsXtrDefineAbbreviationShortcuts .	20	199, 213, 216, 219, 221, 223, 225, 227, 229, 233, 235, 237, 239, 241, 243, 245, 246, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 267, 269, 271, 272, 275, 277, 278, 280, 282, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\GlsXtrDefineAcShortcuts	20, 21	\glstrfullplformat	
\GlsXtrDefineOtherShortcuts	20, 21	213, 216, 219, 221, 223, 225, 227, 229, 233, 235, 237, 239, 241, 243, 245, 246, 248, 250, 251, 254, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 272, 275, 276, 278, 280, 281, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\glstrdetoklocation	147	\glstrfullplformat	
\glstrdiscardperiod	190	199, 213, 216, 219, 221, 223, 225, 227, 229, 233, 236, 237, 239, 241, 243, 245, 246, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 267, 269, 271, 272, 275, 277, 278, 280, 282, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\glstrdisplayendloc	127	\glstrfullplformat	
\glstrdisplayendloohook	128	213, 216, 219, 221, 223, 225, 227, 229, 233, 235, 237, 239, 241, 243, 245, 246, 248, 250, 251, 254, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 272, 275, 276, 278, 280, 281, 283, 286, 287, 290, 293, 296, 298, 301, 304, 306, 309
\glstrdisplaysingleloc	127, 128	\glstrfullsep	
\glstrdisplaystartloc	127	199, 218–222, 224, 226, 227, 229– 232, 235–244, 246–276, 278–280, 282, 284, 285, 294–296, 298, 300, 303–305, 309
\glstrdoautoindexname	85, 86, 179	\glstrgenabbrvfmt	61
\glstrdopostpunc	225, 247, 261, 283	\glstrgeneralpuncIIrules	340
\glstrdowrglossaryhook	85, 86	\glstrgeneralpuncIrules	340
\GlsXtrDualField	332	\glstrgetgrouptitle	126, 407
\glstremssuffix	263, 265, 267, 269, 270, 272, 274, 275, 277, 279, 281, 283	\glstrgroupfield	143
\GlsXtrEnableEntryCounting	113	\Glsxtrheadfirst	313
\GlsXtrEnableEntryUnitCounting	101	\glstrheadfirst	313
\GlsXtrEnableOnTheFly	54, 57	\Glsxtrheadfirstplural	313
\glstrendfor	32	\glstrheadfirstplural	313
\glstrfieldlistgadd	137	\Glsxtrheadfull	313
\glstrfieldtitlecase	174–177, 181		
\glstrfieldtitlecasecs	173		
\glstrfieldxifinlist	142		
\glstrfirstscfont	234		
\glstrfirstsmfont	249		
\GlsXtrFmtDefaultOptions	30		
\glstrfmtdisplay	30		
\GlsXtrFmtField	30, 31		
\glstrfootnotename			
....	222, 224, 245, 247, 259, 261, 281, 282		
\GlsXtrForeignTextField	36		

<code>\glxtrheadfull</code>	313	<code>\glxtrinlinefullformat</code>	200–202, 216,
<code>\Glsxtrheadfullpl</code>	313		224, 225, 227, 229, 230, 232, 239–241,
<code>\glxtrheadfullpl</code>	313		243, 244, 246, 248, 253–255, 257, 258,
<code>\Glsxtrheadlong</code>	313		260, 262, 270, 272–274, 276, 278, 279,
<code>\glxtrheadlong</code>	313		282, 284, 287, 290, 298, 301, 306, 309, 329
<code>\Glsxtrheadlongpl</code>	313	<code>\Glsxtrinlinefullplformat</code> ..	200, 203,
<code>\glxtrheadlongpl</code>	313		216, 224, 226, 227, 229, 230, 232, 239–
<code>\Glsxtrheadname</code>	313		241, 243, 244, 246, 248, 253, 255–257,
<code>\glxtrheadname</code>	138, 313		259, 261, 262, 271–274, 276, 278, 280,
<code>\Glsxtrheadplural</code>	313		282, 284, 288, 290, 298, 302, 307, 309, 330
<code>\glxtrheadplural</code>	313	<code>\glxtrinlinefullplformat</code>	
<code>\Glsxtrheadshort</code>	313	 199, 200, 203, 204, 216,
<code>\glxtrheadshort</code>	313		224, 226, 227, 229, 230, 232, 239–241,
<code>\Glsxtrheadshortpl</code>	313		243, 244, 246, 248, 253–255, 257, 258,
<code>\glxtrheadshortpl</code>	313		260, 262, 270, 272–274, 276, 278, 279,
<code>\Glsxtrheadtext</code>	313		282, 284, 288, 290, 298, 301, 307, 309, 329
<code>\glxtrheadtext</code>	313	<code>\glxtrininsertinsidefalse</code>	218
<code>\glxtrhyperlink</code>	23, 24, 88	<code>\GlsXtrInternalLocationHyperlink</code>	24, 129
<code>\glxtrhyphensuffix</code>	295, 303	<code>\glxtrLatinA</code>	342–347
<code>\glxtrifcounttrigger</code>	104, 105	<code>\glxtrLatinAELigature</code>	344, 346, 347
<code>\glxtrifcustomdiscardperiod</code>	190	<code>\glxtrLatinE</code>	342–347
<code>\glxtrifemptyglossary</code>	127, 133, 140	<code>\glxtrLatinEszettSs</code>	343–345, 347
