

`glossaries-extra.sty v1.31: documented code`

Nicola L.C. Talbot

Dickimaw Books

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

2018-05-09

Abstract

This is the documented code for the glossaries-extra package. See [glossaries-extra-manual.pdf](#) for the user manual.

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

Contents

1	Main Package Code (glossaries-extra.sty)	5
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	27
1.3	Modifications to Commands Provided by glossaries	36
1.3.1	Existence Checks	41
1.3.2	Document Definitions	48
1.3.3	Existing Glossary Style Modifications	54
1.3.4	Entry Formatting, Hyperlinks and Indexing	58
1.3.5	Entry Counting	94
1.3.6	Acronym Modifications	109
1.3.7	Indexing and Displaying Glossaries	112
1.4	Link Counting	148
1.5	Integration with glossaries-accsupp	150
1.6	Categories	164
1.7	Abbreviations	190
1.7.1	Abbreviation Styles Setup	210
1.7.2	Predefined Styles (Default Font)	214
1.7.3	Predefined Styles (Small Capitals)	230
1.7.4	Predefined Styles (Fake Small Capitals)	245
1.7.5	Predefined Styles (Emphasized)	259
1.7.6	Predefined Styles (User Parentheses Hook)	280
1.7.7	Predefined Styles (Hyphen)	290
1.7.8	Predefined Styles (No Short on First Use)	304
1.8	Using Entries in Headings	306
1.9	Multi-Lingual Support	326
1.10	glossaries-extra-bib2gls.sty	327
2	Style Adjustments (glossaries-extra-stylemods.sty)	363
2.1	Package Initialisation	363
2.2	List-Like Styles	364
2.3	Longtable Styles	367
2.4	Long Ragged Styles	369
2.5	Supertabular Styles	371
2.6	Super Ragged Styles	373
2.7	Inline Style	375
2.8	Tree Styles	375
2.9	Multicolumn Styles	393

3 bookindex style (glossary-bookindex.sty)	399
3.1 Package Initialisation and Options	399
Glossary	405
Change History	406
Index	425

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/05/09 v1.31 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

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

```
6 {%
```

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

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

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

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

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

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

```
13 }%
```

Set the defaults.

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

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

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

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

```
18 {}%
```

```
19 {%
```

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

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

```
22 {}%
```

```
23 }%
```

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

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

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

```
27 }
```

```
28 }
```

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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@forglsentries\@glstr@do@redef@forglsentries
71    \or
72    \let\glstrundefaction\@glstr@err@undefaction
73    \let\glstr@warnonexistssordo\@gobble
74    \let\@glstr@redef@forglsentries\relax
75    \fi
76  }

```

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

`\@glstr@record` Does nothing by default.

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

`\glstr@recordsee` Does nothing by default.

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

`\ultnumberformat`

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

```

`\ultNumberFormat`

```

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          Increment associated counter.
110          \glsxtr@inc@wrglossaryctr{#1}%
111          \@@do@@wrglossary
112        \fi
113      }%
114    \fi
115 }

```

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

```

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

```

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

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

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

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

Save the label in case it's needed.

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

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

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

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

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

```

180 \protected@write\@auxout{}\string\glsxtr@record
181   {\@gls@label}\@glo@counterprefix{\@gls@counter}\@glsnumberformat}%
182   {\@glsrecordlocref}}%
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 `\@glslocref` 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*{\@glxtr@recordsee}[2]{%
213   \@@glxtrwrglossmark
214   \def\@gls@xref{#2}%
215   \@onelevel@sanitize\@gls@xref
216   \protected@write\@auxout{}\string\glxtr@recordsee{#1}{\@gls@xref}}%
217 }

```

srtglossaryunit

```

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

```

tr@setup@record Initialise.

```

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

```

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

```

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

```

addloclistfield

```

228 \newcommand*{\glxtr@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}

```

alsoindex

```

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

```

only

```

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

```

off

```

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

```

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@see@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}}
```

lsxtrdocdeffalse

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

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}[true]%
```

```
324 {%
```

```
325   \ifnum\@glxtr@docdefval=2\relax
```

```
326     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
```

```
327   \fi
```

```
328 }
```

ocdefrestricted

```
329 \newcommand*{\if@glxtrdocdefrestricted}{\ifnum\@glxtr@docdefval=2 }
```

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).

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

indexcrossrefs

Automatically index cross references at the end of the document

```
331 \define@boolkey{glossaries-extra.sty}[\@glxtr]{indexcrossrefs}[true]{%
```

```
332   \if@glxtrindexcrossrefs
```

```
333   \else
```

```
334     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
```

```
335   \fi
```

```
336 }
```

Switch off since this can increase the build time.

```
337 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

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

```

339 \define@boolkey{glossaries-extra.sty}[@glxtr@]{autoseeindex}[true]{%
340 }
341 \@glxtr@autoseeindextrue

```

iesExtraWarning Allow users to suppress warnings.

```

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

```

raWarningNoLine Allow users to suppress warnings.

```

343 \newcommand*{\GlossariesExtraWarningNoLine}[1]{%
344   \PackageWarningNoLine{glossaries-extra}{#1}}

345 \@glxtr@declareoption{nowarn}{%
346   \let\GlossariesExtraWarning\@gobble
347   \let\GlossariesExtraWarningNoLine\@gobble
348   \glxtr@doption{nowarn}%
349 }

```

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

```

350 \newcommand*{\@glxtr@defpostpunc}{}

```

postdot Shortcut for `nopostdot=false`

```

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

```

nopostdot Needs to redefine `\@glxtr@defpostpunc`

```

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

```

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

```

365 \define@key{glossaries-extra.sty}{postpunc}{%
366   \glxtr@doption{nopostdot=false}%
367   \ifstrequal{#1}{dot}%
368   {%
369     \renewcommand*{\@glxtr@defpostpunc}{%

```



```

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

```

`\glsxtrabbrvtype` Glossary type for abbreviations.

```

396 \newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}

```

`\bbreviationsdef` Set by abbreviations option.

```

397 \newcommand*{\@glsxtr@abbreviationsdef}{}

```

`\bbreviationsdef`

```

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

```

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

```

408   \ifglsacronym
409   \else
410     \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%

```

```

411 \fi
412 }%

```

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

```

413 \@glsxtr@declareoption{abbreviations}{%
414 \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
415 }

```

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`).

```

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

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

```

Disable this command after it's been used.

```

442 \let\GlsXtrDefineAbbreviationShortcuts\relax
443 }

```

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

```

444 \newcommand*{\GlsXtrDefineAcShortcuts}{%
445 \newcommand*{\ac}{\cglsl}%
446 \newcommand*{\acp}{\cglspl}%
447 \newcommand*{\acs}{\glsxtrshort}%

```

```

448 \newcommand*{\acsp}{\glxtrshortpl}%
449 \newcommand*{\acl}{\glxtrlong}%
450 \newcommand*{\aclp}{\glxtrlongpl}%
451 \newcommand*{\acf}{\glxtrfull}%
452 \newcommand*{\acfp}{\glxtrfullpl}%
453 \newcommand*{\Ac}{\cGls}%
454 \newcommand*{\Acp}{\cGlspl}%
455 \newcommand*{\Acs}{\Glsxtrshort}%
456 \newcommand*{\Acsp}{\Glsxtrshortpl}%
457 \newcommand*{\Acl}{\Glsxtrlong}%
458 \newcommand*{\Aclp}{\Glsxtrlongpl}%
459 \newcommand*{\Acf}{\Glsxtrfull}%
460 \newcommand*{\Acfp}{\Glsxtrfullpl}%
461 \newcommand*{\AC}{\cGLS}%
462 \newcommand*{\ACP}{\cGLSpl}%
463 \newcommand*{\ACS}{\GLSxtrshort}%
464 \newcommand*{\ACSP}{\GLSxtrshortpl}%
465 \newcommand*{\ACL}{\GLSxtrlong}%
466 \newcommand*{\ACLP}{\GLSxtrlongpl}%
467 \newcommand*{\ACF}{\GLSxtrfull}%
468 \newcommand*{\ACFP}{\GLSxtrfullpl}%

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

```

Disable this command after it's been used.

```

470 \let\GlsXtrDefineAcShortcuts\relax
471 }

```

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

```

472 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
473 \newcommand*{\newentry}{\newglossaryentry}%
474 \ifdef\printsymbols
475 {%
476 \newcommand*{\newsym}{\glxtrnewsymbol}%
477 }{}%
478 \ifdef\printnumbers
479 {%
480 \newcommand*{\newnum}{\glxtrnewnumber}%
481 }{}%
482 \let\GlsXtrDefineOtherShortcuts\relax
483 }

```

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.

```

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

```

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

```
485 \newcommand*{\@glxtr@shortcutsval}{\ifglxtrshortcutsacro\else none\fi}%
```

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

```
486 \define@choicekey{glossaries-extra.sty}{shortcuts}%
487 [\@glxtr@shortcutsval\@glxtr@shortcutsnr]%
488 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
489   \ifcase\@glxtr@shortcutsnr\relax % acronyms
490     \renewcommand*{\@glxtr@setupshortcuts}{%
491       \glxtrshortcutstrue
492       \DefineAcronymSynonyms
493     }%
494   \or % acro
495     \renewcommand*{\@glxtr@setupshortcuts}{%
496       \glxtrshortcutstrue
497       \DefineAcronymSynonyms
498     }%
499   \or % abbreviations
500     \renewcommand*{\@glxtr@setupshortcuts}{%
501       \GlsXtrDefineAbbreviationShortcuts
502     }%
503   \or % abbr
504     \renewcommand*{\@glxtr@setupshortcuts}{%
505       \GlsXtrDefineAbbreviationShortcuts
506     }%
507   \or % other
508     \renewcommand*{\@glxtr@setupshortcuts}{%
509       \GlsXtrDefineOtherShortcuts
510     }%
511   \or % all
512     \renewcommand*{\@glxtr@setupshortcuts}{%
513       \glxtrshortcutstrue
514       \GlsXtrDefineAcShortcuts
515       \GlsXtrDefineAbbreviationShortcuts
516       \GlsXtrDefineOtherShortcuts
517     }%
518   \or % true
519     \renewcommand*{\@glxtr@setupshortcuts}{%
520       \glxtrshortcutstrue
521       \GlsXtrDefineAcShortcuts
522       \GlsXtrDefineAbbreviationShortcuts
523       \GlsXtrDefineOtherShortcuts
524     }%
```

```

525 \or % ac
526 \renewcommand*{\@glsxtr@setupshortcuts}{%
527 \glsacrshortcutstrue
528 \GlsXtrDefineAcShortcuts
529 }%

Leave none and false as last option.
530 \else % none, false
531 \renewcommand*{\@glsxtr@setupshortcuts}{}%
532 \fi
533 }

lsxtr@doaccsupp
534 \newcommand*{\@glsxtr@doaccsupp}{}

accsupp If accsupp, load glossaries-accsupp package.
535 \@glsxtr@declareoption{accsupp}{%
536 \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

GlossaryWarning Warning text displayed in document if the external glossary file given by the argument is missing.
537 \newcommand{\glsxtrNoGlossaryWarning}[1]{%
538 \@glsxtr@defaultnoglossarywarning{#1}%
539 }

omissingglsstext If true, suppress the text produced if the external glossary file is missing.
540 \define@choicekey{glossaries-extra.sty}{nomissingglsstext}
541 [\@glsxtr@nomissingglsstextval\@glsxtr@nomissingglsstextnr]%
542 {true,false}[true]{%
543 \ifcase\@glsxtr@nomissingglsstextnr\relax % true
544 \renewcommand{\glsxtrNoGlossaryWarning}[1]{\null}%
545 \else % false
546 \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
547 \@glsxtr@defaultnoglossarywarning{#1}%
548 }%
549 \fi
550 }

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

lsxtr@redefstyles
551 \newcommand*{\@glsxtr@redefstyles}{}

stylemods
552 \define@key{glossaries-extra.sty}{stylemods}[default]{%
553 \ifstrequal{#1}{default}%
554 {%
555 \renewcommand*{\@glsxtr@redefstyles}{%
556 \RequirePackage{glossaries-extra-stylemods}}%

```

```

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

```

glsxtr@do@style

```

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

```

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

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

591 \setglossarystyle{#1}%
592 }%
593 }

```

`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.

```
594 \newcommand*{\glstr@inc@wrglossaryctr}[1]{}
```

`ocationHyperlink`

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

The first two arguments are always control sequences.

```
595 \newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
596   \glstrhyperlink{#1#2#3}{#3}%
597 }
```

`cationhyperlink`

```
598 \newcommand*{@glstr@wrglossary@locationhyperlink}[3]{%
599   \pageref{wrglossary.#3}%
600 }
```

`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.

```
601 \@glstr@declareoption{indexcounter}{%
602   \glstr@doooption{counter=wrglossary}%
603   \ifundef\c@wrglossary
604   {%
605     \newcounter{wrglossary}%
606     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
607   }%
608   {}%
609   \renewcommand*{\glstr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is `wrglossary`.

```
610   \ifdefstring\@gls@counter{wrglossary}%
611   {%
612     \refstepcounter{wrglossary}%
613     \label{wrglossary.\thewrglossary}%
614   }%
615   {}%
616 }%
617 \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
618   \ifdefstring\glsentrycounter{wrglossary}%
619   {%
```

```

620      \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
621    }%
622    {\glsxtrhyperlink{##1##2##3}{##3}}%
623  }%
624 }

sxtrwrglossmark  Marks the place where indexing occurs. Does nothing by default.
625 \newcommand*{\@glsxtrwrglossmark}{}

sxtrwrglossmark  Since \glsadd can be used in the preamble, this action needs to be disabled until the start of
the document.
626 \newcommand*{\@@glsxtrwrglossmark}{}
627 \AtBeginDocument{\renewcommand*{\@glsxtrwrglossmark}{\@@glsxtrwrglossmark}}

sxtrwrglossmark  Does nothing by default.
628 \newcommand*{\glsxtrwrglossmark}{\ensuremath{\cdot}}

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

    Pass all other options to glossaries.
648 \DeclareOptionX*{%
649   \expandafter\glsxtr@doption\expandafter{\CurrentOption}}

    Process options.
650 \ProcessOptionsX

    Load glossaries if not already loaded.
651 \RequirePackage{glossaries}

    Load the glossaries-accsupp package if required.
652 \@glsxtr@doaccsupp

```


Redefine \glspostdescription if required.

```
653 \let\@glstr@defpostpunc
```

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

```
654 \def\glsshowtarget#1{%
655   \glstrtitleorpdforheading
656   {%
657     \ifmmode
658       \texttt{\small [#1]}%
659     \else
660       \ifinner
661         \texttt{\small [#1]}%
662       \else
663         \marginpar{\texttt{\small #1}}%
664       \fi
665     \fi
666   }%
667   {[#1]}%
668   {\texttt{\small [#1]}}%
669 }
```

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

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

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

```
671 \newcommand*{\@glstr@doseeglossary}[2]{%
672   \glsdexists{#1}%
673   {%
674     \@glstrwrglossmark
675     \@glstr@org@doseeglossary{#1}{#2}%
676   }%
677 }
```

oindex@glossary

```
678 \newcommand*{\@glstr@dosee@alsoindex@glossary}[2]{%
679   \@glstr@recordsee{#1}{#2}%
680   \@glstr@doseeglossary{#1}{#2}%
681 }
```

@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.)

```
682 \let\@glstr@org@gloautosee\@glo@autosee
```

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

```
683 \if@glstr@autoseeindex
684 \else
```

```

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

```

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

```

692 \ifdef\@glo@autosee
693 {%
694 \renewcommand*{\@glo@autosee}{%
695 \if@glxtr@autoseeindex\@glxtr@org@gloautosee\fi}%
696 }%
697 {}

```

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.

```

698 \renewcommand*{\gls@checkseeallowed}{%
699 \if@glxtr@autoseeindex\@gls@see@noindex\fi
700 }

```

Define abbreviations glossaries if required.

```

701 \@glxtr@abbreviationsdef
702 \let\@glxtr@abbreviationsdef\relax

```

Setup shortcuts if required.

```

703 \@glxtr@setupshortcuts

```

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

```

704 \@glxtr@redef@for@gl@sentries

```

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

```

705 \renewcommand{\glxtr@doooption}[1]{\setupglossaries{#1}}%

```

Now define the user command:

```

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

```

`@@do@wrglossary` Save original definition of `\@@do@wrglossary`.
717 `\let\glxstr@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.
718 `\newcommand*{\glxstr@@do@wrglossary}[1]{%`
719 `\@@glxstrwrglossmark`
720 `\glxstr@inc@wrglossaryctr{#1}%`
721 `\glxstr@org@@do@wrglossary{#1}%`
722 `}`

`saveentrycounter` Save original definition of `\@gls@saveentrycounter`.
723 `\let\glxstr@saveentrycounter\@gls@saveentrycounter`

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

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

`sxtrdialecthook`
725 `\newcommand*{\@glxstrdialecthook}{}`

Set up record option if required.
726 `\glxstr@setup@record`

Disable preamble-only options and switch on the undefined tag at the start of the document.
727 `\AtBeginDocument{%`
728 `\disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%`
729 `\def\@glxstrundeftag{\glxstrundeftag}%`
730 `}`

1.2 Extra Utilities

`rifemptyglossary` `\glxstrifemptyglossary{<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.)

```
731 \newcommand{\glxstrifemptyglossary}[3]{%
732   \ifcsdef{glolist@#1}%
```

```

733 {%
734   \ifcsstring{glolist@#1}{,}{#2}{#3}%
735 }%
736 {%
737   \glstrundefaction{Glossary type ‘#1’ doesn’t exist}{}%
738   #2%
739 }%
740 }

```

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

```

741 \newcommand*{\glxtrifkeydefined}[3]{%
742   \key@ifundefined{glossentry}{#1}{#3}{#2}%
743 }

```

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

```

744 \newcommand*{\glxtrprovidestoragekey}{%
745   \@ifstar\sglsxtr@provide@storagekey\glxtr@provide@storagekey
746 }

```

vide@storagekey Unstarred version.

```

747 \newcommand*{\@glxtr@provide@storagekey}[3]{%
748   \key@ifundefined{glossentry}{#1}%
749   {%
750     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
751     \appto\@gls@keymap{,{#1}{#1}}%
752     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
753     \appto\@newglossaryentryposthook{%
754       \letcs{\@glo@tmp}{@glo@#1}%
755       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
756     }%

```

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

```

757   \ifblank{#3}
758   {}%
759   {%
760     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
761   }%
762 }%
763 {%

```

Provide the no-link command if not already defined.

```

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

```

vide@storagekey Starred version.

```
771 \newcommand*{\s@glxtr@provide@storagekey}[1]{%
772   \key@ifundefined{glossentry}{#1}%
773   {%
774     \expandafter\newcommand\expandafter*\expandafter
775     {\csname gls@assign@#1@field\endcsname}[2]{%
776       \@@gls@expand@field{##1}{#1}{##2}%
777     }%
778   }%
779   {%
780     \@glxtr@provide@addstoragekey{#1}%
781   }
```

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`

```
782 \newcommand{\GlsXtrFmtField}{useri}
```

`\GlsXtrFmtDefaultOptions`

```
783 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}
```

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

```
784 \newrobustcmd*{\glxtrfmt}{\@ifstar\s@glxtrfmt\@glxtrfmt}
```

`\@glxtrfmt` Unstarred form.

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

`\s@glxtrfmt` Starred form.

```
786 \newcommand*{\s@glxtrfmt}[3][\@glxtrfmt{#1}{#2}{#3}]{%
787   \new@ifnextchar[\s@glxtrfmt{#1}{#2}{#3}]{%
788     {\@glxtrfmt{#1}{#2}{#3}]{}}%
789 }
```

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

```
790 \def\s@glxtrfmt#1#2#3[#4]{\@glxtrfmt{#1}{#2}{#3}{#4}}
```

`\@glxtrfmt` Actual inner working.

```
791 \newcommand*{\@glxtrfmt}[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).

```
792 \begingroup
793   \def\glslabel{#2}%
794   \glsdoifexistsordo{#2}%
```

```

795   {%
796     \ifglshasfield{\GlsXtrFmtField}{#2}%
797     {%
798       \let\do@gl@link@checkfirsthyper\relax
799       \expandafter\@gl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
800       {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
801     }%
802     {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
803   }%
804   {%

```

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

```

805     \begingroup
806       \@gl@setdefault@gl@link@opts
807       \setkeys{gl@link}{\GlsXtrFmtDefaultOptions,#1}%
808       \ifKV@gl@link@noindex\else\gl@sadd{#2}\fi
809     \endgroup
810     \glstrfmtdisplay{@firstofone}{#3}{#4}%
811   }%
812 \endgroup
813 }

```

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).

```

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

```

glxtrentryfmt No link or indexing.

```

815 \ifdef\teorpdfstring
816 {
817   \newcommand*{\glxtrentryfmt}[2]{%
818     \teorpdfstring{\@glxtrentryfmt{#1}{#2}}{#2}%
819   }
820 }
821 {
822   \newcommand*{\glxtrentryfmt}{\@glxtrentryfmt}
823 }

```

@glxtrentryfmt

```

824 \newrobustcmd*{\@glxtrentryfmt}[2]{%
825   \gl@sdoifexistsordo{#1}%
826   {%
827     \ifglshasfield{\GlsXtrFmtField}{#1}%
828     {%
829       \csuse{\glscurrentfieldvalue}{#2}%
830     }%
831     {#2}%
832   }%

```

```
833 {#2}%
834 }
```

`glxtrfieldlistadd` 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.

```
835 \newcommand*{\glxtrfieldlistadd}[3]{%
836   \listcsadd{glo@\glstdetoklabel{#1}@#2}{#3}%
837 }
```

`glxtrfieldlistgadd` Similarly but uses `\listcsgadd`.

```
838 \newcommand*{\glxtrfieldlistgadd}[3]{%
839   \listcsgadd{glo@\glstdetoklabel{#1}@#2}{#3}%
840 }
```

`glxtrfieldlistseadd` Similarly but uses `\listcseadd`.

```
841 \newcommand*{\glxtrfieldlistseadd}[3]{%
842   \listcseadd{glo@\glstdetoklabel{#1}@#2}{#3}%
843 }
```

`glxtrfieldlistxadd` Similarly but uses `\listcsxadd`.

```
844 \newcommand*{\glxtrfieldlistxadd}[3]{%
845   \listcsxadd{glo@\glstdetoklabel{#1}@#2}{#3}%
846 }
```

Now provide commands to iterate over these lists.

`glxtrfielddolistloop`

```
847 \newcommand*{\glxtrfielddolistloop}[2]{%
848   \dolistcsloop{glo@\glstdetoklabel{#1}@#2}%
849 }
```

`glxtrfieldforlistloop`

```
850 \newcommand*{\glxtrfieldforlistloop}[3]{%
851   \forlistcsloop{#3}{glo@\glstdetoklabel{#1}@#2}%
852 }
```

List element tests:

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

```
853 \newcommand*{\glxtrfieldifinlist}[5]{%
854   \ifinlistcs{#3}{glo@\glstdetoklabel{#1}@#2}{#4}{#5}%
855 }
```

`glxtrfieldxifinlist` Expands item.

```
856 \newcommand*{\glxtrfieldxifinlist}[5]{%
857   \xifinlistcs{#3}{glo@\glstdetoklabel{#1}@#2}{#4}{#5}%
858 }
```

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

```
859 \newcommand*{\glxtrforcsvfield}[3]{%
860   \@glxtrifhasfield{#2}{#1}%
861   {%
862     \let\glxtrendfor\@endfortrue
863     \@for\@glxtr@label:=\glscurrentfieldvalue\do
864       {\expandafter#3\expandafter{\@glxtr@label}}}%
865   }%
866 }
```

`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.

```
867 \newrobustcmd{\glxtrifhasfield}{%
868   \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
869 }
```

`lsxtrifhasfield` Unstarred version adds grouping.

```
870 \newcommand{\@glxtrifhasfield}[4]{%
871   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
872 }
```

`lsxtrifhasfield` Starred version omits grouping.

```
873 \newcommand{\s@glxtrifhasfield}[4]{%
874   \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
875   \ifundef\glscurrentfieldvalue
876     {#4}%
877   {%
878     \ifdefempty\glscurrentfieldvalue{#4}{#3}%
879   }%
880 }
```

`rIfFieldNonZero` Designed for numeric fields.

```
881 \newcommand{\GlsXtrIfFieldNonZero}[4]{%
882   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
883 }
```

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

Designed for numeric fields.

```
884 \newcommand{\GlsXtrIfFieldEqNum}[5]{%
885   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
886 }
```


XtrIfFieldCmpNum

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

Designed for numeric fields.

```
887 \newcommand{\GlsXtrIfFieldCmpNum}[6]{%  
888   {%  
889     \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%  
890     \ifundef\glscurrentfieldvalue  
891     {\def\glscurrentfieldvalue{0}}%  
892     {%  
893       \ifdefempty\glscurrentfieldvalue  
894       {\def\glscurrentfieldvalue{0}}%  
895       {}}%  
896     }%  
897     \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi  
898   }%  
899 }
```

sXtrIfFieldUndef

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

Just uses \ifcsundef.

```
900 \newcommand{\GlsXtrIfFieldUndef}[2]{%  
901   \ifcsundef{glo@\glsdetoklabel{#2}@#1}%  
902 }
```

\glxtrusefield

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

```
903 \newcommand*{\glxtrusefield}[2]{%  
904   \@gls@entry@field{#1}{#2}%  
905 }
```

\Glsxtrusefield

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

```
906 \newcommand*{\Glsxtrusefield}[2]{%  
907   \@gls@entry@field{#1}{#2}%  
908 }
```

\glxtrdeffield

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

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

glxtredeffield

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

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

etfieldifexists

```
911 \newcommand*{\glxtrsetfieldifexists}[3]{\glsdodef{#1}{#3}}
```

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

```
912 \newrobustcmd*{\GlsXtrSetField}[3]{%
913   \glstrsetfieldifexists{#1}{#2}%
914   {\csdef{glo@glstoklabel{#1}@#2}{#3}}%
915 }
```

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

```
916 \newrobustcmd*{\GlsXtrLetField}[3]{%
917   \glstrsetfieldifexists{#1}{#2}%
918   {\cslet{glo@glstoklabel{#1}@#2}{#3}}%
919 }
```

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

```
920 \newrobustcmd*{\csGlsXtrLetField}[3]{%
921   \glstrsetfieldifexists{#1}{#2}%
922   {\csletcs{glo@glstoklabel{#1}@#2}{#3}}%
923 }
```

`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.

```
924 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
925   \glstrsetfieldifexists{#1}{#2}%
926   {\csletcs{glo@glstoklabel{#1}@#2}{glo@glstoklabel{#3}@#4}}%
927 }
```

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

```
928 \newrobustcmd*{\gGlsXtrSetField}[3]{%
929   \glstrsetfieldifexists{#1}{#2}%
930   {\csgdef{glo@glstoklabel{#1}@#2}{#3}}%
931 }
```

`xGlsXtrSetField`

```
932 \newrobustcmd*{\xGlsXtrSetField}[3]{%
933   \glstrsetfieldifexists{#1}{#2}%
934   {\protected@csxdef{glo@glstoklabel{#1}@#2}{#3}}%
935 }
```

`eGlsXtrSetField`

```
936 \newrobustcmd*{\eGlsXtrSetField}[3]{%
937   \glstrsetfieldifexists{#1}{#2}%
938   {\protected@csedef{glo@glstoklabel{#1}@#2}{#3}}%
939 }
```

`XtrIfFieldEqStr`

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

```

941 \glstrifhasfield{#1}{#2}%
942 {%
943   \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
944   }%
945   {#5}%
946 }

```

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

```

947 \newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
948   \glstrifhasfield{#1}{#2}%
949   {%
950     \protected@edef\@glstmp{#3}%
951     \ifdefequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
952     }%
953     {#5}%
954 }

```

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

```

955 \newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
956   \glstrifhasfield{#1}{#2}%
957   {%
958     \protected@edef\@glstmp{\glscurrentfieldvalue}%
959     \let\glscurrentfieldvalue\@glstmp
960     \protected@edef\@glstmp{#3}%
961     \ifdefequal{\glscurrentfieldvalue}{\@glstmp}{#4}{#5}%
962     }%
963     {#5}%
964 }

```

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

```

965 \ifdef\GlsEntryCounterLabelPrefix
966 {%
967   \newcommand*{\glstrpageref}[1]{%
968     \ifglssentrycounter
969       \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
970     \else
971       \ifglssubentrycounter
972         \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
973       \else
974         \gls{#1}%
975       \fi
976     \fi
977   }
978 }%
979 {%
980   \newcommand*{\glstrpageref}[1]{%
981     \ifglssentrycounter

```

```

982     \pageref{glsentry-\glsdetoklabel{#1}}%
983   \else
984     \ifglssubentrycounter
985       \pageref{glsentry-\glsdetoklabel{#1}}%
986     \else
987       \gls{#1}%
988     \fi
989   \fi
990 }
991 }%

```

lossarypreamble

```

992 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
993   \ifcsdef{glolist@#1}%
994   {%
995     \ifcsundef{@glossarypreamble@#1}%
996     {\csdef{@glossarypreamble@#1}{}}%
997   }%
998   \csappto{@glossarypreamble@#1}{#2}%
999 }%
1000 {%
1001   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1002 }%
1003 }

```

lossarypreamble

```

1004 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1005   \ifcsdef{glolist@#1}%
1006   {%
1007     \ifcsundef{@glossarypreamble@#1}%
1008     {\csdef{@glossarypreamble@#1}{}}%
1009   }%
1010   \cspreteto{@glossarypreamble@#1}{#2}%
1011 }%
1012 {%
1013   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1014 }%
1015 }

```

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.

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 `\glxtrpostlongdescription` instead.

ewglossaryentry

```
1016 \renewcommand*{\longnewglossaryentry}{%
1017 \ifstar\@glxtr@s@longnewglossaryentry\@glxtr@longnewglossaryentry
1018 }
```

ewglossaryentry Starred version.

```
1019 \newcommand{\@glxtr@s@longnewglossaryentry}[3]{%
1020 \glsdoifnoexists{#1}%
1021 {%
1022 \bgroup
1023 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1024 \long\def\@newglossaryentryprehook{%
1025 \long\def\@glo@desc{#3}%
1026 \@org@newglossaryentryprehook
1027 }%
1028 \renewcommand*{\gls@assign@desc}[1]{%
1029 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1030 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1031 }
1032 \gls@defglossaryentry{#1}{#2}%
1033 \egroup
1034 }%
1035 }
```

ewglossaryentry Unstarred version.

```
1036 \newcommand{\@glxtr@longnewglossaryentry}[3]{%
1037 \glsdoifnoexists{#1}%
1038 {%
1039 \bgroup
1040 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1041 \long\def\@newglossaryentryprehook{%
1042 \long\def\@glo@desc{#3\glxtrpostlongdescription}%
1043 \@org@newglossaryentryprehook
1044 }%
1045 \renewcommand*{\gls@assign@desc}[1]{%
1046 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1047 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1048 }
1049 \gls@defglossaryentry{#1}{#2}%
1050 \egroup
1051 }%
1052 }
```

The following is different from the base glossaries.sty:

```
1047 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1048 }
1049 \gls@defglossaryentry{#1}{#2}%
1050 \egroup
1051 }%
1052 }
```

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

```
1053 \newcommand*{\glxtrpostlongdescription}{\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.

```
1054 \renewcommand{\newignoredglossary}{%
1055   \@ifstar\glxtr@s@newignoredglossary\glxtr@org@newignoredglossary
1056 }
```

ignoredglossary The original definition is patched to check for existence.

```
1057 \newcommand*{\glxtr@org@newignoredglossary}[1]{%
1058   \ifcsdef{glolist@#1}
1059   {%
1060     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1061   }%
1062   {%
1063     \ifdefempty\@ignored@glossaries
1064     {%
1065       \edef\@ignored@glossaries{#1}%
1066     }%
1067     {%
1068       \eappto\@ignored@glossaries{,#1}%
1069     }%
1070     \csgdef{glolist@#1}{,}%
1071     \ifcsundef{gls@#1@entryfmt}%
1072     {%
1073       \defglsentryfmt[#1]{\glsentryfmt}%
1074     }%
1075     {}%
1076     \ifdefempty\@gls@nohyperlist
1077     {%
1078       \renewcommand*{\@gls@nohyperlist}{#1}%
1079     }%
1080     {%
1081       \eappto\@gls@nohyperlist{,#1}%
1082     }%
1083   }%
1084 }
```

ignoredglossary Starred form.

```
1085 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
1086   \ifcsdef{glolist@#1}
1087   {%
1088     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1089   }%
1090   {%
1091     \ifdefempty\@ignored@glossaries
1092     {%
1093       \edef\@ignored@glossaries{#1}%
1094     }%
1095     {%
1096       \eappto\@ignored@glossaries{,#1}%
1097     }%
1098   }%
1099 }
```

```

1098 \csgdef{glolist@#1}{,}%
1099 \ifcsundef{gls@#1@entryfmt}%
1100 {%
1101     \defglsentryfmt[#1]{\glsentryfmt}%
1102 }%
1103 {}%
1104 }%
1105 }

```

`\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.

```

1106 \glsifusetranslator
1107 {%
1108 \renewcommand*{\glsettoctitle}[1]{%
1109     \ifcsdef{gls@tr@set@#1@toctitle}%
1110     {%
1111         \csuse{gls@tr@set@#1@toctitle}%
1112     }%
1113     {%
1114         \ifcsdef{@glotype@#1@title}%
1115         {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1116         {\def\glossarytoctitle{\glossarytitle}}%
1117     }%
1118 }%
1119 }
1120 {
1121 \renewcommand*{\glsettoctitle}[1]{%
1122     \ifcsdef{@glotype@#1@title}%
1123     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1124     {\def\glossarytoctitle{\glossarytitle}}%
1125 }
1126 }

```

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

```

1127 \newcommand{\provideignoredglossary}{%
1128 \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1129 }

```

`ignoredglossary` Unstarred version.

```

1130 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1131 \ifcsdef{glolist@#1}
1132 {}%
1133 {%
1134     \ifdefempty\@ignored@glossaries
1135     {%
1136         \edef\@ignored@glossaries{#1}%
1137     }%
1138     {%
1139         \eappto\@ignored@glossaries{, #1}%

```

```

1140 }%
1141 \csgdef{glolist@#1}{,}%
1142 \ifcsundef{gls@#1@entryfmt}%
1143 {%
1144   \defglsentryfmt[#1]{\glsentryfmt}%
1145 }%
1146 {}%
1147 \ifdefempty{@gls@nohyperlist
1148 {%
1149   \renewcommand*{@gls@nohyperlist}{#1}%
1150 }%
1151 {%
1152   \eappto{@gls@nohyperlist}{, #1}%
1153 }%
1154 }%
1155 }

```

`ignoredglossary` Starred form.