<code>\GlsXtrIfFieldCmpNum</code>	33	<code>\glxtrLatinEszettSz</code>	343, 346
<code>\GlsXtrIfFieldEqNum</code>	138	<code>\glxtrLatinEth</code>	342–346
<code>\GlsXtrIfFieldNonZero</code>	331	<code>\glxtrLatinH</code>	342–347
<code>\glxtrifhasfield</code>	35, 36, 84, 332, 403	<code>\glxtrLatinI</code>	342–347
<code>\glxtrifhyphenstart</code>		<code>\glxtrLatinInsularG</code>	346
 294, 297, 299, 300, 303, 305	<code>\glxtrLatinK</code>	342–347
<code>\glxtrifindexing</code>	85	<code>\glxtrLatinL</code>	342–347
<code>\glxtrifinmark</code>	68, 95–97, 312–314	<code>\glxtrLatinM</code>	342–347
<code>\glxtrifnextpunc</code>	193, 194	<code>\glxtrLatinN</code>	342–347
<code>\glxtrifperiod</code>	190, 192	<code>\glxtrLatinO</code>	342–347
<code>\glxtrifrecordtrigger</code>	149–151	<code>\glxtrLatinOELigature</code>	344, 346, 347
<code>\glxtrifwasfirstuse</code> ..	69–72, 77–80, 83,	<code>\glxtrLatinP</code>	342–347
	115, 191, 201, 204–211, 224, 225, 247,	<code>\glxtrLatinS</code>	342–347
	261, 283, 287–289, 291, 300, 302, 305, 307	<code>\glxtrLatinT</code>	342–347
<code>\glxtrinclinkcounter</code>	153	<code>\glxtrLatinThorn</code>	346
<code>\glxtrindexaliased</code>	84, 85	<code>\glxtrLatinX</code>	342–347
<code>\glxtrindexseealso</code>	49, 50	<code>\glxtrlocationhyperlink</code>	128
<code>\glxtrinithyperoutside</code>	66	<code>\glxtrlocrangefmt</code>	127, 128
<code>\glxtrinitwrgloss</code>	66, 148	<code>\GlsXtrlong</code>	18, 19, 320, 321
<code>\glxtrinitwrglossbeforefalse</code>	64	<code>\GlsXtrlong</code>	18, 19, 321
<code>\glxtrinitwrglossbeforetrue</code>	64	<code>\glxtrlong</code>	18, 19, 320
<code>\Glsxtrinilinefullformat</code>	200, 202,	<code>\glxtrlonghyphen</code>	301
	216, 224, 226, 227, 229, 230, 232, 239–	<code>\glxtrlonghyphennoshort</code>	297, 298
	241, 243, 244, 246, 248, 253, 254, 256–	<code>\glxtrlonghyphenshort</code>	296
	258, 260, 262, 271–274, 276, 278, 280,	<code>\glxtrlongnosshortdescname</code> ..	231, 279, 297
	282, 284, 288, 290, 298, 301, 307, 309, 329	<code>\glxtrlongnosshortname</code>	
		 233, 242, 256, 273, 275, 299

<code>\GLSxtrlongpl</code>	18, 19, 320, 321	<code>\GlsXtrNoGlsWarningTail</code>	134
<code>\Glsxtrlongpl</code>	18, 19, 321, 322	<code>\glsxtrnopostpunc</code>	121
<code>\glsxtrlongpl</code>	18, 19, 321	<code>\glsxtronlydescname</code>	310
<code>\glsxtrlongshortdescname</code>		<code>\glsxtronlydescsort</code>	310
.....	219, 236, 250, 264, 266, 296, 302	<code>\glsxtronlyname</code>	308
<code>\glsxtrlongshortdescsort</code>		<code>\glsxtronlysuffix</code>	309
....	219, 236, 250, 264, 266, 291, 296, 302	<code>\glsxtrorg@ifKV@glslink@hyper</code>	62
<code>\glsxtrlongshortname</code>		<code>\glsxtrorglong</code>	196, 219, 297
218, 235, 249, 263, 265, 285, 286, 295, 300		<code>\glsxtrorgshort</code>	196, 219
<code>\glsxtrlongshortuserdescname</code> ..	288, 291	<code>\GLSxtrp</code>	94
<code>\glsxtrmarkhook</code>	311	<code>\Glsxtrp</code>	94
<code>\glsxtrMathItalicAlpha</code> .	351, 352, 355, 356	<code>\glsxtrp</code>	93, 95, 96
<code>\glsxtrMathItalicBeta</code> ..	351, 352, 355, 356	<code>\glsxtrparen</code>	191,
<code>\glsxtrMathItalicChi</code>	352, 356, 357	199, 218–222, 224, 226, 227, 229–232,	
<code>\glsxtrMathItalicDelta</code> .	351, 352, 355, 356	235–244, 246–276, 278–280, 282, 284,	
<code>\glsxtrMathItalicEpsilon</code>	351, 352, 355, 356	285, 294–296, 298, 300, 303–305, 309, 310	
<code>\glsxtrMathItalicEta</code> ...	351, 352, 355, 356	<code>\Glsxtrpl</code>	56
<code>\glsxtrMathItalicGamma</code> .	351, 352, 355, 356	<code>\glsxtrpl</code>	56
<code>\glsxtrMathItalicIota</code> ..	351, 352, 355, 356	<code>\glsxtrpostdescription</code> .	121, 172, 189, 379
<code>\glsxtrMathItalicKappa</code> .	351, 352, 355, 356	<code>\glsxtrposthyphenlong</code>	305, 307
<code>\glsxtrMathItalicLambda</code>	351, 352, 355, 356	<code>\glsxtrposthyphenshort</code>	300, 302
<code>\glsxtrMathItalicMu</code>	351, 352, 355, 356	<code>\glsxtrposthyphensequent</code>	
<code>\glsxtrMathItalicNu</code>	351, 352, 355, 356	300, 302, 306, 307