```

1156 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1157   \ifcsdef{glolist@#1}
1158   {%
1159   {%
1160     \ifdefempty{@ignored@glossaries
1161     {%
1162       \edef{@ignored@glossaries{#1}%
1163     }%
1164     {%
1165       \eappto{@ignored@glossaries}{, #1}%
1166     }%
1167     \csgdef{glolist@#1}{,}%
1168     \ifcsundef{gls@#1@entryfmt}%
1169     {%
1170       \defglsentryfmt[#1]{\glsentryfmt}%
1171     }%
1172     {}%
1173   }%
1174 }

```

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

```

1175 \newcommand*{\glxtrcopytoglossary}[2]{%
1176   \glsoifexists{#1}%
1177   {%
1178     \ifcsdef{glolist@#2}
1179     {%
1180       \cseappto{glolist@#2}{#1,}%
1181     }%
1182     {%
1183       \glxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%

```



```

1184 }%
1185 }%
1186 }

```

1.3.1 Existence Checks

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

```

1187 \renewcommand{\glsdoifexists}[2]{%
1188   \ifglstryexists{#1}{#2}%
1189   {%

```

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

```

1190   \edef\glslabel{\glsdetoklabel{#1}}%
1191   \glstrundefaction{Glossary entry '\glslabel'
1192     has not been defined}{You need to define a glossary entry before
1193     you can reference it.}%
1194   }%
1195 }

```

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

```

1196 \renewcommand{\glsdoifnoexists}[2]{%
1197   \ifglstryexists{#1}{%
1198     \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1199       has already been defined}{}}{#2}%
1200 }

```

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

```

1201 \ifdef\glsdoifexistsordo
1202 {%
1203   \renewcommand{\glsdoifexistsordo}[3]{%
1204     \ifglstryexists{#1}{#2}%
1205     {%
1206       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1207         has not been defined}{You need to define a glossary entry
1208         before you can use it.}%
1209       #3%
1210     }%
1211   }%
1212 }
1213 {%
1214   \glstr@warnonexistsordo\glsdoifexistsordo
1215   \newcommand{\glsdoifexistsordo}[3]{%
1216     \ifglstryexists{#1}{#2}%
1217     {%
1218       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1219         has not been defined}{You need to define a glossary entry
1220         before you can use it.}%

```

```

1221      #3%
1222    }%
1223  }%
1224 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

1225 \ifdef\doifglossarynoexistsordo
1226 {%
1227   \renewcommand{\doifglossarynoexistsordo}[3]{%
1228     \ifglossaryexists{#1}%
1229     {%
1230       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1231       #3%
1232     }%
1233     {#2}%
1234   }%
1235 }
1236 {%
1237   \glstr@warnonexistsordo\doifglossarynoexistsordo
1238   \newcommand{\doifglossarynoexistsordo}[3]{%
1239     \ifglossaryexists{#1}%
1240     {%
1241       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1242       #3%
1243     }%
1244     {#2}%
1245   }%
1246 }
1247

```

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.

```

1248 \appto\@newglossaryentryposthook{%
1249   \ifdefvoid\@glo@see
1250   {\csxdef{glo@\@glo@label @see}{}}%
1251   {%
1252     \csxdef{glo@\@glo@label @see}{\@glo@see}%
1253     \if@glstr@autoseeindex
1254       \@glstr@autoindexcrossrefs
1255     \fi
1256   }%
1257 }
1258 \appto\@gls@keymap{,{see}{see}}

```

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

```

1259 \newcommand*{\glstrusesee}[1]{%

```

```

1260 \glsdoifexists{#1}%
1261 {%
1262   \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1263   \ifdefempty\@glo@see
1264     {}%
1265     {%
1266       \expandafter\glstr@usesee\@glo@see\end@glstr@usesee
1267     }%
1268   }%
1269 }

```

\glstr@usesee

```

1270 \newcommand*{\glstr@usesee}[1][\seename]{%
1271   \@glstr@usesee{#1}%
1272 }

```

\@glstr@usesee

```

1273 \def\@glstr@usesee[#1]#2\end@glstr@usesee{%
1274   \glstruseseeformat{#1}{#2}%
1275 }

```

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

```

1276 \newcommand*{\glstruseseeformat}[2]{%
1277   \glseeformat{#1}{#2}{}%
1278 }

```

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

```

1279 \renewcommand*{\glseeitemformat}[1]{%
1280   \ifglshashshort{\glslabel}{\glssaccesstext{#1}}{\glssaccessname{#1}}%
1281 }

```

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

```

1282 \newcommand*{\glstruseseealso}[1]{%
1283   \glsdoifexists{#1}%
1284   {%
1285     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
1286     \ifdefempty\@glo@see
1287       {}%
1288       {%
1289         \expandafter\glstruseseealsoformat\expandafter{\@glo@see}%
1290       }%
1291     }%
1292 }

```

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

```
1293 \newcommand*{\glxtruseeseealsoformat}[1]{%
1294   \glssseeformat[\seealsoformat]{#1}{}%
1295 }
```

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

```
1296 \newrobustcmd{\glxtrseelist}[1]{%
1297   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1298 }
```

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

```
1299 \providecommand{\seealsoformat}{see also}
```

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

```
1300 \ifdefined\@xdycrossrefhook
1301 {
```

Add the cross-reference class definition to the hook.

```
1302   \appto\@xdycrossrefhook{%
1303     \write\glswrite{(define-crossref-class \string"seealso\string"
1304       :unverified )}%
1305     \write\glswrite{(markup-crossref-list
1306       :class \string"seealso\string"^^J\space\space\space
1307       :open \string"\string\glxtruseeseealsoformat\glsoopenbrace\string"
1308       :close \string"\glsclosebrace\string")}%
1309   }
```

Append to class list.

```
1310   \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

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

```
1311   \newrobustcmd*{\glxtrindexseealso}[2]{%
1312     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1313       \@glxtr@recordsee{#1}{#2}%
1314     \fi
1315     \glsdofexists{#1}%
1316     {%
1317       \@@glxtrwrglossmark
1318       \def\@glx@xref{#2}%
1319       \@onelevel@sanitize\@glx@xref
1320       \@glx@checkmkidxchars\@glx@xref
1321       \glsglossary{\csname glo@#1@type\endcsname}{%
1322         (indexentry
1323           :tkey (\csname glo@#1@index\endcsname)
```

```

1324         :xref (\string"\@gls@xref\string")
1325         :attr \string"seealso\string"
1326     )
1327 }%
1328 }%
1329 }
1330 }
1331 {
    xindy not in use or glossaries version too old to support this.
1332 \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
1333 }

```

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`.

```

1334 \ifdef\gls@set@xr@key
1335 {

```

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

```

1336 \define@key{glossentry}{alias}{%
1337   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1338 }
1339 \define@key{glossentry}{seealso}{%
1340   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1341 }

```

Add to the key mappings.

```

1342 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1343 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1344 \appto\@newglossaryentryposthook{%
1345   \ifdefvoid\@glo@seealso
1346     {\csxdef{glo@\@glo@label @seealso}{}}%
1347     {%
1348       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1349       \if@glsxtr@autoseeindex
1350         \@glsxtr@autoindexcrossrefs
1351       \fi
1352     }%

```

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

```

1353 \ifdefvoid\@glo@alias

```

```

1354     {\csxdef{glo@\@glo@label @alias}{}}}%
1355     {%
1356     \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1357     }%
1358 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

1359 \newcommand*{\glxtralias}[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1360 \newcommand*{\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1361 \appto\@glo@autoseehook{%
1362   \ifdefvoid\@glo@alias
1363   {%
1364     \ifdefvoid\@glo@seealso
1365     }%
1366     {%
1367       \edef\@do@glsssee{\noexpand\glxtrindexseealso
1368         {\@glo@label}{\@glo@seealso}}%
1369       \@do@glsssee
1370     }%
1371   }%
1372   {%

```

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

```

1373   \ifdefvoid\@glo@see
1374   {%
1375     \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1376     \@do@glsssee
1377   }%
1378   }%
1379 }%
1380 }%
1381 }
1382 {

```

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

`\glxtralias`

```

1383 \glsaddstoragekey*{alias}{}{\glxtralias}

```

`trseealsolabels`

```

1384 \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.

```
1385 \appto\@newglossaryentryposthook{%
1386   \ifcsvoid{glo@\@glo@label @alias}%
1387   {%
1388     \ifcsvoid{glo@\@glo@label @seealso}%
1389     {}%
1390     {%
1391       \edef\@do@glsee{\noexpand\glstrindexseealso
1392         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1393       \@do@glsee
1394     }%
1395   }%
1396   {%
```

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

```
1397   \ifdefvoid\@glo@see
1398   {%
1399     \edef\@do@glsee{\noexpand\glsee
1400       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1401     \@do@glsee
1402   }%
1403   {}%
1404   }%
1405 }

1406 }
```

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

```
1407 \AtEndDocument{\if@glstrindexcrossrefs\glstraddallcrossrefs\fi}
```

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

```
1408 \newcommand*\@glstraddallcrossrefs{%
1409   \forallglossaries{\@glo@type}%
1410   {%
1411     \forglsentries[\@glo@type]{\@glo@label}%
1412     {%
1413       \ifglused{\@glo@label}%
1414       {\expandafter\@glstr@addunusedxrefs\expandafter{\@glo@label}}}%
1415     }%
1416   }%
1417 }
```

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

```
1418 \newcommand*\@glstr@addunusedxrefs[1]{%
1419   \letcs{\@glo@see}{glo\@glsetoklabel{#1}@see}%
1420   \ifdefvoid\@glo@see
1421   {%
```

```

1422 {%
1423   \expandafter\glsxtr@addunused\@glo@see\@end\glsxtr@addunused
1424 }%
1425 \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
1426 \ifdefvoid\@glo@see
1427 {}%
1428 {%
1429   \expandafter\glsxtr@addunused\@glo@see\@end\glsxtr@addunused
1430 }%
1431 }

```

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

```

1432 \newcommand*{\glsxtr@addunused}[1][]{%
1433   \@glsxtr@addunused
1434 }

```

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

```

1435 \def\@glsxtr@addunused#1\@end\glsxtr@addunused{%
1436   \@for\@glsxtr@label:=#1\do
1437   {%
1438     \ifglsused{\@glsxtr@label}{}%
1439     {%
1440       \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1441       \glsunset{\@glsxtr@label}%
1442       \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1443     }%
1444   }%
1445 }

```

`\glsxtrunusedformat`

```

1446 \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`.

```

1447 \ifdef\gls@begindocdefs
1448 {%
1449   \renewcommand*{\gls@begindocdefs}{%
1450     \ifnum\@glsxtr@docdefval=1\relax
1451       \@gls@enablesavenonumberlist
1452       \edef\@gls@restoreat{%
1453         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1454       \makeatletter
1455       \InputIfFileExists{\jobname.glsdefs}{}{}%
1456       \@gls@restoreat
1457       \undef\@gls@restoreat

```



```

1458     \gls@defdocnewglossaryentry
1459   \fi
1460 }
1461 }
1462 {}

```

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 allow with the record option.

```

1463 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1464 \renewcommand{\makenoidxglossaries}{%
1465   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1466   {%
1467     \glsxtr@orgmakenoidxglossaries

```

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

```

1468   \renewcommand{\do@seeglossary}[2]{%
1469     \@glsxtrwrglossmark
1470     \edef\@gls@label{\glsdetoklabel{##1}}%
1471     \protected@write\@auxout{}{%
1472       \string\@gls@reference
1473       {\csname glo@\@gls@label @type\endcsname}%
1474       {\@gls@label}%
1475       {%
1476         \string\glsseeformat##2}%
1477       }%
1478     }%
1479   }%

```

Check for docdefs=restricted:

```

1480   \if@glsxtrdocdefrestricted

```

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

```

1481     \renewcommand*\@gls@reference}[3]{%
1482       \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
1483       \ifinlistcs{##2}{@glsref@##1}%
1484       {}%
1485       {\listcsgadd{@glsref@##1}{##2}}%
1486       \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1487       {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1488       {}%
1489       \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1490     }%
1491   \else

```

Disable document definitions.

```

1492     \@glsxtrdocdeffalse
1493   \fi
1494   \disable@keys{glossaries-extra.sty}{docdef}%

```

```

1495 }%
1496 {%
1497   \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1498     not permitted\MessageBreak
1499     with record=\@glstr@record@setting\space package option}%
1500   {You may only use \string\makenoidxglossaries\ space with the
1501     record=off option}%
1502 }%
1503 }

```

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

```

1504 \renewcommand*{\gls@defdocnewglossaryentry}{%
1505   \ifcase\@glstr@docdefval
1506     docdef=false:
1507     \renewcommand*{\newglossaryentry}[2]{%
1508       \PackageError{glossaries-extra}{Glossary entries must
1509       be \MessageBreak defined in the preamble with \MessageBreak
1510       package option ‘docdef=false’\MessageBreak(consider using
1511       ‘docdef=restricted’)}{Move your glossary definitions to
1512       the preamble. You can also put them in a \MessageBreak separate file
1513       and load them with \string\loadglsentries.}%
1514     }%
1515   \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.

```

1515   \let\gls@checkseeallowed\relax
1516   \let\newglossaryentry\newglossaryentry
1517 \or

```

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

```

1518   \let\gls@checkseeallowed\relax
1519 \fi
1520 }%

```

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`

```

1521 \newcommand*{\GlsXtrEnableOnTheFly}{%
1522   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1523 }

```

`\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.)

```

1524 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1525   \renewcommand*{\glsdetoklabel}[1]{%
1526     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1527     {%
1528       \expandafter\detokenize\expandafter{##1}%
1529     }%
1530     {\detokenize{##1}}}%
1531   }%
1532   \@GlsXtrEnableOnTheFly
1533 }
1534 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1535   \expandafter\if\glsbackslash#1%
1536     #3%
1537   \else
1538     #4%
1539   \fi
1540 }

```

sxtrstarflywarn

```

1541 \newcommand*{\glsxtrstarflywarn}{%
1542   \GlossariesExtraWarning{Experimental starred version of
1543   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1544   read the warnings in the glossaries-extra user manual)}}%
1545 }

```

rEnableOnTheFly

```

1546 \newcommand*{\@GlsXtrEnableOnTheFly}{%

```

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

These definitions are all assigned the category given by:

`\glsxtrcat`

```

1547 \newcommand*{\glsxtrcat}{general}

```

`\glsxtr`

```

1548 \newcommand*{\glsxtr}[1][]{%
1549   \def\glsxtr@keylist{##1}%
1550   \@glsxtr
1551 }

```

`\@glsxtr`

```

1552 \newcommand*{\@glsxtr}[2][]{%
1553   \ifglsentryexists{##2}%
1554   {%
1555     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1556   }%
1557   {%
1558     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,

```

```

1559     description={\nopostdesc},##1}%
1560 }%
1561 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1562 }

```

\Glsxtr

```

1563 \newcommand*\Glsxtr}[1] [] {%
1564   \def\glsxtr@keylist{##1}%
1565   \@Glsxtr
1566 }

```

\@Glsxtr

```

1567 \newcommand*\@Glsxtr}[2] [] {%
1568   \ifglsentryexists{##2}%
1569   {%
1570     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1571   }%
1572   {%
1573     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1574       description={\nopostdesc},##1}%
1575   }%
1576   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1577 }

```

\glsxtrpl

```

1578 \newcommand*\glsxtrpl}[1] [] {%
1579   \def\glsxtr@keylist{##1}%
1580   \@glsxtrpl
1581 }

```

\@glsxtrpl

```

1582 \newcommand*\@glsxtrpl}[2] [] {%
1583   \ifglsentryexists{##2}%
1584   {%
1585     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1586   }%
1587   {%
1588     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1589       description={\nopostdesc},##1}%
1590   }%
1591   \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
1592 }

```

\Glsxtrpl

```

1593 \newcommand*\Glsxtrpl}[1] [] {%
1594   \def\glsxtr@keylist{##1}%
1595   \@Glsxtrpl
1596 }

```

\@Glsxtrpl

```

1597 \newcommand*{\@Glsxtrpl}[2][]{%
1598   \ifglentryexists{##2}
1599   {%
1600     \ifblank{##1}{\@GlsXtrWarning{##1}{##2}}%
1601   }%
1602   {%
1603     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1604       description={\nopostdesc},##1}%
1605   }%
1606   \expandafter\Glspl\expandafter[\glxtr@keylist]{##2}%
1607 }

```

\GlsXtrWarning

```

1608 \newcommand*{\GlsXtrWarning}[2]{%
1609   \def\@glxtr@optlist{##1}%
1610   \@onelevel@sanitize\@glxtr@optlist
1611   \GlossariesExtraWarning{The options ‘\@glxtr@optlist’ have
1612     been ignored for entry ‘##2’ as it has already been defined}%
1613 }

```

Disable commands after the glossary:

```

1614 \renewcommand\@printglossary[2]{%
1615   \def\@glxtr@printglossopts{##1}%
1616   \@glxtr@orgprintglossary{##1}{##2}%
1617   \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
1618   \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1619   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1620   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1621 }

```

abledflycommand

```

1622 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1623   \PackageError{glossaries-extra}%
1624   {\string##1\space can't be used after any of the \MessageBreak
1625     glossaries have been displayed}%
1626   {The on-the-fly commands enabled by
1627     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1628     before the glossaries. If you want to use any entries \MessageBreak
1629     after any of the glossaries, you must use the standard \MessageBreak
1630     method of first defining the entry and then using the \MessageBreak
1631     entry with commands like \string\gls}%
1632   @@glxtr@disabledflycommand
1633 }%
1634 \newcommand*{\@@glxtr@disabledflycommand}[2][]{##2}

```

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

```

1635 \let\GlsXtrEnableOnTheFly\relax
1636 }
1637 \@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.