<code>\glsxtrMathItalicOmega</code>	352, 356, 357	<code>\glsxtrpostlink</code>	190
<code>\glsxtrMathItalicOmicron</code>	351, 352, 355, 357	<code>\glsxtrpostlinkendsentence</code>	190
<code>\glsxtrMathItalicPhi</code> ...	351, 352, 356, 357	<code>\glsxtrpostlinkhook</code>	190
<code>\glsxtrMathItalicPi</code>	351, 352, 356, 357	<code>\glsxtrpostlocalreset</code>	100, 102, 110
<code>\glsxtrMathItalicPsi</code>	352, 356, 357	<code>\glsxtrpostlocalunset</code>	100, 102, 110
<code>\glsxtrMathItalicRho</code> ...	351, 352, 356, 357	<code>\glsxtrpostlongdescription</code>	40
<code>\glsxtrMathItalicSigma</code> .	351, 352, 356, 357	<code>\glsxtrpostnamehook</code>	176–179, 181
<code>\glsxtrMathItalicTau</code> ...	351, 352, 356, 357	<code>\GlsXtrPostNewAbbreviation</code>	
<code>\glsxtrMathItalicTheta</code> .	351, 352, 355, 356	198, 215, 216,
<code>\glsxtrMathItalicUpsilon</code>	351, 352, 356, 357	218, 220, 222–224, 226, 228, 230, 231,	
<code>\glsxtrMathItalicXi</code>	351, 352, 355, 356	233–238, 240, 242, 245, 247, 249–254,	
<code>\glsxtrMathItalicZeta</code> ..	351, 352, 355, 356	256, 259, 261, 263–270, 272, 274, 275,	
<code>\glsxtrmultisupplocation</code>	333	277, 279–281, 283, 286–289, 291–293,	
<code>\glsxtrnewabbrevpresetkeyhook</code>	197	295–297, 299, 300, 302–305, 307, 308, 310	
<code>\glsxtrnewnumber</code>	19	<code>\glsxtrpostreset</code>	100, 102, 110
<code>\glsxtrnewsymbol</code>	19	<code>\glsxtrpostunset</code>	98, 102, 110
<code>\glsxtrNoGlossaryWarning</code>	14, 21, 129	<code>\glsxtrprelocation</code>	
<code>\GlsXtrNoGlsWarningAutoMake</code>	133	369, 371, 373, 375, 377, 380, 403
<code>\GlsXtrNoGlsWarningBuildInfo</code>	133	<code>\glsxtrprotectlinks</code>	87–89
<code>\GlsXtrNoGlsWarningCheckFile</code>	133	<code>\GlsXtrRecordCounter</code>	11
<code>\GlsXtrNoGlsWarningEmptyMain</code>	133	<code>\glsxtrrecordtriggervalue</code>	148
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	133	<code>\GlsXtrRecordWarning</code>	134
<code>\GlsXtrNoGlsWarningEmptyStart</code>	133	<code>\glsxtrregularfont</code>	61, 69
<code>\GlsXtrNoGlsWarningHead</code>	133	<code>\glsxtrresourcecount</code>	135
<code>\GlsXtrNoGlsWarningMisMatch</code>	133	<code>\glsxtrresourcefile</code>	135
<code>\GlsXtrNoGlsWarningNoOut</code>	134	<code>\glsxtrresourceinit</code>	134

<code>\glxtrrestoremarkhook</code>	311	<code>\glxtrtagfont</code>	188
<code>\glxtrrestorepostpunc</code>	121	<code>\Glsxtrtitlefirst</code>	313, 314, 327
<code>\glxtrscfont</code>	234	<code>\glxtrtitlefirst</code>	313, 314, 326
<code>\glxtrscsuffix</code>		<code>\Glsxtrtitlefirstplural</code>	313, 314, 327
..... 235, 237, 238, 240, 242, 244, 246, 248		<code>\glxtrtitlefirstplural</code>	313, 314, 327
<code>\GlsXtrSetActualChar</code>	185	<code>\Glsxtrtitlefull</code>	313, 314, 329
<code>\glxtrsetaliasnoindex</code>	13, 14, 84, 85	<code>\glxtrtitlefull</code>	313, 314, 329
<code>\GlsXtrSetEncapChar</code>	185	<code>\Glsxtrtitlefullpl</code>	313, 314, 330
<code>\GlsXtrSetEscChar</code>	185	<code>\glxtrtitlefullpl</code>	313, 314, 329
<code>\glxtrsetfieldifexists</code>	34, 35	<code>\Glsxtrtitlelong</code>	313, 314, 328
<code>\GlsXtrSetLevelChar</code>	185	<code>\glxtrtitlelong</code>	313, 314, 328
<code>\glxtrsetpopts</code>	93	<code>\Glsxtrtitlelongpl</code>	313, 314, 328
<code>\glxtrsetupfulldefs</code>		<code>\glxtrtitlelongpl</code>	313, 314, 328
..... 201–204, 225, 247, 261, 283		<code>\Glsxtrtitlename</code>	312–314, 325
<code>\GLSxtrshort</code>	18, 19, 97, 314, 315	<code>\glxtrtitlename</code>	312–314, 325
<code>\Glsxtrshort</code>	18, 19, 315	<code>\glxtrtitleorpdforheading</code>	
<code>\glxtrshort</code>	18, 19, 314 25, 138, 139, 312–314	
<code>\glxtrshortdescname</code> ...	228, 240, 254, 271	<code>\Glsxtrtitleplural</code>	313, 314, 326
<code>\glxtrshorthyphen</code>	306	<code>\glxtrtitleplural</code>	313, 314, 326
<code>\glxtrshorthyphenlong</code>	304	<code>\Glsxtrtitleshort</code>	312–314, 324
<code>\glxtrshortlongdescname</code>		<code>\glxtrtitleshort</code>	312–314, 323, 324