`\current@style` Initialise the current style to the default style.

```
1638 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}
```

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

`\etglossarystyle`

```
1639 \renewcommand*{\setglossarystyle}[1]{%
1640   \ifcsundef{@glstyle@#1}%
1641   {%
1642     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1643   }%
1644   {%
1645     \csname @glstyle@#1\endcsname
```

Only set the current style if it exists.

```
1646     \protected@edef\@glxtr@current@style{#1}%
1647   }%
1648   \ifx\@glossary@default@style\relax
1649     \protected@edef\@glossary@default@style{#1}%
1650   \fi
1651 }
```

In case we have an old version of `glossaries`:

```
1652 \ifdef\@glossary@default@style
1653 {}
1654 {%
1655   \let\@glossary@default@style\relax
1656 }
```

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

```
1657 \ifdef\glslistdottedwidth
1658 {%
1659   \ifdim\glslistdottedwidth=.5\hsize
1660     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1661     \AtBeginDocument{%
1662       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1663         \setlength{\glslistdottedwidth}{.5\columnwidth}%
1664       \fi
1665     }%
1666   \fi
1667 }
1668 {}%
```

Similarly for `\glstdescwidth`:

\glsdescwidth

```
1669 \ifdef\glsdescwidth
1670 {%
1671   \ifdim\glsdescwidth=.6\hsize
1672     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1673     \AtBeginDocument{%
1674       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1675         \setlength{\glsdescwidth}{.6\columnwidth}%
1676       \fi
1677     }%
1678   \fi
1679 }
1680 {}%
```

and for \glspagelistwidth:

\glspagelistwidth

```
1681 \ifdef\glspagelistwidth
1682 {%
1683   \ifdim\glspagelistwidth=.1\hsize
1684     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1685     \AtBeginDocument{%
1686       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1687         \setlength{\glspagelistwidth}{.1\columnwidth}%
1688       \fi
1689     }%
1690   \fi
1691 }
1692 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```
1693 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1694 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1695   \glsnonumberlistfalse
1696   \renewcommand*{\glossaryentrynumbers}[1]{%
1697     \ifglsentryexists{\glscurrententrylabel}%
1698     {%
1699       \@glsxtrpreloctag
1700       \GlsXtrFormatLocationList{#1}%
1701       \@glsxtrpostloctag
1702       \gls@save@numberlist{#1}%
1703     }{}%
1704   }%
1705 \else
1706   \glsnonumberlisttrue
1707   \renewcommand*{\glossaryentrynumbers}[1]{%
1708     \ifglsentryexists{\glscurrententrylabel}%
1709     {%
1710       \gls@save@numberlist{#1}%

```

```

1711     }{}%
1712 }%
1713 \fi

```

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

```

1714 \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`

```

1715 \newcommand*\GlsXtrEnablePreLocationTag}[2]{%
1716   \let\@glstrpreloctag\@glstrpreloctag
1717   \let\@glstrpostloctag\@glstrpostloctag
1718   \renewcommand*\@glstr@pagetag{#1}%
1719   \renewcommand*\@glstr@pagetag{#2}%
1720   \renewcommand*\@glstr@savepreloctag}[2]{%
1721     \csgdef{\@glstr@preloctag@##1}{##2}%
1722   }%
1723   \renewcommand*\@glstr@doloctag}{%
1724     \ifcsundef{\@glstr@preloctag\@glscurrententrylabel}%
1725     {%
1726       \GlossariesWarning{Missing pre-location tag for ‘\@glscurrententrylabel’.
1727         Rerun required}%
1728     }%
1729     {%
1730       \csuse{\@glstr@preloctag\@glscurrententrylabel}%
1731     }%
1732   }%
1733 }
1734 \@onlypreamble\GlsXtrEnablePreLocationTag

```

`\glstrpreloctag`

```

1735 \newcommand*\@glstrpreloctag{%
1736   \let\@glstr@org@delimN\delimN
1737   \let\@glstr@org@delimR\delimR
1738   \let\@glstr@org@glsgignore\glsgignore

```

`\gdef` is required as the delimiters may occur inside a scope.

```

1739   \gdef\@glstr@thisloctag{\@glstr@pagetag}%
1740   \renewcommand*\@delimN{%
1741     \gdef\@glstr@thisloctag{\@glstr@pagetag}%
1742     \@glstr@org@delimN}%
1743   \renewcommand*\@delimR{%
1744     \gdef\@glstr@thisloctag{\@glstr@pagetag}%

```



```

1745 \@glsxtr@org@delimR}%
1746 \renewcommand*{\glsignore}[1]{%
1747 \gdef\@glsxtr@thisloctag{\relax}%
1748 \@glsxtr@org@glsignore{##1}}%
1749 \@glsxtr@doloctag
1750 }

```

glsxtrpreloctag

```

1751 \newcommand*{\@glsxtrpreloctag}{%

```

@glsxtr@pagetag

```

1752 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

1753 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

1754 \newcommand*{\@glsxtrpostloctag}{%
1755 \let\delimN\@glsxtr@org@delimN
1756 \let\delimR\@glsxtr@org@delimR
1757 \let\glsignore\@glsxtr@org@glsignore
1758 \protected@write\@auxout{}%
1759 {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
1760 }

```

lsxtrpostloctag

```

1761 \newcommand*{\@glsxtrpostloctag}{%

```

lsxtr@preloctag

```

1762 \newcommand*{\@glsxtr@savepreloctag}[2]{%
1763 \protected@write\@auxout{}%
1764 \string\providecommand\string\@glsxtr@savepreloctag[2]{%

```

glsxtr@doloctag

```

1765 \newcommand*{\@glsxtr@doloctag}{%

```

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

```

1766 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
1767 \XKV@plfalse
1768 \XKV@sttrue
1769 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1770 {%
1771 \csname glsnonumberlist\XKV@resa\endcsname
1772 \ifglsnonumberlist
1773 \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1774 \else
1775 \def\glossaryentrynumbers##1{%

```

```

1776      \@glstrpreloctag
1777      \GlsXtrFormatLocationList{##1}%
1778      \@glstrpostloctag
1779      \gls@save@numberlist{##1}}%
1780  \fi
1781 }%
1782 }

```

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.

```

1783 \renewcommand*{\glsentryfmt}{%
1784   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}%
1785   \glsifregular{\glslabel}%
1786   {\glstrregularfont{\glsgenentryfmt}}%
1787   {%
1788     \ifglshasshort{\glslabel}%
1789     {\glstrabbreviationfont{\glstrgenabbrvfmt}}%
1790     {\glstrregularfont{\glsgenentryfmt}}%
1791   }%
1792 }

```

glstrregularfont Font used for regular entries.

```
1793 \newcommand*{\glstrregularfont}[1]{#1}
```

glstrabbreviationfont Font used for abbreviation entries.

```
1794 \newcommand*{\glstrabbreviationfont}[1]{#1}
```

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

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

```
1795 \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).

```

1796   \@glstr@record{#2}{#3}{\glslink}%
1797   \glsdoifexists{#3}%
1798   {%

```

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).

```

1799 \let\glstrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1800 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1801 \def\glscustomtext{#4}%
1802 \@glstr@field@linkdefs
1803 #1%
1804 \@gls@link[#2]{#3}{#4}%
1805 \let\ifKV@glslink@hyper\glstrorg@ifKV@glslink@hyper
1806 }%
1807 \glspostlinkhook
1808 }

```

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

\@gls@ Save the original definition and redefine.

```

1809 \let\@glstr@org@gls@\@gls@
1810 \def\@gls@#1#2{%
1811 \@glstr@record{#1}{#2}{glslink}%
1812 \@glstr@org@gls@{#1}{#2}%
1813 }%

```

\@glspl@ Save the original definition and redefine.

```

1814 \let\@glstr@org@glspl@\@glspl@
1815 \def\@glspl@#1#2{%
1816 \@glstr@record{#1}{#2}{glslink}%
1817 \@glstr@org@glspl@{#1}{#2}%
1818 }%

```

\@Gls@ Save the original definition and redefine.

```

1819 \let\@glstr@org@Gls@\@Gls@
1820 \def\@Gls@#1#2{%
1821 \@glstr@record{#1}{#2}{glslink}%
1822 \@glstr@org@Gls@{#1}{#2}%
1823 }%

```

\@Glspl@ Save the original definition and redefine.

```

1824 \let\@glstr@org@Glspl@\@Glspl@
1825 \def\@Glspl@#1#2{%
1826 \@glstr@record{#1}{#2}{glslink}%
1827 \@glstr@org@Glspl@{#1}{#2}%
1828 }%

```

\@GLS@ Save the original definition and redefine.

```

1829 \let\@glstr@org@GLS@\@GLS@
1830 \def\@GLS@#1#2{%
1831 \@glstr@record{#1}{#2}{glslink}%

```

```

1832 \@glstr@org@GLS@{#1}{#2}%
1833 }%

```

\@GLSpl@ Save the original definition and redefine.

```

1834 \let\@glstr@org@GLSpl@\@GLSpl@
1835 \def\@GLSpl@#1#2{%
1836   \@glstr@record{#1}{#2}{glslink}%
1837   \@glstr@org@GLSpl@{#1}{#2}%
1838 }%

```

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

```

1839 \renewcommand*{\@glsdisp}[3][{}]{%
1840   \@glstr@record{#1}{#2}{glslink}%
1841   \glsdoifexists{#2}{%
1842     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1843     \let\glsifplural\@secondoftwo
1844     \let\glscapscase\@firstofthree
1845     \def\glscustomtext{#3}%
1846     \def\glsinsert{}%
1847     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1848     \@gl@link[#1]{#2}{\@glo@text}%
1849     \ifKV@glslink@local
1850       \glslocalunset{#2}%
1851     \else
1852       \glsunset{#2}%
1853     \fi
1854   }%
1855   \glspostlinkhook
1856 }

```

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

```

1857 \renewcommand*{\@gls@link}[3][{}]{%
1858   \@glstr@record{#1}{#2}{glslink}%
1859   \glsdoifexistsordo{#2}%
1860   {%
1861     \let\do@gl@link@checkfirsthyper\relax
1862     \@gl@link[#1]{#2}{#3}%
1863   }%
1864   {%
1865     \glstextformat{#3}%
1866   }%
1867   \glspostlinkhook
1868 }

```

sxtrinitwrgloss Set the default if the wrgloss is omitted.

```

1869 \newcommand*{\glstrinitwrgloss}{%
1870   \glsifattribute{\glslabel}{wrgloss}{after}%
1871   {%

```

```

1872 \glstrinitwrglossbeforefalse
1873 }%
1874 {%
1875 \glstrinitwrglossbeforetrue
1876 }%
1877 }

```

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

```

1878 \newif\ifglstrinitwrglossbefore
1879 \glstrinitwrglossbeforetrue

```

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

```

1880 \define@choicekey{glslink}{wrgloss}%
1881 [\@glstr@wrglossval\@glstr@wrglossnr]%
1882 {before,after}%
1883 {%
1884 \ifcase\@glstr@wrglossnr\relax
1885 \glstrinitwrglossbeforetrue
1886 \or
1887 \glstrinitwrglossbeforefalse
1888 \fi
1889 }

```

```

1890 \define@key{glslink}{thevalue}{\def\@glstr@thevalue{#1}}

```

```

1891 \define@key{glslink}{theHvalue}{\def\@glstr@theHvalue{#1}}

```

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

```

1892 \define@boolkey{glslink}[glstr@]{hyperoutside}[true]{}
1893 \glstr@hyperoutsidettrue

```

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

```

1894 \define@key{glslink}{textformat}{%
1895 \ifcsdef{#1}
1896 {%
1897 \letcs{\@glstr@local@textformat}{#1}%
1898 }%
1899 {%
1900 \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
1901 }%
1902 }

```

```

1903 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

```

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

```

1904 \newcommand*{\glstrnithyperoutside}{%
1905 \glsifattribute{\glslabel}{hyperoutside}{false}%

```

```

1906 {%
1907   \glstr@hyperoutsidefalse
1908 }%
1909 {%
1910   \glstr@hyperoutsidetrue
1911 }%
1912 }

```

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

```
1913 \newcommand*{\glstr@inc@linkcount}{}

```

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

```
1914 \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.

```

1915 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
1916   \protected@edef\@glstr@tmp{#2}%
1917   \expandafter#1\expandafter{\@glstr@tmp}%
1918 }

```

`\@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`.

```

1919 \def\@gls@link[#1]#2#3{%
1920   \leavevmode
1921   \edef\glslabel{\glsdetoklabel{#2}}%
1922   \def\@gls@link@opts{#1}%
1923   \let\@gls@link@label\glslabel
1924   \let\@glsnumberformat\glstr@defaultnumberformat
1925   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
1926   \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
1927   \let\@org@ifKV@glslink@hyper@ifKV@glslink@hyper

```

Save current value of `\gls@linkprefix`:

```
1928 \let\@glstr@org@gls@linkprefix\gls@linkprefix

```

Initialise `\@glstr@local@textformat`

```
1929 \let\@glstr@local@textformat\relax

```

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

```

1930 \def\@glstr@thevalue{%
1931 \def\@glstr@theHvalue{\@glstr@thevalue}%

```

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

```
1932 \glstr@initwrgloss

```

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

```
1933 \glstr@inithyperoutside

```

Note that the default link options may override \glstrinitwrgloss.

```
1934 \@gls@setdefault@glslink@opts
```

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

```
1935 \glstr@inc@linkcount
```

As the original definition.

```
1936 \do@glssdisablehyperinlist
```

```
1937 \do@gls@link@checkfirsthyper
```

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

```
1938 \glslinkpresetkeys
```

Set options.

```
1939 \setkeys{glslink}{#1}%
```

User hook after options are set:

```
1940 \glslinkpostsetkeys
```

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

```
1941 \ifdefempty{\@glstr@thevalue}%
```

```
1942 {%
```

```
1943 \@gls@saveentrycounter
```

```
1944 }%
```

```
1945 {%
```

```
1946 \let\theglssentrycounter\@glstr@thevalue
```

```
1947 \def\theHglssentrycounter{\@glstr@theHvalue}%
```

```
1948 }%
```

```
1949 \@gls@setsort{\glslabel}%
```

Check if the textformat key has been used.

```
1950 \ifx\@glstr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
1951 \glshasattribute{\glslabel}{textformat}%
```

```
1952 {%
```

```
1953 \edef\@glstr@attrval{\glssgetattribute{\glslabel}{textformat}}%
```

```
1954 \ifcsdef{\@glstr@attrval}%
```

```
1955 {%
```

```
1956 \letcs{\@glstr@textformat}{\@glstr@attrval}%
```

```
1957 }%
```

```
1958 {%
```

```
1959 \GlossariesExtraWarning{Unknown control sequence name
```

```
1960 '\@glstr@attrval' supplied in textformat attribute
```

```
1961 for entry '\glslabel'. Reverting to default \string\glstextformat}%
```

```
1962 \let\@glstr@textformat\glstextformat
```

```
1963 }%
```

```
1964 }%
```

```
1965 {%
```

```
1966 \let\@glstr@textformat\glstextformat
```

```
1967 }%
```

```
1968 \else
```

```

1969 \let\@glxstr@textformat\@glxstr@local@textformat
1970 \fi

```

Do write if it should occur before the link text:

```

1971 \ifglxstr@nitrwrglossbefore
1972 \do@wrglossary{#2}%
1973 \fi

```

Do the link text:

```

1974 \ifKV@glslink@hyper
1975 \ifglxstr@hyperoutside
1976 \glslink{\glolinkprefix\glslabel}{\@glxstr@textformat{#3}}%
1977 \else
1978 \glxstr@textformat{\glslink{\glolinkprefix\glslabel}{#3}}%
1979 \fi
1980 \else
1981 \ifglxstr@hyperoutside
1982 \glsdonohyperlink{\glolinkprefix\glslabel}{\@glxstr@textformat{#3}}%
1983 \else
1984 \glxstr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
1985 \fi
1986 \fi

```

Do write if it should occur after the link text:

```

1987 \ifglxstr@nitrwrglossbefore
1988 \else
1989 \do@wrglossary{#2}%
1990 \fi

```

Restore original value of \glolinkprefix:

```

1991 \let\glolinkprefix\@glxstr@org@glolinkprefix

```

As the original definition:

```

1992 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
1993 }

```

```

1994 \define@key{glossadd}{thevalue}{\def\@glxstr@thevalue{#1}}

```

```

1995 \define@key{glossadd}{theHvalue}{\def\@glxstr@theHvalue{#1}}

```

lsaddpresetkeys

```

1996 \newcommand*{\glsaddpresetkeys}{}

```

saddpostsetkeys

```

1997 \newcommand*{\glsaddpostsetkeys}{}

```

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

```

1998 \renewrobustcmd*{\glsadd}[2][\]{%
1999 \glxstr@finmark
2000 }%
2001 {%

```



```

2002 \gls@adjustmode
2003 \glsxtr@record{#1}{#2}{glossadd}%
2004 \glsdoifexists{#2}%
2005 {%
2006 \let\glsnumberformat\glsxtr@defaultnumberformat
2007 \edef\gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
2008 \def\glsxtr@thevalue{}%
2009 \def\glsxtr@theHvalue{\glsxtr@thevalue}%

```

Implement any default settings (before options are set)

```

2010 \glsaddpresetkeys
2011 \setkeys{glossadd}{#1}%

```

Implement any default settings (after options are set)

```

2012 \glsaddpostsetkeys
2013 \ifdefempty{\glsxtr@thevalue}%
2014 {%
2015 \gls@saveentrycounter
2016 }%
2017 {%
2018 \let\theglsentrycounter\glsxtr@thevalue
2019 \def\theHglentrycounter{\glsxtr@theHvalue}%
2020 }%

```

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

```

2021 \gls@setsort{#2}%
2022 \@@do@wrglossary{#2}%
2023 }%
2024 }%
2025 }

```

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

```

2026 \newrobustcmd{\glsaddeach}[2][{}]{%
2027 \@for\gls@thislabel:=#2\do{\glsadd[#1]{\gls@thislabel}}%
2028 }

```

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

```

2029 \newcommand*{\glsxtr@field@linkdefs}{%
2030 \let\glsxtrifwasfirstuse\@secondoftwo
2031 \let\glsifplural\@secondoftwo
2032 \let\glsupcase\@firstofthree
2033 \let\glsinsert\@empty
2034 }

```

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

assignfieldfont

```

2035 \newcommand*{\glsxtrassignfieldfont}[1]{%
2036 \ifglentryexists{#1}%

```

```

2037 {%
2038   \ifglshasshort{#1}%
2039   {%
2040     \glsetabbrvfmt{\glscategory{#1}}%
2041     \glusifregular{#1}%
2042     {\let\@gls@field@font\glxtrregularfont}%
2043     {\let\@gls@field@font\@firstofone}%
2044   }%
2045   {%
2046     \glusifnotregular{#1}%
2047     {\let\@gls@field@font\@firstofone}%
2048     {\let\@gls@field@font\glxtrregularfont}%
2049   }%
2050 }%
2051 {%
2052   \let\@gls@field@font\@gobble
2053 }%
2054 }

```

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

```

2055 \def\@glstext@#1#2[#3]{%
2056   \glxtrassignfieldfont{#2}%
2057   \@gls@field@link{#1}{#2}{\@gls@field@font{\glssaccestext{#2}#3}}%
2058 }

```

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

```

2059 \def\@GLStext@#1#2[#3]{%
2060   \glxtrassignfieldfont{#2}%
2061   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2062   {\@gls@field@font{\@GLSaccestext{#2}\mfirstucMakeUppercase{#3}}}%
2063 }

```

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

```

2064 \def\@Glstext@#1#2[#3]{%
2065   \glxtrassignfieldfont{#2}%
2066   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2067   {\@gls@field@font{\@Glsaccestext{#2}#3}}%
2068 }

```

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`

```

2069 \newcommand*\glxtrchecknohyperfirst}[1]{%
2070   \glusifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2071 }

```

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

```

2072 \def\@glsfirst@#1#2[#3]{%
2073   \glxtrassignfieldfont{#2}%

```

Ensure that \glsfirst honours the nohyperfirst attribute.

```
2074 \@gls@field@link
2075 [\let\glsxtrifwasfirstuse\@firstoftwo
2076 \glsxtrchecknohyperfirst{#2}%
2077 ]{#1}{#2}%
2078 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2079 }
```

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

```
2080 \def\@Glsfirst@#1#2[#3]{%
2081 \glsxtrassignfieldfont{#2}%
    Ensure that \Glsfirst honours the nohyperfirst attribute.
2082 \@gls@field@link
2083 [\let\glsxtrifwasfirstuse\@firstoftwo
2084 \let\glscapscase\@secondofthree
2085 \glsxtrchecknohyperfirst{#2}%
2086 ]%
2087 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2088 }
```

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

```
2089 \def\@GLSfirst@#1#2[#3]{%
2090 \glsxtrassignfieldfont{#2}%
    Ensure that \GLSfirst honours the nohyperfirst attribute.
2091 \@gls@field@link
2092 [\let\glsxtrifwasfirstuse\@firstoftwo
2093 \let\glscapscase\@thirdofthree
2094 \glsxtrchecknohyperfirst{#2}%
2095 ]%
2096 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2097 }
```

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

```
2098 \def\@glsplural@#1#2[#3]{%
2099 \glsxtrassignfieldfont{#2}%
2100 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2101 {\@gls@field@font{\glsaccessplural{#2}#3}}%
2102 }
```

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

```
2103 \def\@Glsplural@#1#2[#3]{%
2104 \glsxtrassignfieldfont{#2}%
2105 \@gls@field@link
2106 [\let\glsifplural\@firstoftwo
2107 \let\glscapscase\@secondofthree
2108 ]%
2109 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2110 }
```

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

```

2111 \def\@GLSplural@#1#2[#3]{%
2112   \glstrassignfieldfont{#2}%
2113   \@gls@field@link
2114   [\let\glsifplural\@firstoftwo
2115   \let\glscapscase\@thirdofthree
2116   ]%
2117   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2118 }
```

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

```

2119 \def\@glsfirstplural@#1#2[#3]{%
2120   \glstrassignfieldfont{#2}%
2121   \@gls@field@link
2122   [\let\glstrifwasfirstuse\@firstoftwo
2123   \let\glsifplural\@firstoftwo
2124   \glstrchecknohyperfirst{#2}%
2125   ]%
2126   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2127 }
```

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

```

2128 \def\@Glsfirstplural@#1#2[#3]{%
2129   \glstrassignfieldfont{#2}%
2130   \@gls@field@link
2131   [\let\glstrifwasfirstuse\@firstoftwo
2132   \let\glsifplural\@firstoftwo
2133   \let\glscapscase\@secondofthree
2134   \glstrchecknohyperfirst{#2}%
2135   ]%
2136   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2137 }
```

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

```

2138 \def\@GLSfirstplural@#1#2[#3]{%
2139   \glstrassignfieldfont{#2}%
2140   \@gls@field@link
2141   [\let\glstrifwasfirstuse\@firstoftwo
2142   \let\glsifplural\@firstoftwo
2143   \let\glscapscase\@thirdofthree
2144   \glstrchecknohyperfirst{#2}%
2145   ]%
2146   {#1}{#2}%
2147   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2148 }
```

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

```
2149 \def\@glsname@#1#2[#3]{%
2150   \glstrassignfieldfont{#2}%
2151   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
2152 }
```

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

```
2153 \def\@Glsname@#1#2[#3]{%
2154   \glstrassignfieldfont{#2}%
2155   \@gls@field@link
2156   [\let\glscaps@case\@secondoftwo]{#1}{#2}%
2157   {\@gls@field@font{\Glsaccessname{#2}#3}}%
2158 }
```

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

```
2159 \def\@GLSname@#1#2[#3]{%
2160   \glstrassignfieldfont{#2}%
2161   \@gls@field@link[\let\glscaps@case\@thirdoftwo]%
2162   {#1}{#2}%
2163   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2164 }
```

\@glsdesc@

```
2165 \def\@glsdesc@#1#2[#3]{%
2166   \glstrassignfieldfont{#2}%
2167   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
2168 }
```

\@Glsdesc@ First letter uppercase version.

```
2169 \def\@Glsdesc@#1#2[#3]{%
2170   \glstrassignfieldfont{#2}%
2171   \@gls@field@link
2172   [\let\glscaps@case\@secondoftwo]{#1}{#2}%
2173   {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
2174 }
```

\@GLSdesc@ All uppercase version.

```
2175 \def\@GLSdesc@#1#2[#3]{%
2176   \glstrassignfieldfont{#2}%
2177   \@gls@field@link[\let\glscaps@case\@thirdoftwo]%
2178   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2179 }
```

@glsdescplural@ No case-changing version.

```
2180 \def\@glsdescplural@#1#2[#3]{%
2181   \glstrassignfieldfont{#2}%
2182   \@gls@field@link
2183   [\let\glscaps@case\@secondoftwo
```

```

2184 \let\glsifplural\@firstoftwo
2185 ]{\#1}{\#2}{\@gls@field@font{\glsaccessdescplural{\#2}{\#3}}}%
2186 }

```

@Glsdescplural@ First letter uppercase version.

```

2187 \def\@Glsdescplural@#1#2[#3]{%
2188 \glsxtrassignfieldfont{\#2}%
2189 \@gls@field@link
2190 [\let\glscapscase\@secondoftwo
2191 \let\glsifplural\@firstoftwo
2192 ]{\#1}{\#2}{\@gls@field@font{\Glsaccessdescplural{\#2}{\#3}}}%
2193 }

```

@GLSdescplural@ All uppercase version.

```

2194 \def\@GLSdesc@#1#2[#3]{%
2195 \glsxtrassignfieldfont{\#2}%
2196 \@gls@field@link
2197 [\let\glscapscase\@thirdoftwo
2198 \let\glsifplural\@firstoftwo
2199 ]%
2200 {\#1}{\#2}%
2201 {\@gls@field@font{\GLSaccessdescplural{\#2}\mfirstucMakeUppercase{\#3}}}%
2202 }

```

\@glssymbol@

```

2203 \def\@glssymbol@#1#2[#3]{%
2204 \glsxtrassignfieldfont{\#2}%
2205 \@gls@field@link{\#1}{\#2}{\@gls@field@font{\glsaccesssymbol{\#2}{\#3}}}%
2206 }

```

\@Glsymbol@ First letter uppercase version.

```

2207 \def\@Glsymbol@#1#2[#3]{%
2208 \glsxtrassignfieldfont{\#2}%
2209 \@gls@field@link
2210 [\let\glscapscase\@secondoftwo]%
2211 {\#1}{\#2}{\@gls@field@font{\Glsaccesssymbol{\#2}{\#3}}}%
2212 }

```

\@GLSsymbol@ All uppercase version.

```

2213 \def\@GLSsymbol@#1#2[#3]{%
2214 \glsxtrassignfieldfont{\#2}%
2215 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2216 {\#1}{\#2}{\@gls@field@font{\GLSaccesssymbol{\#2}\mfirstucMakeUppercase{\#3}}}%
2217 }

```

lssymbolplural@ No case-changing version.

```

2218 \def\@glssymbolplural@#1#2[#3]{%
2219 \glsxtrassignfieldfont{\#2}%

```

```

2220 \@gls@field@link
2221 [\let\glscapscase\@secondoftwo
2222 \let\glsifplural\@firstoftwo
2223 ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2224 }

```

lssymbolplural@ First letter uppercase version.

```

2225 \def\@GLssymbolplural@#1#2[#3]{%
2226 \glstrassignfieldfont{#2}%
2227 \@gls@field@link
2228 [\let\glscapscase\@secondoftwo
2229 \let\glsifplural\@firstoftwo
2230 ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2231 }

```

LSsymbolplural@ All uppercase version.

```

2232 \def\@GLSsymbol@#1#2[#3]{%
2233 \glstrassignfieldfont{#2}%
2234 \@gls@field@link
2235 [\let\glscapscase\@thirdoftwo
2236 \let\glsifplural\@firstoftwo
2237 ]%
2238 {#1}{#2}%
2239 {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2240 }

```

\@Glsuseri@ First letter uppercase version.

```

2241 \def\@Glsuseri@#1#2[#3]{%
2242 \glstrassignfieldfont{#2}%
2243 \@gls@field@link
2244 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2245 {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2246 }

```

\@GLSuseri@ All uppercase version.

```

2247 \def\@GLSuseri@#1#2[#3]{%
2248 \glstrassignfieldfont{#2}%
2249 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2250 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
2251 }

```

\@Glsuserii@ First letter uppercase version.

```

2252 \def\@Glsuserii@#1#2[#3]{%
2253 \glstrassignfieldfont{#2}%
2254 \@gls@field@link
2255 [\let\glscapscase\@secondoftwo]%
2256 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2257 }

```

\@GLSuserii@ All uppercase version.

```
2258 \def\@GLSuserii@#1#2[#3]{%
2259   \glstrassignfieldfont{#2}%
2260   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2261   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2262 }
```

\@GLSuseriii@ First letter uppercase version.

```
2263 \def\@GLSuseriii@#1#2[#3]{%
2264   \glstrassignfieldfont{#2}%
2265   \@gls@field@link
2266   [\let\glscapscase\@secondoftwo]%
2267   {#1}{#2}{\@gls@field@font{\Glentryuseriii{#2}#3}}}%
2268 }
```

\@GLSuseriii@ All uppercase version.

```
2269 \def\@GLSuseriii@#1#2[#3]{%
2270   \glstrassignfieldfont{#2}%
2271   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2272   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2273 }
```

\@GLSuseriv@ First letter uppercase version.

```
2274 \def\@GLSuseriv@#1#2[#3]{%
2275   \glstrassignfieldfont{#2}%
2276   \@gls@field@link
2277   [\let\glscapscase\@secondoftwo]%
2278   {#1}{#2}{\@gls@field@font{\Glentryuseriv{#2}#3}}}%
2279 }
```

\@GLSuseriv@ All uppercase version.

```
2280 \def\@GLSuseriv@#1#2[#3]{%
2281   \glstrassignfieldfont{#2}%
2282   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2283   {#1}{#2}%
2284   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2285 }
```

\@GLSuserv@ First letter uppercase version.

```
2286 \def\@GLSuserv@#1#2[#3]{%
2287   \glstrassignfieldfont{#2}%
2288   \@gls@field@link
2289   [\let\glscapscase\@secondoftwo]%
2290   {#1}{#2}{\@gls@field@font{\Glentryuserv{#2}#3}}}%
2291 }
```

\@GLSuserv@ All uppercase version.

```
2292 \def\@GLSuserv@#1#2[#3]{%
```



```

2293 \glstrassignfieldfont{#2}%
2294 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2295 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2296 }

```

\@Glsuservi@ First letter uppercase version.

```

2297 \def\@Glsuservi@#1#2[#3]{%
2298 \glstrassignfieldfont{#2}%
2299 \@gls@field@link
2300 [\let\glscapscase\@secondoftwo]%
2301 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2302 }

```

\@GLSuservi@ All uppercase version.

```

2303 \def\@GLSuservi@#1#2[#3]{%
2304 \glstrassignfieldfont{#2}%
2305 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2306 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2307 }

```

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

\@acrshort No case change.

```

2308 \def\@acrshort#1#2[#3]{%
2309 \glsdoifexists{#2}%
2310 {%
2311 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2312 \let\glxtrifwasfirstuse\@secondoftwo
2313 \let\glsifplural\@secondoftwo
2314 \let\glscapscase\@firstofthree
2315 \let\glsinsert\@empty
2316 \def\glscustomtext{%
2317 \acronymfont{\glsaccesssshort{#2}}#3%
2318 }%
2319 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2320 }%
2321 \glspostlinkhook
2322 }

```

\@Acrshort First letter uppercase.

```

2323 \def\@Acrshort#1#2[#3]{%
2324 \glsdoifexists{#2}%
2325 {%
2326 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2327 \let\glxtrifwasfirstuse\@secondoftwo
2328 \let\glsifplural\@secondoftwo
2329 \let\glscapscase\@secondofthree
2330 \let\glsinsert\@empty

```

```

2331 \def\glscustomtext{%
2332 \acronymfont{\Glsaccesssshort{#2}}#3%
2333 }%
2334 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2335 }%
2336 \glspostlinkhook
2337 }

```

\@ACRshort All uppercase.

```

2338 \def\@ACRshort#1#2[#3]{%
2339 \glsdoidexists{#2}%
2340 {%
2341 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2342 \let\glstrifwasfirstuse\@secondoftwo
2343 \let\glsifplural\@secondoftwo
2344 \let\glscapscase\@thirdofthree
2345 \let\glsinsert\@empty
2346 \def\glscustomtext{%
2347 \mfirstucMakeUppercase{\acronymfont{\Glsaccesssshort{#2}}#3}%
2348 }%
2349 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2350 }%
2351 \glspostlinkhook
2352 }

```

\@acrshortpl No case change.

```

2353 \def\@acrshortpl#1#2[#3]{%
2354 \glsdoidexists{#2}%
2355 {%
2356 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2357 \let\glstrifwasfirstuse\@secondoftwo
2358 \let\glsifplural\@firstoftwo
2359 \let\glscapscase\@firstofthree
2360 \let\glsinsert\@empty
2361 \def\glscustomtext{%
2362 \acronymfont{\Glsaccesssshortpl{#2}}#3%
2363 }%
2364 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2365 }%
2366 \glspostlinkhook
2367 }

```

\@Acrshortpl First letter uppercase.

```

2368 \def\@Acrshortpl#1#2[#3]{%
2369 \glsdoidexists{#2}%
2370 {%
2371 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2372 \let\glstrifwasfirstuse\@secondoftwo
2373 \let\glsifplural\@firstoftwo

```

```

2374 \let\glscapscase\@secondofthree
2375 \let\glsinsert\@empty
2376 \def\glscustomtext{%
2377     \acronymfont{\Glsaccessshortpl{#2}}#3%
2378 }%
2379 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2380 }%
2381 \glspostlinkhook
2382 }

```

\@ACRshortpl All uppercase.

```

2383 \def\@ACRshortpl#1#2[#3]{%
2384     \glsdoifexists{#2}%
2385     {%
2386         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2387         \let\glstrifwasfirstuse\@secondoftwo
2388         \let\glsifplural\@firstoftwo
2389         \let\glscapscase\@thirdofthree
2390         \let\glsinsert\@empty
2391         \def\glscustomtext{%
2392             \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
2393         }%
2394         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2395     }%
2396     \glspostlinkhook
2397 }

```

\@acrlong No case change.

```

2398 \def\@acrlong#1#2[#3]{%
2399     \glsdoifexists{#2}%
2400     {%
2401         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2402         \let\glstrifwasfirstuse\@secondoftwo
2403         \let\glsifplural\@secondoftwo
2404         \let\glscapscase\@firstofthree
2405         \let\glsinsert\@empty
2406         \def\glscustomtext{%
2407             \acronymfont{\Glsaccesslong{#2}}#3%
2408         }%
2409         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2410     }%
2411     \glspostlinkhook
2412 }

```

\@Acrlong First letter uppercase.