..... 221, 238, 252, 268, 269, 304, 307		<code>\Glsxtrtitleshortpl</code>	312–314, 324
<code>\glxtrshortlongdescsort</code>		<code>\glxtrtitleshortpl</code>	312–314, 324
..... 221, 238, 252, 268, 269, 293, 304, 307		<code>\Glsxtrtitletext</code>	313, 314, 326
<code>\glxtrshortlongname</code>		<code>\glxtrtitletext</code>	313, 314, 325
..... 220, 236, 251, 266, 268, 289, 292, 303, 305		<code>\GlsXtrTotalRecordCount</code>	147
<code>\glxtrshortlonguserdescname</code> ..	291, 293	<code>\glxtrtreetopindent</code>	386, 394
<code>\glxtrshortnolongname</code> .	226, 238, 253, 270	<code>\glxtrundefaction</code> ..	7, 13, 14, 28, 41, 43–45
<code>\GLSxtrshortpl</code>	18, 19, 315, 316	<code>\glxtrundeftag</code>	27, 123, 124
<code>\Glsxtrshortpl</code>	18, 19, 316	<code>\GlsXtrUnknownDialectWarning</code>	37
<code>\glxtrshortpl</code>	18, 19, 315	<code>\glxtrunsrtdo</code>	141, 142
<code>\glxtrsmfont</code>	249	<code>\glxtrUpAlpha</code>	350, 355, 356
<code>\glxtrsmsuffix</code>		<code>\glxtrUpBeta</code>	350, 355, 356
..... 249, 251, 253, 254, 256, 258, 260, 262		<code>\glxtrUpChi</code>	350, 351, 356, 357
<code>\GlsXtrStandaloneGlossaryType</code> .	138, 139	<code>\glxtrUpDelta</code>	350, 351, 355, 356
<code>\GlsXtrStandaloneSubEntryItem</code> .	138, 139	<code>\glxtrUpDigamma</code>	350, 351, 355
<code>\Glsxtrsubsequentfmt</code>		<code>\glxtrUpEpsilon</code>	350, 351, 355, 356
..... 212, 216, 232, 242, 244,		<code>\glxtrUpEta</code>	350, 351, 355, 356
257, 258, 274, 276, 278, 279, 298, 301, 306		<code>\glxtrUpGamma</code>	350, 351, 355, 356
<code>\glxtrsubsequentfmt</code>		<code>\glxtrUpIota</code>	350, 351, 355, 356
..... 212, 216, 232, 242, 244,		<code>\glxtrUpKappa</code>	350, 351, 355, 356
256, 258, 274, 276, 277, 279, 297, 301, 306		<code>\glxtrUpLambda</code>	350, 351, 355, 356
<code>\Glsxtrsubsequentplfmt</code>		<code>\glxtrUpMu</code>	350, 351, 355, 356
..... 212, 216, 232, 242, 244,		<code>\glxtrUpNu</code>	350, 351, 355, 356
257, 258, 274, 276, 278, 279, 298, 301, 306		<code>\glxtrUpOmega</code>	350, 351, 356, 357
<code>\glxtrsubsequentplfmt</code>		<code>\glxtrUpOmicron</code>	350, 351, 355, 357
..... 212, 216, 232, 242, 244,		<code>\glxtrUpPhi</code>	350, 351, 356, 357
256, 258, 274, 276, 277, 279, 298, 301, 306		<code>\glxtrUpPi</code>	350, 351, 356, 357
<code>\glxtrsupplocationurl</code>	129, 333	<code>\glxtrUpPsi</code>	350, 351, 356, 357

<code>\glstrUpRho</code>	350, 351, 356, 357	<code>\ifcsstring</code>	28, 169, 214
<code>\glstrUpSigma</code>	350, 351, 356, 357	<code>\ifcsundef</code>	
<code>\glstrUpTau</code>	350, 351, 356, 357	.. 33, 36–39, 41–43, 53, 57, 59, 89, 102,	
<code>\glstrUpTheta</code>	350, 351, 355, 356	107–111, 125, 126, 129, 142, 153, 154,	
<code>\glstrUpUpsilon</code>	350, 351, 356, 357	169, 214, 216, 217, 368, 386, 387, 395, 408	
<code>\glstrUpXi</code>	350, 351, 355, 357	<code>\ifcsvoid</code>	49, 168
<code>\glstrUpZeta</code>	350, 351, 355, 356	<code>\ifdef</code>	14, 19, 26, 30, 36, 38, 44,
<code>\GlsXtrUseAbbrStyleFmts</code>		47, 48, 51, 57, 58, 83, 87, 88, 94, 96, 97,	
.....	220, 222, 228, 230, 231, 233,	119, 123, 124, 136, 145, 171, 172, 185,	
	234, 236, 238, 241, 251, 252, 255, 265,	188, 284, 312, 323–329, 331, 365, 366,	
	266, 268, 270, 273, 277, 281, 289, 291,	368–371, 379–385, 396–402, 404, 407, 408	
	292, 294, 296, 297, 299, 302, 304, 308, 310	<code>\ifdefempty</code>	7–9, 33, 36, 37, 41–43, 45,
<code>\GlsXtrUseAbbrStyleSetup</code>	228, 230, 231,	46, 66, 68, 101, 113, 116, 119, 128, 140,	
	233, 234, 241, 243, 255, 258, 273, 277, 280	143, 147, 148, 162–164, 187, 195, 212, 405	
<code>\glstruserfield</code>	284, 285	<code>\ifdefequal</code>	35, 52, 133, 143, 180
<code>\glstruserparen</code>	285–293	<code>\ifdefstring</code>	6, 23, 24, 35, 182, 187, 404
<code>\glstrusersuffix</code>	286, 287, 289, 292	<code>\ifdefvoid</code> ..	45, 48–50, 88, 107, 125, 129, 144
<code>\glstruseeseealsoformat</code>	46, 47	<code>\ifdim</code>	58, 115, 386–394
<code>\glstruseeseeformat</code>	46	<code>\IfFileExists</code>	22, 129, 133, 134, 136, 365, 367
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	194, 217	<code>\ifglossaryexists</code>	45
<code>\GlsXtrWarning</code>	55, 56	<code>\ifglshasacronym</code>	18, 133
<code>\glstrword</code>	195	<code>\ifglshasrshortcuts</code>	20
<code>\glstrwordsep</code>	195, 294, 297, 299, 300, 303, 305	<code>\ifglshasautomake</code>	119, 133, 136
<code>\glstrwrglossmark</code>	24	<code>\ifglshasentrycounter</code>	38
		<code>\ifglshasentryexists</code>	9,
			27, 43, 44, 55, 56, 59, 69, 144, 169, 189, 190
		<code>\ifglshasfield</code>	168
		<code>\ifglshasfield</code>	30, 31, 284, 285
		<code>\ifglshaslong</code>	106, 107, 151, 152
		<code>\ifglshasparent</code>	138, 139, 141, 144, 388–390
		<code>\ifglshasshort</code>	46, 61, 69
		<code>\ifglshassymbol</code>	191, 382, 383, 385
		<code>\ifglshasindexonlyfirst</code>	85
		<code>\ifglshasgroupskip</code>	
		369, 371–379, 381, 382, 384, 396, 404
		<code>\ifglshasnonumberlist</code>	61
		<code>\ifglshasnopostdot</code>	16, 121
		<code>\ifglshasnormalizesort</code>	119
		<code>\ifglshassubentrycounter</code>	38
		<code>\ifglshasused</code>	50, 51, 83, 85,
			103, 112, 115, 212, 388, 389, 391, 392, 394
		<code>\ifglshasxindy</code>	129, 131
		<code>\ifglshasxtr@hyperoutside</code>	67
		<code>\ifglshasxtrininitwrglossbefore</code>	64, 67, 148, 149
		<code>\ifglshasxtrininsertinside</code>	
		204–211, 214, 218, 219, 221,
			223–227, 229–233, 235–237, 239–248,
			250–262, 264–267, 269–284, 286–288,
			290, 293, 294, 297, 300–303, 305, 307, 309

H

<code>\hangindent</code>	382, 384, 385, 395–397, 401, 402
<code>\hbox</code>	368
<code>\hfill</code>	368
<code>\href</code>	88
<code>\hsize</code>	58
<code>\hss</code>	368
<code>\hyperlink</code>	88
<code>\hyperpage</code>	182
<code>\hyperref</code>	87, 129, 331, 332
<code>hyperref package</code>	89, 181, 310, 323

I

<code>\if</code>	54
<code>\if@glstr@autoseeindex</code>	26, 45, 48
<code>\if@glstr@format@override</code>	182
<code>\if@glstr@docdefrestricted</code>	52
<code>\if@glstr@indexcrossrefs</code>	15, 50
<code>\ifblank</code>	29, 55, 56, 116
<code>\ifbool</code>	28, 39
<code>\ifcase</code> ..	7, 13, 20, 21, 24, 53, 64, 122, 380, 406
<code>\ifcsdef</code>	28,
	38–43, 65, 67, 81, 82, 93–97, 107, 120,
	125, 126, 138, 143, 145, 147, 152, 174,
	175, 177–180, 190, 194, 211, 215, 371–378

<code>\ifHy@hyperindex</code>	181	153, 163, 174–184, 187–189, 193–197,
<code>\ifinlist</code>	99	201–211, 214, 216, 225, 247, 261, 283,
<code>\ifinlistcs</code>	32, 53	311–314, 365, 380, 385, 387, 398, 405–408
<code>\ifinner</code>	25	<code>\letabbreviationstyle</code> .. 224, 226, 228,
<code>\ifKV@glslink@hyper</code>	62, 66–68	230, 233, 234, 240, 241, 254, 255, 271, 273
<code>\ifKV@glslink@local</code>	63, 149	<code>\letcs</code>
<code>\ifKV@glslink@noindex</code> ..	8, 9, 12, 30, 84, 85	28, 32, 33, 45, 46, 50,
<code>\ifmmode</code>	25	65, 67, 81, 123–125, 143, 144, 174–181, 408
<code>\ifnum</code>	15, 33, 51, 52, 103, 104,	<code>\levelchar</code>
	111, 112, 125, 135, 148, 382, 384, 395, 396	185
<code>\ifstrempy</code>	138, 145, 154	<code>\listadd</code>
<code>\ifstrequal</code>	17, 22	107
<code>\ifthenelse</code>	133, 190	<code>\listbreak</code>
<code>\IfTrackedDialectHasMapping</code>	36	187
<code>\IfTrackedLanguageFileExists</code>	330	<code>\listcsadd</code>
<code>\ifundef</code>	23, 32, 33, 36, 116, 187–189, 365	31
<code>\ifx</code>	8–11,	<code>\listcseadd</code>
	47, 57, 59, 67, 116, 120, 122, 127, 128,	31, 108
	137, 182–184, 186, 193, 195, 197, 294, 402	<code>\listcsgadd</code>
<code>\immediate</code> ...	52, 103, 111, 112, 129, 130, 136	31, 53
<code>\index</code>	182	<code>\listcsxadd</code>
<code>\indexspace</code> .	369, 381–385, 396–402, 404, 407	31, 108
<code>\input</code>	330	<code>\listxadd</code>
<code>\inputencodingname</code>	136	99
<code>\InputIfFileExists</code>	51	<code>\loadglsentries</code>
<code>\istfilename</code>	116	53, 131