```

2413 \def\@Acrlong#1#2[#3]{%
2414     \glsdoifexists{#2}%
2415     {%
2416         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2417 \let\glxtrifwasfirstuse\@secondoftwo
2418 \let\glsifplural\@secondoftwo
2419 \let\glscapscase\@secondofthree
2420 \let\glsinsert\@empty
2421 \def\glscustomtext{%
2422   \acronymfont{\Glsaccesslong{#2}}#3%
2423 }%
2424 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2425 }%
2426 \glspostlinkhook
2427 }

```

\@ACRlong All uppercase.

```

2428 \def\@ACRlong#1#2[#3]{%
2429   \glsdoifexists{#2}%
2430   {%
2431     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2432     \let\glxtrifwasfirstuse\@secondoftwo
2433     \let\glsifplural\@secondoftwo
2434     \let\glscapscase\@thirdofthree
2435     \let\glsinsert\@empty
2436     \def\glscustomtext{%
2437       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
2438     }%
2439     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2440   }%
2441   \glspostlinkhook
2442 }

```

\@acrlongpl No case change.

```

2443 \def\@acrlongpl#1#2[#3]{%
2444   \glsdoifexists{#2}%
2445   {%
2446     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2447     \let\glxtrifwasfirstuse\@secondoftwo
2448     \let\glsifplural\@firstoftwo
2449     \let\glscapscase\@firstofthree
2450     \let\glsinsert\@empty
2451     \def\glscustomtext{%
2452       \acronymfont{\Glsaccesslongpl{#2}}#3%
2453     }%
2454     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2455   }%
2456   \glspostlinkhook
2457 }

```

\@Acrlongpl First letter uppercase.

```

2458 \def\@Acrlongpl#1#2[#3]{%
2459   \glsdoifexists{#2}%

```

```

2460 {%
2461   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2462   \let\glxtrifwasfirstuse\@secondoftwo
2463   \let\gl@sifplural\@firstoftwo
2464   \let\gl@scapscase\@secondofthree
2465   \let\gl@insert\@empty
2466   \def\gl@customtext{%
2467     \acronymfont{\Glsaccesslongpl{#2}}#3%
2468   }%
2469   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2470 }%
2471 \glspostlinkhook
2472 }

```

\@ACRlongpl All uppercase.

```

2473 \def\@ACRlongpl#1#2[#3]{%
2474   \gl@dofexists{#2}%
2475   {%
2476     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2477     \let\glxtrifwasfirstuse\@secondoftwo
2478     \let\gl@sifplural\@firstoftwo
2479     \let\gl@scapscase\@thirdofthree
2480     \let\gl@insert\@empty
2481     \def\gl@customtext{%
2482       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
2483     }%
2484     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2485   }%
2486   \glspostlinkhook
2487 }

```

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

\@glsaddkey

```

2488 \renewcommand*{\@glsaddkey}[7]{%
2489   \key@ifundefined{glossentry}{#1}%
2490   {%
2491     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2492     \appto\@gl@keymap{, {#1}{#1}}%
2493     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2494     \appto\@newglossaryentryposthook{%
2495       \letcs{\@glo@tmp}{@glo@#1}%
2496       \gl@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2497     }%
2498     \newcommand*{#3}[1]{\@gl@entry@field{##1}{#1}}%
2499     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2500   \ifcsdef{@gl@user@#1@}%

```

```

2501 {%
2502   \PackageError{glossaries}%
2503   {Can't define '\string#5' as helper command
2504     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2505     exists}%
2506   }%
2507 }%
2508 {%
2509   \expandafter\newcommand\expandafter*\expandafter
2510   {\csname @gls@user@#1@\endcsname}[2][\%
2511     \new@ifnextchar[%
2512       {\csuse{@gls@user@#1@}{##1}{##2}}%
2513       {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2514   \csdef{@gls@user@#1@}##1##2[##3]{%
2515     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2516   }%
2517   \newrobustcmd*{#5}{%
2518     \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
2519   }%

```

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

```

2520 \ifcsdef{@Gls@user@#1@}%
2521 {%
2522   \PackageError{glossaries}%
2523   {Can't define '\string#6' as helper command
2524     '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2525     exists}%
2526   }%
2527 }%
2528 {%
2529   \expandafter\newcommand\expandafter*\expandafter
2530   {\csname @Gls@user@#1@\endcsname}[2][\%
2531     \new@ifnextchar[%
2532       {\csuse{@Gls@user@#1@}{##1}{##2}}%
2533       {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2534   \csdef{@Gls@user@#1@}##1##2[##3]{%
2535     \@gls@field@link[\let\glscaps@case\@secondofthree]%
2536     {##1}{##2}{#4{##2}##3}%
2537   }%
2538   \newrobustcmd*{#6}{%
2539     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
2540   }%

```

Finally the all caps version (modified):

```

2541 \ifcsdef{@GLS@user@#1@}%
2542 {%
2543   \PackageError{glossaries}%
2544   {Can't define '\string#7' as helper command
2545     '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2546     exists}%

```

```

2547     {}%
2548 }%
2549 {%
2550     \expandafter\newcommand\expandafter*\expandafter
2551     {\csname @GLS@user@#1\endcsname}[2][{}]{%
2552         \new@ifnextchar[%
2553             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2554             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2555     \csdef{@GLS@user@#1@}##1##2[##3]{%
2556         \@gls@field@link[\let\gls@caps@case\@thirdofthree]%
2557         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2558     }%
2559     \newrobustcmd*{#7}{%
2560         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2561     }%
2562 }%
2563 {%
2564     \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
2565 }%
2566 }

```

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

```

2567 \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.

```

2568 \let\@gls@xtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2569 \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.

```

2570 \ifglsused{\glslabel}%
2571 {\let\gls@xtrifwasfirstuse\@secondoftwo}
2572 {\let\gls@xtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2573 \edef\gls@categorylabel{\gls@category{\glslabel}}%
2574 \ifglsused{\glslabel}%
2575 {%
2576     \glsifcategoryattribute{\gls@categorylabel}{nohypernext}{true}%
2577     {\KV@glslink@hyperfalse}}}%
2578 }%
2579 {%
2580     \glsifcategoryattribute{\gls@categorylabel}{nohyperfirst}{true}%
2581     {\KV@glslink@hyperfalse}}}%
2582 }%
2583 \glslinkcheckfirsthyperhook
2584 }

```

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.

```

2585 \ifdef\do@glsglssdisablehyperinlist
2586 {%
2587   \let\@glsglssxtr@do@glsglssdisablehyperinlist\do@glsglssdisablehyperinlist
2588   \renewcommand*{\do@glsglssdisablehyperinlist}{%
2589     \@glsglssxtr@do@glsglssdisablehyperinlist
2590     \glsglssifattribute{\glsglsslabel}{nohyper}{true}{\KV@glsglsslink@hyperfalse}}}%
2591   }
2592 }
2593 {}

```

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

```

2594 \define@boolkey{glsglsslink}{noindex}[true]{}
2595 \KV@glsglsslink@noindexfalse

```

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

lt@glsglsslink@opts

```

2596 \ifdef\@glsglss@setdefault@glsglsslink@opts
2597 {
2598   \renewcommand*{\@glsglss@setdefault@glsglsslink@opts}{%
2599     \KV@glsglsslink@noindexfalse
2600     \@glsglssxtrsetaliasnoindex
2601   }
2602 }
2603 {

```

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

```

2604   \newcommand*{\@glsglss@setdefault@glsglsslink@opts}{%
2605     \KV@glsglsslink@noindexfalse
2606     \@glsglssxtrsetaliasnoindex
2607   }
2608   \preto\do@glsglssdisablehyperinlist{\@glsglss@setdefault@glsglsslink@opts}
2609 }

```

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.)

```

2610 \providecommand*{\glsglssxtrsetaliasnoindex}{%
2611   \KV@glsglsslink@noindextrue
2612 }

```

setaliasnoindex

```

2613 \newcommand*{\@glsglssxtrsetaliasnoindex}{%
2614   \glsglssxtrifhasfield{alias}{\glsglsslabel}%
2615   {%
2616     \let\glsglssxtrindexaliased\@glsglssxtrindexaliased

```



```

2617 \glxtrsetaliasnoindex
2618 \let\glxtrindexaliased\@no@glxtrindexaliased
2619 }%
2620 {}%
2621 }

```

xtrindexaliased

```

2622 \newcommand{\@glxtrindexaliased}{%
2623 \ifKV@glslink@noindex
2624 \else
2625 \begingroup
2626 \let\@glslnumberformat\@glxtr@defaultnumberformat
2627 \edef\@glscounter{\csname glo@\glsdetoklabel{\glslabel}@counter\endcsname}%
2628 \glxtr@saveentrycounter
2629 \@do@wrglossary{\glxtralias{\glslabel}}%
2630 \endgroup
2631 \fi
2632 }

```

xtrindexaliased

```

2633 \newcommand{\@no@glxtrindexaliased}{%
2634 \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2635 not permitted outside definition of \string\glxtrsetaliasnoindex}%
2636 {}%
2637 }

```

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

```

2638 \let\glxtrindexaliased\@no@glxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

2639 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2640 \renewcommand*{\@glscsetdefault@glslink@opts}{%
2641 \setkeys{glslink}{#1}%
2642 \@glxtrsetaliasnoindex
2643 }%
2644 }

```

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

```

2645 \newcommand*{\glxtrifindexing}[2]{%
2646 \ifKV@glslink@noindex #2\else #1\fi
2647 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

2648 \renewcommand*{\glswriteentry}[2]{%
2649 \glxtrifindexing
2650 {%
2651 \ifglindexonlyfirst
2652 \ifglused{#1}

```

```

2653     {\glxtrdoautoindexname{#1}{dualindex}}}%
2654     {#2}}%
2655   \else
2656     \glusifattribute{#1}{indexonlyfirst}{true}%
2657     {\ifglused{#1}
2658      {\glxtrdoautoindexname{#1}{dualindex}}}%
2659      {#2}}}%
2660     {#2}}%
2661   \fi
2662 }%
2663 {}%
2664 }

```

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

```

2665 \appto\@do@@wrglossary{\@glxtr@do@@wrindex
2666  \glxtrdowrglossaryhook{\@gls@label}}%
2667 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2668 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2669  \glxtrdowrglossaryhook{\@gls@label}}%
2670 }

```

`xtr@do@@wrindex`

```

2671 \newcommand*{\@glxtr@do@@wrindex}{%
2672  \glxtrdoautoindexname{\@gls@label}{dualindex}}%
2673 }

```

`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.)

```

2674 \newcommand*{\glxtrdowrglossaryhook}[1]{%

```

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

```

2675 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2676  \let\glslinkvar\@firstofthree
2677  \let\@gls@hyp@opt@cs#1\relax
2678  \@ifstar{\s@gls@hyp@opt}%
2679  {\@ifnextchar+%
2680   {\@firstoftwo{\p@gls@hyp@opt}}}%
2681  {%
2682   \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2683   {\@firstoftwo{\@alt@gls@hyp@opt}}}%
2684   {#1}}%

```

```

2685 }%
2686 }%
2687 }

```

`\alt@gls@hyp@opt` User version

```

2688 \newcommand*{\@alt@gls@hyp@opt}[1] [] {%
2689 \let\glslinkvar\@firstofthree
2690 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}

```

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

```

2691 \newcommand*{\@gls@alt@hyp@opt@char}{}

```

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

```

2692 \newcommand*{\@gls@alt@hyp@opt@keys}{}

```

`\rSetAltModifier`

```

2693 \newcommand*{\GlsXtrSetAltModifier}[2] {%
2694 \let\@gls@hyp@opt\@gls@alt@hyp@opt
2695 \def\@gls@alt@hyp@opt@char{#1}%
2696 \def\@gls@alt@hyp@opt@keys{#2}%
2697 }

```

`\org@dohyperlink`

```

2698 \let\glsxtr@org@dohyperlink\glsdohyperlink

```

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

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

```

2699 \ifdef\glsnavhyperlink
2700 {
2701 \renewcommand*{\glsnavhyperlink}[3] [\@glo@type] {%
2702 \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%

```

Scope:

```

2703 {%
2704 \let\glsdohyperlink\glsxtr@org@dohyperlink
2705 \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
2706 }%
2707 }%
2708 }
2709 {}

```

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

and `\glxtrshort` to use `\glstryshort` and, similarly, the long form commands like `\acrlong` and `\glxtrlong` to use `\glstrylong`. Added attribute check.

```

2710 \renewcommand*{\glsdohyperlink}[2]{%
2711   \glshasattribute{\glslabel}{targeturl}%
2712   {%
2713     \glshasattribute{\glslabel}{targetname}%
2714     {%
2715       \glshasattribute{\glslabel}{targetcategory}%
2716       {%
2717         \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2718           {\glsggetattribute{\glslabel}{targetcategory}}}%
2719         {\glsggetattribute{\glslabel}{targetname}}}%
2720         {\glxtrprotectlinks#2}}%
2721       }%
2722     {%
2723       \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2724         }%
2725       {\glsggetattribute{\glslabel}{targetname}}}%
2726       {\glxtrprotectlinks#2}}%
2727     }%
2728   }%
2729   {%
2730     \href{\glsggetattribute{\glslabel}{targeturl}}{%
2731       {\glxtrprotectlinks#2}}%
2732     }%
2733   }%
2734   {%

```

Check for alias.

```

2735   \glsgfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2736   \ifdefvoid\gloaliaslabel
2737   {%
2738     \glxtrhyperlink{#1}{\glxtrprotectlinks#2}}%
2739   }%
2740   {%

```

Redirect link to the alias target.

```

2741   \glxtrhyperlink
2742   {\glolinkprefix\glsgdetoklabel{\gloaliaslabel}}%
2743   {\glxtrprotectlinks#2}}%
2744   }%
2745 }%
2746 }

```

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

```

2747 \ifdef\@glsshowtarget
2748 {
2749   \newcommand{\glxtrhyperlink}[2]{%
2750     \@glsshowtarget{#1}%
2751     \hyperlink{#1}{#2}%

```

```

2752 }%
2753 }
2754 {
2755   \newcommand{\glxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
2756 }

```

`\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.

```

2757 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
2758   \glsdoifexists{#2}%
2759   {%
2760     \def\@glo@label{#2}%
2761     {\edef\glslabel{#2}%
2762       \@glslink{\glo@linkprefix\glslabel}{#1}}%
2763   }%
2764 }

```

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

```

2765 \renewcommand{\glsdisablehyper}{%
2766   \KV@glslink@hyperfalse
2767   \def\@glslink{\glsdonohyperlink}%
2768   \let\@glstarget\@secondoftwo
2769 }

```

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

```

2770 \renewcommand{\glsenablehyper}{%
2771   \KV@glslink@hypertrue
2772   \def\@glslink{\glsdohyperlink}%
2773   \def\@glstarget{\glsdohypertarget}%
2774 }

```

`\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).

```

2775 \def\glsdonohyperlink#1#2{{\glxtrprotectlinks #2}}

```

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

```

2776 \ifcsundef{hyperlink}%
2777 {%
2778   \def\@glslink{\glsdonohyperlink}
2779 }%
2780 {%

```

```

2781 \def\@glslink{\glsdohyperlink}
2782 }

```

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

```

2783 \newcommand*{\glsxtrprotectlinks}{%
2784 \KV@glslink@hyperfalse
2785 \KV@glslink@noindextrue
2786 \let\@gls@\@glsxtr@p@text@
2787 \let\@Gls@\@Glsxtr@p@text@
2788 \let\@GLS@\@GLSxtr@p@text@
2789 \let\@glspl@\@glsxtr@p@plural@
2790 \let\@Glspl@\@Glsxtr@p@plural@
2791 \let\@GLSpl@\@GLSxtr@p@plural@
2792 \let\@glsxtrshort@\@glsxtr@p@short@
2793 \let\@Glsxtrshort@\@Glsxtr@p@short@
2794 \let\@GLSxtrshort@\@GLSxtr@p@short@
2795 \let\@glsxtrlong@\@glsxtr@p@long@
2796 \let\@Glsxtrlong@\@Glsxtr@p@long@
2797 \let\@GLSxtrlong@\@GLSxtr@p@long@
2798 \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
2799 \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
2800 \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
2801 \let\@glsxtrlongpl@\@glsxtr@p@longpl@
2802 \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
2803 \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
2804 \let\@acrshort@\@glsxtr@p@acrshort@
2805 \let\@Acrshort@\@Glsxtr@p@acrshort@
2806 \let\@ACRshort@\@GLSxtr@p@acrshort@
2807 \let\@acrshortpl@\@glsxtr@p@acrshortpl@
2808 \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
2809 \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
2810 \let\@acrlong@\@glsxtr@p@acrlong@
2811 \let\@Acrlong@\@Glsxtr@p@acrlong@
2812 \let\@ACRlong@\@GLSxtr@p@acrlong@
2813 \let\@acrlongpl@\@glsxtr@p@acrlongpl@
2814 \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
2815 \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
2816 }

```

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

`@glsxtr@p@text@`

```

2817 \def\@glsxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}

```

`@Glsxtr@p@text@`

```

2818 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}

```

`@GLSxtr@p@text@`

```

2819 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}

```

```

lsxtr@p@plural@
2820 \def\@glxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}

lsxtr@p@plural@
2821 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}

LSxtr@p@plural@
2822 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}

glxtr@p@short@
2823 \def\@glxtr@p@short@#1#2[#3]{%
2824 {%
2825   \glsssetabbrvfmt{\glscategory{#2}}}%
2826   \glsabbrvfont{\glsentryshort{#2}}#3%
2827 }%
2828 }