<code>\item</code>	131, 132, 368–371, 380, 381, 397, 398	<code>\long</code>
		40
J		
<code>\jobname</code>	51, 52, 129, 131–136	
K		
<code>\key@ifundefined</code> .	12, 13, 28, 29, 81, 140, 143	
<code>\KV@glslink@hyperfalse</code>	70, 83, 89	
<code>\KV@glslink@hypertrue</code>	89	
<code>\KV@glslink@noindexfalse</code>	83, 84	
<code>\KV@glslink@noindextrue</code>	84, 89	
L		
<code>\L</code>	347, 350	
<code>\l</code>	350	
<code>\label</code>	23	
<code>\LaTeX</code>	131, 132	
<code>\leaders</code>	368	
<code>\leavevmode</code>	40, 66	
<code>\let</code>	5, 7–14, 16, 18, 19,	
	25–27, 30, 32, 35–37, 39, 40, 52–54, 57–	
	60, 62–64, 66–80, 82–87, 89, 90, 93, 98,	
	99, 101–103, 110, 111, 113–117, 119–	
	126, 134, 136, 137, 140, 141, 143, 148,	
		153, 163, 174–184, 187–189, 193–197,
		201–211, 214, 216, 225, 247, 261, 283,
		311–314, 365, 380, 385, 387, 398, 405–408
		<code>\letabbreviationstyle</code> .. 224, 226, 228,
		230, 233, 234, 240, 241, 254, 255, 271, 273
		<code>\letcs</code>
		28, 32, 33, 45, 46, 50,
		65, 67, 81, 123–125, 143, 144, 174–181, 408
		<code>\levelchar</code>
		185
		<code>\listadd</code>
		107
		<code>\listbreak</code>
		187
		<code>\listcsadd</code>
		31
		<code>\listcseadd</code>
		31, 108
		<code>\listcsgadd</code>
		31, 53
		<code>\listcsxadd</code>
		31, 108
		<code>\listxadd</code>
		99
		<code>\loadglsentries</code>
		53, 131
		<code>\long</code>
		40
M		
<code>\MakeAcronymsAbbreviations</code>	115	
<code>\makeatletter</code>	51, 129, 134, 184	
<code>\makeatother</code>	184	
<code>\makebox</code>	368, 395, 396	
<code>\makefirstuc</code>	188	
<code>makeglossaries</code>	122	
<code>\makeglossaries</code>	116, 130, 132, 133, 137	
<code>\makeglossary</code>	117	
<code>makeindex</code>	409	
<code>makeindex</code>	14, 116	
<code>\makenoidxglossaries</code>	132	
<code>\MakeTextUppercase</code>	313	
<code>\MakeUppercase</code>	312–314	
<code>\marginpar</code>	25	
<code>\markboth</code>	312	
<code>\markright</code>	312	
<code>\mathit</code>	335	
<code>\mathrm</code>	334–336	
<code>\maxdimen</code>	58	
<code>\mbox</code>	370, 371, 395	
<code>\medskip</code>	133, 142	
<code>\MessageBreak</code>		
	.. 53, 57, 103, 112, 116, 119, 120, 134, 215	
<code>mfistuc package</code>	187, 188	
<code>\mfistucMakeUppercase</code>		
	.. 70–80, 82, 91–94, 97,	
	106, 114, 152, 154–161, 164–167, 176,	
	177, 181, 202, 204, 206, 207, 209, 211–213	
<code>\mfu@checkword@arg</code>	187, 188	
<code>\mfu@checkword@do</code>	188	

N

<code>\NeedsTeXFormat</code>	5, 331, 367, 403	<code>short</code>	160, 167, 195
<code>\new@atom@glossaryentry</code>	52	<code>shortaccess</code>	162–164
<code>\new@glossaryentry</code>	53, 119	<code>shortaccessplural</code>	163
<code>\new@ifnextchar</code>	30, 81, 82, 106, 145, 149–151, 192, 201–211	<code>shortplural</code>	160, 167, 195
<code>\newabbr</code>	18, 19	<code>symbol</code>	157, 165, 166
<code>\newabbreviation</code>	18, 19	<code>symbolplural</code>	157, 158, 166
<code>\newabbreviationhook</code>	198	<code>text</code>	87, 155, 165, 217, 219, 317, 325
<code>\newabbreviationstyle</code> 218–222, 224, 226, 228, 230, 231, 233–236, 238, 240–243, 245, 247, 249–252, 254–256, 258, 259, 261, 263–266, 268–271, 273, 275, 277, 279–282, 285, 286, 288, 289, 291–293, 295–297, 299, 300, 302–305, 307, 308, 310		<code>textaccess</code>	163
<code>\newacronym</code>	113, 114	<code>user2</code>	37
<code>\newacronymhook</code>	113	<code>\newglossarystyle</code>	405
<code>\newacronymstyle</code>	114, 115	<code>\newif</code>	64, 181, 218
<code>\newcommand</code> ... 5–13, 15–34, 36–46, 49–51, 54–57, 59–62, 64, 65, 68–70, 81, 82, 84–86, 88, 89, 93–103, 106–116, 120–135, 137–145, 147–162, 164–173, 179–196, 198–212, 214–222, 226, 228, 231, 233–235, 249, 263, 284, 285, 288, 290, 294, 295, 297, 299, 300, 303, 305, 308, 310, 312, 314–333, 336–365, 367, 369, 379–383, 385–387, 394, 395, 403, 404, 407, 408		<code>\newlength</code>	385, 386
<code>\newcount</code>	135, 145	<code>\newnum</code>	19
<code>\newcounter</code>	23, 152	<code>\newrobustcmd</code> 29, 31, 32, 34, 35, 47, 48, 52, 65, 69, 81, 82, 93, 94, 106, 121, 125, 138, 139, 142, 145, 146, 149–151, 180, 187, 188, 201–212, 294, 312, 314–323, 387–394	
<code>\newentry</code>	19	<code>\newsym</code>	19
<code>\newglossary</code>	17, 117	<code>\newterm</code>	171
<code>\newglossaryentry</code>	19, 52, 53, 102, 109, 114, 171, 172, 198	<code>\newtoks</code>	194
<code>\newglossaryentry options</code>		<code>\newwrite</code>	52, 116
<code>access</code>	163	<code>\nobreak</code> 370, 371, 381, 383–385, 397–400	
<code>alias</code>	16, 45, 48–50	<code>\NoCaseChange</code>	95–97, 139, 314–323
<code>desc</code>	158, 166	<code>\noexpand</code> 10, 11, 22, 47, 49–51, 114, 129, 130, 134, 141–143, 153, 183, 184, 198, 367, 406	
<code>descplural</code>	159, 166	<code>\nofiles</code>	133
<code>first</code> 87, 156, 165, 217, 318–320, 326, 409		<code>\noindent</code>	133, 383–385, 398–400, 402
<code>firstaccess</code>	164	<code>\nopagebreak</code> 381, 383–385, 396–403, 407	
<code>firstplural</code> .. 156, 157, 165, 217, 319, 327, 409		<code>\nopostdesc</code>	40, 55, 56, 121, 172
<code>group</code>	143	<code>\ns@GLSxtrfull</code>	202
<code>loclist</code>	31	<code>\ns@Glsxtrfull</code>	201
<code>long</code>	161, 167, 327	<code>\ns@glxtrfull</code>	201
<code>longplural</code>	161, 167, 328	<code>\ns@GLSxtrfullpl</code>	203
<code>name</code>	46, 154, 164, 182, 316, 317, 325	<code>\ns@Glsxtrfullpl</code>	203
<code>plural</code> 155, 156, 165, 217, 317, 318, 326		<code>\ns@glxtrfullpl</code>	202
<code>see</code>	15, 16, 26, 45, 48, 50, 53, 117	<code>\ns@GLSxtrlong</code>	207
<code>seealso</code>	16, 45, 46, 48–50, 420	<code>\ns@Glsxtrlong</code>	206
		<code>\ns@glxtrlong</code>	206
		<code>\ns@GLSxtrlongpl</code>	211
		<code>\ns@Glsxtrlongpl</code>	210
		<code>\ns@glxtrlongpl</code>	210
		<code>\ns@GLSxtrshort</code>	205
		<code>\ns@Glsxtrshort</code>	205
		<code>\ns@glxtrshort</code>	204
		<code>\ns@GLSxtrshortpl</code>	209
		<code>\ns@Glsxtrshortpl</code>	208
		<code>\ns@glxtrshortpl</code>	208
		<code>\null</code>	21
		<code>\number</code>	51, 108–111, 134, 145

<code>\numexpr</code>	108, 109, 111	<code>stylemods</code>	22
O		<code>symbols</code>	19, 172
<code>\O</code>	347, 350	<code>undefaction</code>	43, 44
<code>\o</code>	350	error	6
<code>\openout</code>	52	warn	6
<code>\or</code>	7, 13, 14, 20, 21, 24, 53, 64, 380, 406	xindy	47, 48
<code>\org@glossaryentrynumbers</code>	59, 120	<code>\PackageError</code>	6, 11, 22, 26, 52, 53, 57, 65, 81, 82, 85, 93, 94, 101–103, 109, 111, 113, 115–119, 126, 137, 141, 142, 145, 190, 214–217, 367, 368
<code>\org@glossarytitle</code>	120	<code>\PackageWarning</code>	16
<code>\org@ifKV@glslink@hyper</code>	66, 68	<code>\PackageWarningNoLine</code>	16
P		<code>\pageref</code>	23, 38
<code>\p@gl@hyp@opt</code>	86	<code>\par</code>	132– 134, 370, 371, 380, 382–385, 395–402, 404
package options:		<code>\parindent</code> ..	380, 382, 384–386, 395–402, 405
abbreviations	17, 18	<code>\parskip</code>	380, 382, 384, 398–400, 405
accsupp	21, 154	<code>\PassOptionsToPackage</code>	5, 22
acronym	17	<code>\pdfbookmark</code>	404
automake	119, 131, 136	<code>\preglossarypreamble</code>	39
true	136	<code>\preto</code>	84
autoseeindex	26	<code>\print@noop@unsrtglossaryunit</code> ...	12, 13
false	26	<code>\print@op@unsrtglossaryunit</code>	14
counter		<code>\printabbreviations</code>	17
wrglossary	23	<code>\printglossaries</code>	117, 132
debug		<code>\printglossary</code>	17, 117, 132, 134
showtargets	88	<code>\printglossary options</code>	
docdef	14, 15, 52, 53, 102, 109	nonumberlist	61
atom	51	type	119
false	52, 53	<code>\printnoidxglossaries</code>	132
restricted	15	<code>\printnoidxglossary</code>	118, 132
true	51, 53	<code>\printnumbers</code>	19, 172
docdefs		<code>\printsymbols</code>	19, 172
restricted	52	<code>\printunsrtglossary</code>	132, 134, 140
indexcounter	331	<code>\printunsrtglossaryentryprocesshook</code>	
nonumberlist	59	140, 141
nopostdot	16	<code>\printunsrtglossaryhandler</code>	141, 142