Glsxtr@p@short@
2829 \def\@Glsxtr@p@short@#1#2[#3]{%
2830 {%
2831   \glsssetabbrvfmt{\glscategory{#2}}}%
2832   \glsabbrvfont{\Glsentryshort{#2}}#3%
2833 }%
2834 }

GLSxtr@p@short@
2835 \def\@GLSxtr@p@short@#1#2[#3]{%
2836 {%
2837   \glsssetabbrvfmt{\glscategory{#2}}}%
2838   \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
2839 }%
2840 }

sxtr@p@shortpl@
2841 \def\@glxtr@p@shortpl@#1#2[#3]{%
2842 {%
2843   \glsssetabbrvfmt{\glscategory{#2}}}%
2844   \glsabbrvfont{\glsentryshortpl{#2}}#3%
2845 }%
2846 }

sxtr@p@shortpl@
2847 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2848 {%
2849   \glsssetabbrvfmt{\glscategory{#2}}}%
2850   \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2851 }%
2852 }

```

Sxtr@p@shortpl@

```

2853 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2854   {%
2855     \glsetabbrvfmt{\glscategory{#2}}%
2856     \mfirstucMakeUppercase{\glsabrvfont{\glentryshortpl{#2}}#3}%
2857   }%
2858 }

```

@glxtr@p@long@

```

2859 \def\@glxtr@p@long@#1#2[#3]{\glentrylong{#2}#3}

```

@Glsxtr@p@long@

```

2860 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

```

@GLSxtr@p@long@

```

2861 \def\@GLSxtr@p@long@#1#2[#3]{%
2862   {\mfirstucMakeUppercase{\glslongfont{\glentrylong{#2}}#3}}

```

lsxtr@p@longpl@

```

2863 \def\@glxtr@p@longpl@#1#2[#3]{\glentrylongpl{#2}#3}

```

lSxtr@p@longpl@

```

2864 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsentrylongpl{#2}}#3}

```

LSxtr@p@longpl@

```

2865 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2866   {\mfirstucMakeUppercase{\glslongfont{\glentrylongpl{#2}}#3}}

```

xtr@p@acrshort@

```

2867 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\glentryshort{#2}}#3}

```

xtr@p@acrshort@

```

2868 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

```

xtr@p@acrshort@

```

2869 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2870   {\mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}}#3}}

```

r@p@acrshortpl@

```

2871 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glentryshortpl{#2}}#3}

```

r@p@acrshortpl@

```

2872 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}

```

r@p@acrshortpl@

```

2873 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2874   {\mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}}#3}}

```



```

sctr@p@acrlong@
2875 \def\@glxstr@p@acrlong@#1#2[#3]{\@glentrylong{#2}#3}}

sctr@p@acrlong@
2876 \def\@Glsxtr@p@acrlong@#1#2[#3]{\@Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
2877 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2878 {\mfirstucMakeUppercase{\@glentrylong{#2}#3}}}}

tr@p@acrlongpl@
2879 \def\@glxstr@p@acrlongpl@#1#2[#3]{\@glentrylongpl{#2}#3}}

tr@p@acrlongpl@
2880 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\@Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
2881 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2882 {\mfirstucMakeUppercase{\@glentrylongpl{#2}#3}}}}

```

Commands to minimise conflict.

```

\@glxstrp@opt
2883 \newcommand*{\@glxstrp@opt}{hyper=false,noindex}

\glxstrsetpopts  Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.
2884 \newcommand*{\glxstrsetpopts}[1]{%
2885   \renewcommand*{\@glxstrp@opt}{#1}%
2886 }

lossxtrsetpopts  Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.
2887 \newcommand*{\glossxtrsetpopts}{%
2888   \glxstrsetpopts{noindex}%
2889 }

\@@glxstrp
2890 \newrobustcmd*{\@@glxstrp}[2]{%
  Add scope.
2891   {%
2892     \let\glspostlinkhook\relax
2893     \csname#1\expandafter\endcsname\expandafter[\@glxstrp@opt]{#2}[]%
2894   }%
2895 }

```

\@glsxtrp

```
2896 \newrobustcmd*{\@glsxtrp}[2]{%
2897   \ifcsdef{gls#1}%
2898   {%
2899     \@glsxtrp{gls#1}{#2}%
2900   }%
2901   {%
2902     \ifcsdef{glsxtr#1}%
2903     {%
2904       \@glsxtrp{glsxtr#1}{#2}%
2905     }%
2906     {%
2907       \PackageError{glossaries-extra}{‘#1’ not recognised by
2908         \string\glsxtrp}{}%
2909     }%
2910   }%
2911 }
```

\@Glsxtrp

```
2912 \newrobustcmd*{\@Glsxtrp}[2]{%
2913   \ifcsdef{Gls#1}%
2914   {%
2915     \@glsxtrp{Gls#1}{#2}%
2916   }%
2917   {%
2918     \ifcsdef{Glsxtr#1}%
2919     {%
2920       \@glsxtrp{Glsxtr#1}{#2}%
2921     }%
2922     {%
2923       \PackageError{glossaries-extra}{‘#1’ not recognised by
2924         \string\Glsxtrp}{}%
2925     }%
2926   }%
2927 }
```

\@GLSxtrp

```
2928 \newrobustcmd*{\@GLSxtrp}[2]{%
2929   \ifcsdef{GLS#1}%
2930   {%
2931     \@glsxtrp{GLS#1}{#2}%
2932   }%
2933   {%
2934     \ifcsdef{GLSxtr#1}%
2935     {%
2936       \@glsxtrp{GLSxtr#1}{#2}%
2937     }%
2938     {%
2939       \PackageError{glossaries-extra}{‘#1’ not recognised by
```

```

2940      \string\GLSxtrp}{}%
2941    }%
2942  }%
2943 }

```

\glsxtr@entry@p

```

2944 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
2945   \glsifattribute{#1}{headuc}{true}%
2946   {%
2947     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
2948   }%
2949   {%
2950     \@gls@entry@field{#1}{#2}%
2951   }%
2952 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

2953 \ifdef\teorpdfstring
2954 {
2955   \newcommand{\glsxtrp}[2]{%
2956     \protect\NoCaseChange
2957     {%
2958       \protect\teorpdfstring
2959       {%
2960         \protect\glsxtrifinmark
2961         {%
2962           \ifcsdef{glsxtrhead#1}%
2963             {%
2964               {\protect\csuse{glsxtrhead#1}{#2}}%
2965             }%
2966             {%
2967               \glsxtr@headentry@p{#2}{#1}%
2968             }%
2969           }%
2970           {%
2971             \@glsxtrp{#1}{#2}%
2972           }%
2973         }%
2974         {%
2975           \protect\@gls@entry@field{#2}{#1}%
2976         }%
2977       }%
2978     }
2979 }
2980 {
2981   \newcommand{\glsxtrp}[2]{%
2982     \protect\NoCaseChange
2983     {%
2984       \protect\glsxtrifinmark

```

```

2985     {%
2986     \ifcsdef{glxtrhead#1}%
2987     {%
2988         {\protect\csuse{glxtrhead#1}}%
2989     }%
2990     {%
2991         \glxtr@headentry@p{#2}{#1}%
2992     }%
2993 }%
2994 {%
2995     \@glxtrp{#1}{#2}%
2996 }%
2997 }%
2998 }
2999 }

```

Provide short synonyms for the most common option.

`\glsp`

```
3000 \newcommand*{\glsp}{\glxtrp{short}}
```

`\glsp`

```
3001 \newcommand*{\glsp}{\glxtrp{text}}
```

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

```

3002 \ifdef\teorpdfstring
3003 {
3004   \newcommand{\Glsxtrp}[2]{%
3005     \protect\NoCaseChange
3006     {%
3007       \protect\teorpdfstring
3008       {%
3009         \protect\glxtrifinmark
3010         {%
3011             \ifcsdef{Glsxtrhead#1}%
3012             {%
3013                 {\protect\csuse{Glsxtrhead#1}{#2}}%
3014             }%
3015             {%
3016                 \protect\@Gls@entry@field{#2}{#1}%
3017             }%
3018             }%
3019             {%
3020                 \@Glsxtrp{#1}{#2}%
3021             }%
3022             }%
3023             {%
3024                 \protect\@Gls@entry@field{#2}{#1}%

```

```

3025     }%
3026 }%
3027 }
3028 }
3029 {
3030   \newcommand{\Glsxtrp}[2]{%
3031     \protect\NoCaseChange
3032     {%
3033       \protect\glsxtrifinmark
3034       {%
3035         \ifcsdef{Glsxtrhead#1}%
3036         {%
3037           {\protect\csuse{Glsxtrhead#1}}%
3038         }%
3039         {%
3040           \protect\@Gls@entry@field{#2}{#1}%
3041         }%
3042       }%
3043       {%
3044         \@Glsxtrp{#1}{#2}%
3045       }%
3046     }%
3047   }
3048 }

```

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

```

3049 \ifdef\texorpdfstring
3050 {
3051   \newcommand{\Glsxtrp}[2]{%
3052     \protect\NoCaseChange
3053     {%
3054       \protect\texorpdfstring
3055       {%
3056         \protect\glsxtrifinmark
3057         {%
3058           \ifcsdef{Glsxtr#1}%
3059           {%
3060             {\protect\Glsxtrshort[noindex,hyper=false]{#1}[]}%
3061           }%
3062           {%
3063             \protect\mfirstucMakeUppercase
3064             {%
3065               \protect\@gls@entry@field{#2}{#1}%
3066             }%
3067           }%
3068         }%
3069         {%
3070           \@Glsxtrp{#1}{#2}%
3071         }%

```

```

3072     }%
3073     {%
3074         \protect\@gls@entry@field{#2}{#1}%
3075     }%
3076 }%
3077 }
3078 }
3079 {
3080 \newcommand{\GLSxtrp}[2]{%
3081     \protect\NoCaseChange
3082     {%
3083         \protect\glsxtrifinmark
3084         {%
3085             \ifcsdef{GLSxtr#1}%
3086             {%
3087                 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3088             }%
3089             {%
3090                 \protect\mfirstucMakeUppercase
3091                 {%
3092                     \protect\@gls@entry@field{#2}{#1}%
3093                 }%
3094             }%
3095         }%
3096     }%
3097     \@GLSxtrp{#1}{#2}%
3098 }%
3099 }%
3100 }
3101 }

```

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`s 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.
3102 \newcommand*{\@glsxtr@unset}[1]{%

```

```

3103 \@@glsunset{#1}%
3104 \glsxtrpostunset{#1}%
3105 }%

\@glsunset Global unset.
3106 \let\@glsunset\@glsxtr@unset

glsxtrpostunset
3107 \newcommand*{\glsxtrpostunset}[1]{%

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

tUnsetBuffering
3108 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3109 \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3110 }

tUnsetBuffering Unstarred version doesn't check for duplicates.
3111 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3112 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3113 \def\@glsxtr@unset@buffer{}}%
3114 \let\@glsunset\@glsxtrbuffer@unset
3115 }

tUnsetBuffering Starred version checks for duplicates.
3116 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3117 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3118 \def\@glsxtr@unset@buffer{}}%
3119 \let\@glsunset\@glsxtrbuffer@nodup@unset
3120 }

xtrbuffer@unset This must use a global change since \gls may have to be placed inside \mbox (for example,
with soul commands).
3121 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3122 \listxadd\@glsxtr@unset@buffer{#1}%
3123 }

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the ar-
gument 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.)
3124 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3125 \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}}%
3126 {\listxadd\@glsxtr@unset@buffer{#1}}%
3127 }

pUnsetBuffering
3128 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3129 \ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3130 }

```

pUnsetBuffering Unstarred form (global unset).

```

3131 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3132   \let\@glsunset\@glxtr@unset
3133   \forlistloop\@glsunset\@glxtr@unset@buffer
3134   \let\@glxtr@unset@buffer\@glxtr@org@unset@buffer
3135 }

```

pUnsetBuffering Starred form (local unset).

```

3136 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3137   \forlistloop\@glslocalunset\@glxtr@unset@buffer
3138   \let\@glsunset\@glxtr@unset
3139 }

```

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

```

3140 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3141   \forlistloop#1\@glxtr@unset@buffer
3142 }

```

\@glslocalunset Local unset.

```

3143 \renewcommand*{\@glslocalunset}[1]{%
3144   \@glslocalunset{#1}%
3145   \glxtrpostlocalunset{#1}%
3146 }%

```

rpostlocalunset

```

3147 \newcommand*{\glxtrpostlocalunset}[1]{%

```

\@glsreset Global reset.

```

3148 \renewcommand*{\@glsreset}[1]{%
3149   \@glsreset{#1}%
3150   \glxtrpostreset{#1}%
3151 }%

```

glxtrpostreset

```

3152 \newcommand*{\glxtrpostreset}[1]{%

```

\@glslocalreset Local reset.

```

3153 \renewcommand*{\@glslocalreset}[1]{%
3154   \@glslocalreset{#1}%
3155   \glxtrpostlocalreset{#1}%
3156 }%

```

rpostlocalreset

```

3157 \newcommand*{\glxtrpostlocalreset}[1]{%

```

slocalreseteach Locally reset a list of entries.

```

3158 \newcommand*{\glslocalreseteach}[1]{%
3159   \gls@ifnotmeasuring

```



```

3160  {%
3161    \@for\@gls@thislabel:=#1\do{%
3162      \glsdoifexists{\@gls@thislabel}%
3163      {%
3164        \@glslocalreset{\@gls@thislabel}%
3165      }%
3166    }%
3167  }%
3168 }

```

glslocalunseteach Locally unset a list of entries.

```

3169 \newcommand*{\glslocalunseteach}[1]{%
3170   \gls@ifnotmeasuring
3171   {%
3172     \@for\@gls@thislabel:=#1\do{%
3173       \glsdoifexists{\@gls@thislabel}%
3174       {%
3175         \@glslocalunset{\@gls@thislabel}%
3176       }%
3177     }%
3178   }%
3179 }

```

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

```

3180 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```

3181   \glsenableentrycount

```

Redefine \gls etc:

```

3182   \renewcommand*{\gls}{\cglsl}%
3183   \renewcommand*{\Gls}{\cGls}%
3184   \renewcommand*{\glspl}{\cglspl}%
3185   \renewcommand*{\Glspl}{\cGlspl}%
3186   \renewcommand*{\GLS}{\cGLS}%
3187   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3188   \@glsxtr@setentrycountunsetattr{#1}{#2}%

```

In case this command is used again:

```

3189   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3190   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3191     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3192       can't be used with \string\GlsXtrEnableEntryCounting}%
3193     {Use one or other but not both commands}}%
3194 }

```

glsycountunsetattr

```

3195 \newcommand*{\@glxtr@setentrycountunsetattr}[2]{%
3196   \@for\@glxtr@cat:=#1\do
3197   {%
3198     \ifdefempty{\@glxtr@cat}{}%
3199     {%
3200       \glsssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
3201     }%
3202   }%
3203 }

```

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

nableentrycount

```

3204 \renewcommand*{\glsenableentrycount}{%

```

Enable new fields:

```

3205   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%

```

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

```

3206   \renewcommand*{\gls@defdocnewglossaryentry}{%
3207     \renewcommand*{newglossaryentry}[2]{%
3208       \PackageError{glossaries}{\string\newglossaryentry\space
3209         may only be used in the preamble when entry counting has
3210         been activated}{If you use \string\glsenableentrycount\space
3211         you must place all entry definitions in the preamble not in
3212         the document environment}%
3213     }%
3214   }%

```

New commands to access new fields:

```

3215   \newcommand*{\glsentrycurrcount}[1]{%
3216     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3217     {0}{\@gls@entry@field{##1}{currcount}}%
3218   }%
3219   \newcommand*{\glsentryprevcount}[1]{%
3220     \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3221     {0}{\@gls@entry@field{##1}{prevcount}}%
3222   }%

```

Adjust post unset and reset:

```

3223   \let\@glxtr@entrycount@org@unset\glxtrpostunset
3224   \renewcommand*{\glxtrpostunset}[1]{%
3225     \@glxtr@entrycount@org@unset{##1}%
3226     \@gls@increment@currcount{##1}%
3227   }%
3228   \let\@glxtr@entrycount@org@localunset\glxtrpostlocalunset
3229   \renewcommand*{\glxtrpostlocalunset}[1]{%
3230     \@glxtr@entrycount@org@localunset{##1}%
3231     \@gls@local@increment@currcount{##1}%
3232   }%
3233   \let\@glxtr@entrycount@org@reset\glxtrpostreset

```

```

3234 \renewcommand*{\glxtrpostreset}[1]{%
3235   \@glxtr@entrycount@org@reset{##1}%
3236   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
3237 }%
3238 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
3239 \renewcommand*{\glxtrpostlocalreset}[1]{%
3240   \@glxtr@entrycount@org@localreset{##1}%
3241   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
3242 }%

```

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

```

3243 \let\@cgls@\@cgls@
3244 \let\@cglspl@\@cglspl@

3245 \let\@cGls@\@cGls@
3246 \let\@cGlspl@\@cGlspl@
3247 \let\@cGLS@\@cGLS@
3248 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

3249 \AtEndDocument{\@gls@write@entrycounts}%
3250 \renewcommand*{\@gls@entry@count}[2]{%
3251   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
3252 }%
3253 \let\glsenableentrycount\relax
3254 \renewcommand*{\glsenableentryunitcount}{%
3255   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3256     can't be used with \string\glsenableentrycount}%
3257   {Use one or other but not both commands}%
3258 }%
3259 }