false	16	<code>\printunsrtglossarypredoglossary</code> ..	141
numbers	19	<code>\printunsrtglossaryskipentry</code>	140
postdot	16	<code>\printunsrtglossaryunit</code>	13, 14, 142
record ..	7, 13, 52, 62, 116, 134, 201–211, 418	<code>\printunsrtglossaryunitsetup</code>	142
alsoindex	8, 10	<code>\ProcessOptions</code>	368
only	8, 132	<code>\ProcessOptionsX</code>	24
shortcuts	20	<code>\protect</code>	95–97, 164– 167, 195, 199, 218–228, 230–283, 285– 293, 295–300, 302–305, 307–310, 314–323
ac	20	<code>\protected@csedef</code>	34, 35, 126, 386, 387
all	20	<code>\protected@csxdef</code>	35, 126, 386, 387
false	20		
none	20		
true	20		
sort			
use	68		
style	22		

<code>\protected@edef</code>	35, 57, 65, 87, 114, 125, 126, 142, 143, 182, 183, 198
<code>\protected@write</code>	11, 12, 52, 60, 61, 116, 117, 134–137, 408
<code>\providecommand</code>	17–19, 29, 47, 61, 82, 84, 103, 112, 116, 130, 135, 136, 331, 334–336, 368, 403
<code>\ProvidesFile</code>	330
<code>\ProvidesPackage</code>	5, 331, 367, 403
Q	
<code>\quotechar</code>	185
R	
<code>\raggedright</code>	373, 374, 377, 378, 405
<code>\refstepcounter</code>	23
<code>\relax</code>	7, 13–15, 18–21, 24, 26, 30, 33, 51–54, 57, 58, 60, 64, 66, 67, 86, 93, 103, 104, 111, 112, 117, 120, 121, 124, 125, 127, 134–137, 144, 148, 183, 184, 186, 188, 190, 191, 195–197, 294, 380, 382, 384, 387–397, 401, 402, 405–408
<code>relsize package</code>	249
<code>\renewcommand</code>	6, 7, 13–18, 20–24, 26, 39–44, 46, 51–54, 56, 57, 59–64, 81, 83– 85, 87, 89, 93, 100–103, 106, 107, 109– 122, 125–130, 142, 147, 171, 173–176, 178, 186–190, 200, 215, 216, 218–293, 295–311, 365, 368–385, 395–402, 405–407
<code>\renewenvironment</code>	369, 371– 378, 380, 382, 384, 395, 398–400, 402, 405
<code>\renewglossarystyle</code>	368–378, 380–384, 395–402
<code>\renewrobustcmd</code>	68, 88
<code>\RequireGlossariesExtraLang</code> ...	330, 365
<code>\RequirePackage</code> ...	5, 14, 21, 22, 25, 367, 403
<code>\reserved@a</code>	193
<code>\reserved@b</code>	193
<code>\reserved@d</code>	193
<code>\RestoreAcronyms</code>	115
<code>\rGLS</code>	147
<code>\rGls</code>	147
<code>\rgls</code>	147
<code>\rGLSformat</code>	151
<code>\rGlsformat</code>	150
<code>\rglsformat</code>	149, 152
<code>\rGLSpl</code>	147
<code>\rGlspl</code>	147
<code>\rglspl</code>	147
<code>\rGLSplformat</code>	151
<code>\rGlsplformat</code>	150
<code>\romannumeral</code>	386, 387, 395
S	
<code>\s@glstrfmt</code>	30
<code>\s@glshyp@opt</code>	86
<code>\s@glstr@enabletagging</code>	187
<code>\s@glstrfmt</code>	29
<code>\s@glstrifhasfield</code>	32
<code>\s@GlsXtrStartUnsetBuffering</code>	98
<code>\s@GlsXtrStopUnsetBuffering</code>	99
<code>\s@printunsrtglossary</code>	139, 142
<code>\seealsoname</code>	46, 48
<code>\seename</code>	46
<code>\setabbreviationstyle</code>	115, 219, 228
<code>\setacronymstyle</code>	114, 115
<code>\setentrycounter</code>	127, 333
<code>\SetGenericNewAcronym</code>	115
<code>\setglossarystyle</code>	23, 120, 368, 370, 371, 381, 383, 384, 396–402, 405
<code>\setkeys</code>	9, 22, 26, 30, 66, 68, 85, 113, 120, 148, 196, 197
<code>\setlength</code>	58, 380, 382, 384, 385, 396, 398–400, 405
<code>\settowidth</code>	115, 386–395
<code>\setupglossaries</code>	5, 26
<code>\sfcode</code>	16, 17, 190, 191, 379
<code>\small</code>	25
<code>soul package</code>	99
<code>\space</code> 6, 11, 47, 53, 54, 57, 85, 101–103, 109, 111–113, 115–118, 120, 130, 133, 134, 137, 141, 142, 145, 191, 195, 199, 219, 368, 369, 379, 382, 383, 385, 386, 395, 403	
<code>\spacefactor</code>	16, 17, 190, 191, 197, 379
<code>\stepcounter</code>	153
<code>\string</code> . 6, 11, 12, 47, 52–54, 57, 60, 61, 67, 81, 82, 85, 93, 94, 101–103, 109, 111– 113, 115–120, 130–137, 141, 142, 145, 175, 177–180, 183, 190, 331, 336–365, 408	
<code>\strut</code>	368–378, 384
<code>\subglossentry</code>	120, 144, 368–378, 380, 382, 384, 396, 406
<code>\subitem</code>	380
<code>\subsubitem</code>	380
T	
<code>\tablehead</code>	375–378
<code>\tabletail</code>	375–378
<code>\tabularnewline</code>	371–379