```

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

```

3260 \renewcommand*{\@gls@write@entrycounts}{%
3261   \immediate\write\@auxout
3262     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3263   \count@=0\relax
3264   \forallglsentries{\@glsentry}{%
3265     \gls@hasattribute{\@glsentry}{entrycount}%
3266     {%
3267       \ifglsused{\@glsentry}%
3268       {%
3269         \immediate\write\@auxout
3270           {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3271       }%
3272     }%
3273     \advance\count@ by \@ne
3274   }%

```

```

3275     {}%
3276 }%
3277 \ifnum\count@=0
3278   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3279   \MessageBreak with \string\glsenableentrycount\space but the
3280   \MessageBreak attribute ‘entrycount’ hasn’t
3281   \MessageBreak been assigned to any of the defined
3282   \MessageBreak entries}%
3283 \fi
3284 }

```

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

```

3285 \newcommand*{\glxtrifcounttrigger}[3]{%
3286   \glshasattribute{#1}{entrycount}%
3287   {%
3288     \ifnum\gl Sentryprevcount{#1}>\gl sgetattribute{#1}{entrycount}\relax
3289     #3%
3290   \else
3291     #2%
3292   \fi
3293 }%
3294 {#3}%
3295 }

```

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

`\@@cgl`s@

```

3296 \def\@@cgl s@#1#2[#3]{%
3297   \glxtrifcounttrigger{#2}%
3298   {%
3299     \cgl sformat{#2}{#3}%
3300     \gl sunset{#2}%
3301   }%
3302   {%
3303     \@gl s@{#1}{#2}[#3]%
3304   }%
3305 }%

```

`\@@cgl spl`@

```

3306 \def\@@cgl spl@#1#2[#3]{%
3307   \glxtrifcounttrigger{#2}%
3308   {%
3309     \cgl splformat{#2}{#3}%
3310     \gl sunset{#2}%
3311   }%

```

```

3312  {%
3313    \@glsp1@{#1}-{#2}[#3]%
3314  }%
3315 }%

```

\@@cGls@

```

3316 \def\@@cGls@#1#2[#3]{%
3317   \glxtrifcounttrigger{#2}%
3318   {%
3319     \cGlsformat{#2}{#3}%
3320     \glunset{#2}%
3321   }%
3322   {%
3323     \@Gls@{#1}-{#2}[#3]%
3324   }%
3325 }%

```

\@@cGlsp1@

```

3326 \def\@@cGlsp1@#1#2[#3]{%
3327   \glxtrifcounttrigger{#2}%
3328   {%
3329     \cGlsp1format{#2}{#3}%
3330     \glunset{#2}%
3331   }%
3332   {%
3333     \@Glsp1@{#1}-{#2}[#3]%
3334   }%
3335 }%

```

\@@cGLS@

```

3336 \def\@@cGLS@#1#2[#3]{%
3337   \glxtrifcounttrigger{#2}%
3338   {%
3339     \cGLSformat{#2}{#3}%
3340     \glunset{#2}%
3341   }%
3342   {%
3343     \@GLS@{#1}-{#2}[#3]%
3344   }%
3345 }%

```

\@@cGLSp1@

```

3346 \def\@@cGLSp1@#1#2[#3]{%
3347   \glxtrifcounttrigger{#2}%
3348   {%
3349     \cGLSp1format{#2}{#3}%
3350     \glunset{#2}%
3351   }%
3352   {%

```

```

3353 \cGLSp1@{#1}-{#2}[#3]%
3354 }%
3355 }%

```

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

```

\cgl's@
3356 \def\cgl's@#1#2[#3]{\gls@{#1}-{#2}[#3]}

```

```

\cGls@
3357 \def\cGls@#1#2[#3]{\Gls@{#1}-{#2}[#3]}

```

```

\cgl'spl@
3358 \def\cgl'spl@#1#2[#3]{\glspl@{#1}-{#2}[#3]}

```

```

\cGlspl@
3359 \def\cGlspl@#1#2[#3]{\Glspl@{#1}-{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

```

\cGLS
3360 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\cGLS}

```

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

```

3361 \newcommand*{\cGLS}[2][ ]{%
3362 \new@ifnextchar[{\cGLS@{#1}-{#2}}{\cGLS@{#1}-{#2}[ ]}]%
3363 }

```

```

\cGLS@
3364 \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.

```

3365 \newcommand*{\cGLSformat}[2]{%
3366 \expandafter\mfirstucMakeUppercase\expandafter{\cgl'sformat{#1}-{#2}}%
3367 }

```

```

\cGLSp1
3368 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\cGLSp1}

```

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

```

3369 \newcommand*{\cGLSp1}[2][ ]{%
3370 \new@ifnextchar[{\cGLSp1@{#1}-{#2}}{\cGLSp1@{#1}-{#2}[ ]}]%
3371 }

```

```

\cGLSp1@
3372 \def\cGLSp1@#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.

```
3373 \newcommand*{\cGLSplformat}[2]{%
3374   \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3375 }
```

Modify the trigger formats to check for the regular attribute.

`\cglformat`

```
3376 \renewcommand*{\cglformat}[2]{%
3377   \glsifregular{#1}
3378   {\glsentryfirst{#1}}%
3379   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3380 }
```

`\cGlsformat`

```
3381 \renewcommand*{\cGlsformat}[2]{%
3382   \glsifregular{#1}
3383   {\Glsentryfirst{#1}}%
3384   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3385 }
```

`\cglsplformat`

```
3386 \renewcommand*{\cglsplformat}[2]{%
3387   \glsifregular{#1}
3388   {\glsentryfirstplural{#1}}%
3389   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3390 }
```

`\cGlsplformat`

```
3391 \renewcommand*{\cGlsplformat}[2]{%
3392   \glsifregular{#1}
3393   {\Glsentryfirstplural{#1}}%
3394   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%#2%
3395 }
```

New code similar to above for unit counting.

`defunitcounters`

```
3396 \newcommand*{\@newglossaryentry@defunitcounters}{%
3397   \edef\@glo@countunit{\csuse{@glxtr@categoryattr@{\@glo@category @unitcount}}}%
3398   \ifdefvoid\@glo@countunit
3399   {}%
3400   {%
3401     \@glxtr@ifunitcounter{\@glo@countunit}%
3402     {}%
3403     {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
3404     }%
3405 }
```

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

```
3406 \newcommand*{\@glsxtr@unitcountlist}{}
```

@addunitcounter

```
3407 \newcommand*{\@glsxtr@addunitcounter}[1]{%
3408 \listadd{\@glsxtr@unitcountlist}{#1}%
3409 \ifcsundef{glsxtr@theunit@#1}
3410 {%
3411 \ifcsdef{theH#1}%
3412 {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
3413 {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
3414 }%
3415 {}%
3416 }
```

r@ifunitcounter

```
3417 \newcommand*{\@glsxtr@ifunitcounter}[3]{%
3418 \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
3419 }
```

urrentunitcount

```
3420 \newcommand*\@glsxtr@currentunitcount[1]{%
3421 glo@\glsdetoklabel{#1}@currunit@glsggetattribute{#1}{unitcount}.%
3422 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3423 }
```

eviousunitcount

```
3424 \newcommand*\@glsxtr@previousunitcount[1]{%
3425 glo@\glsdetoklabel{#1}@prevunit@glsggetattribute{#1}{unitcount}.%
3426 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3427 }
```

t@currunitcount

```
3428 \newcommand*{\@gls@increment@currunitcount}[1]{%
3429 \glshasattribute{#1}{unitcount}%
3430 {%
3431 \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3432 \ifcsundef{\@glsxtr@csname}%
3433 {%
3434 \csgdef{\@glsxtr@csname}{1}%
3435 \listcsxadd
3436 {glo@\glsdetoklabel{#1}@unitlist}%
3437 {\glsggetattribute{#1}{unitcount}.%
3438 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3439 }%
3440 }%
3441 {%
3442 \csxdef{\@glsxtr@csname}%

```



```

3443      {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3444    }%
3445  }%
3446  {}%
3447 }

```

t@currunitcount

```

3448 \newcommand*{\@glx@local@increment@currunitcount}[1]{%
3449   \glshasattribute{#1}{unitcount}%
3450   {%
3451     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3452     \ifcsundef{\@glxtr@csname}%
3453     {%
3454       \csdef{\@glxtr@csname}{1}%
3455       \listcseadd
3456         {glo@\glsdetoklabel{#1}@unitlist}%
3457         {\glsggetattribute{#1}{unitcount}.%
3458          \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
3459     }%
3460   }%
3461   {%
3462     \csedef{\@glxtr@csname}%
3463       {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3464   }%
3465 }%
3466 {}%
3467 }

```

r@currunitcount

```

3468 \newcommand*{\@glxtr@currunitcount}[2]{%
3469   \ifcsundef
3470     {glo@\glsdetoklabel{#1}@currunit@#2}%
3471     {0}%
3472   {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
3473 }%

```

r@prevunitcount

```

3474 \newcommand*{\@glxtr@prevunitcount}[2]{%
3475   \ifcsundef
3476     {glo@\glsdetoklabel{#1}@prevunit@#2}%
3477     {0}%
3478   {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
3479 }%

```

eentryunitcount

```

3480 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3481   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%

```

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

```

3482 \renewcommand*{\gls@defdocnewglossaryentry}{%
3483 \renewcommand*\newglossaryentry[2]{%
3484 \PackageError{glossaries}{\string\newglossaryentry\space
3485 may only be used in the preamble when entry counting has
3486 been activated}{If you use \string\glsenableentryunitcount\space
3487 you must place all entry definitions in the preamble not in
3488 the document environment}}%
3489 }%
3490 }%

```

New commands to access new fields:

```

3491 \newcommand*{\glsentrycurrcount}[1]{%
3492 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3493 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3494 }%
3495 \newcommand*{\glsentryprevcount}[1]{%
3496 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3497 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
3498 }%

```

Access total count:

```

3499 \newcommand*{\glsentryprevtotalcount}[1]{%
3500 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}}%
3501 {0}%
3502 {%
3503 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3504 }%
3505 }%

```

Access max value:

```

3506 \newcommand*{\glsentryprevmaxcount}[1]{%
3507 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}}%
3508 {0}%
3509 {%
3510 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3511 }%
3512 }%

```

Adjust post unset and reset:

```

3513 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
3514 \renewcommand*{\glsxtrpostunset}[1]{%
3515 \@glsxtr@entryunitcount@org@unset{##1}}%
3516 \@gls@increment@currunitcount{##1}}%
3517 }%
3518 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
3519 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3520 \@glsxtr@entryunitcount@org@localunset{##1}}%
3521 \@gls@local@increment@currunitcount{##1}}%
3522 }%
3523 \let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset

```

```

3524 \renewcommand*{\glxtrpostreset}[1]{%
3525   \glshasattribute{##1}{unitcount}%
3526   {%
3527     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3528     \ifcsundef{\@glxtr@csname}%
3529     {}%
3530     {\csgdef{\@glxtr@csname}{0}}%
3531   }%
3532   {}%
3533 }%
3534 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
3535 \renewcommand*{\glxtrpostlocalreset}[1]{%
3536   \@glxtr@entryunitcount@org@localreset{##1}%
3537   \glshasattribute{##1}{unitcount}%
3538   {%
3539     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3540     \ifcsundef{\@glxtr@csname}%
3541     {}%
3542     {\csdef{\@glxtr@csname}{0}}%
3543   }%
3544   {}%
3545 }%

```

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

```

3546 \let\@cgl@\@cgl@
3547 \let\@cgl@pl@\@cgl@pl@

3548 \let\@cGl@\@cGl@
3549 \let\@cGl@pl@\@cGl@pl@
3550 \let\@cGL@\@cGL@
3551 \let\@cGL@pl@\@cGL@pl@

```

Write information to the aux file.

```

3552 \AtEndDocument{\@gl@write@entryunitcounts}%
3553 \renewcommand*{\@gl@entry@unitcount}[3]{%
3554   \csgdef{glo@gl@detoklabel{##1}@prevunit@##3}{##2}%
3555   \ifcsundef{glo@gl@detoklabel{##1}@prevunittotal}%
3556   {\csgdef{glo@gl@detoklabel{##1}@prevunittotal}{##2}}%
3557   {%
3558     \csxdef{glo@gl@detoklabel{##1}@prevunittotal}{
3559       \number\numexpr\csuse{glo@gl@detoklabel{##1}@prevunittotal}+##2}%
3560     }%
3561     \ifcsundef{glo@gl@detoklabel{##1}@prevunitmax}%
3562     {\csgdef{glo@gl@detoklabel{##1}@prevunitmax}{##2}}%
3563     {%
3564       \ifnum\csuse{glo@gl@detoklabel{##1}@prevunitmax}<##2
3565       \csgdef{glo@gl@detoklabel{##1}@prevunitmax}{##2}%
3566       \fi
3567     }%

```

```

3568 }%
3569 \let\glsenableentryunitcount\relax
3570 \renewcommand*{\glsenableentrycount}{%
3571   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3572     can't be used with \string\glsenableentryunitcount}%
3573   {Use one or other but not both commands}%
3574 }%
3575 }
3576 \@onlypreamble\glsenableentryunitcount

entry@unitcount
3577 \newcommand*{\@gls@entry@unitcount}[3]{%

ryunitcounts@do
3578 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3579   \immediate\write\@auxout
3580   {\string\@gls@entry@unitcount
3581     {\@glsentry}%
3582     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3583     }%
3584     {#1}}%
3585 }

entryunitcounts
3586 \newcommand*{\@gls@write@entryunitcounts}{%
3587   \immediate\write\@auxout
3588   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3589   \count@=0\relax
3590   \forallglsentries{\@glsentry}{%
3591     \gls@hasattribute{\@glsentry}{unitcount}%
3592     {%
3593       \ifglsused{\@glsentry}%
3594       {%
3595         \forlistcsloop
3596           {\@gls@write@entryunitcounts@do}%
3597           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3598       }%
3599     }%
3600     \advance\count@ by \@ne
3601   }%
3602 }%
3603 }%
3604 \ifnum\count@=0
3605   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3606     \MessageBreak with \string\glsenableentryunitcount\space but the
3607     \MessageBreak attribute 'unitcount' hasn't
3608     \MessageBreak been assigned to any of the defined
3609     \MessageBreak entries}%
3610 \fi

```

3611 }

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.

3612 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

Enable entry counting:

3613 \glsenableentryunitcount

Redefine \gls etc:

3614 \renewcommand*{\gls}{\cglsl}%
 3615 \renewcommand*{\Gls}{\cGls}%
 3616 \renewcommand*{\glspl}{\cglspl}%
 3617 \renewcommand*{\Glspl}{\cGlspl}%
 3618 \renewcommand*{\GLS}{\cGLS}%
 3619 \renewcommand*{\GLSpl}{\cGLSpl}%

Set the entrycount attribute:

3620 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

In case this command is used again:

3621 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
 3622 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
 3623 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
 3624 can't be used with \string\GlsXtrEnableEntryUnitCounting}%
 3625 {Use one or other but not both commands}}%
 3626 }

countunsetattr

3627 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
 3628 \@for\@glsxtr@cat:=#1\do
 3629 {%
 3630 \ifdefempty{\@glsxtr@cat}{}%
 3631 {%
 3632 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
 3633 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
 3634 }%
 3635 }%
 3636 }

1.3.6 Acronym Modifications

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

nericNewAcronym

```

3637 \renewcommand*{\SetGenericNewAcronym}{%
3638   \let\@Gls@entryname\@Gls@acrenentryname
3639   \renewcommand{\newacronym}[4][{}]{%
3640     \ifdefempty{\@glsacronymlists}%
3641     {%
3642       \def\@glo@type{\acronymtype}%
3643       \setkeys{glossentry}{##1}%
3644       \DeclareAcronymList{\@glo@type}%
3645     }%
3646   }%
3647   \glskeylisttok{##1}%
3648   \glslabeltok{##2}%
3649   \glsshorttok{##3}%
3650   \gslongtok{##4}%
3651   \newacronymhook
3652   \protected@edef\@do@newglossaryentry{%
3653     \noexpand\newglossaryentry{\the\glslabeltok}%
3654     {%
3655       type=\acronymtype,%
3656       name={\expandonce{\acronymentry{##2}}},%
3657       sort={\acronymsort{\the\glsshorttok}{\the\gslongtok}},%
3658       text={\the\glsshorttok},%
3659       short={\the\glsshorttok},%
3660       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3661       long={\the\gslongtok},%
3662       longplural={\the\gslongtok\noexpand\acrpluralsuffix},%
3663       category=acronym,%
3664       \GenericAcronymFields,%
3665       \the\glskeylisttok
3666     }%
3667   }%
3668   \@do@newglossaryentry
3669 }%
3670 \renewcommand*{\acrfullfmt}[3]{%
3671   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3672 \renewcommand*{\Acrfullfmt}[3]{%
3673   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3674 \renewcommand*{\ACRfullfmt}[3]{%
3675   \glslink[##1]{##2}{%
3676     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3677 \renewcommand*{\acrfullplfmt}[3]{%
3678   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3679 \renewcommand*{\Acrfullplfmt}[3]{%
3680   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3681 \renewcommand*{\ACRfullplfmt}[3]{%
3682   \glslink[##1]{##2}{%
3683     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3684 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%

```

```

3685 \renewcommand*{\Glsentryfull}[1]{\Genacrformat{##1}{}}%
3686 \renewcommand*{\glentryfullpl}[1]{\genplacrformat{##1}{}}%
3687 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrformat{##1}{}}%
3688 }

```

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:

```

3689 \let\@glxtr@org@setacronymstyle\setacronymstyle
3690 \let\@glxtr@org@newacronymstyle\newacronymstyle

```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbreviationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```

3691 \newcommand*{\MakeAcronymsAbbreviations}{%
3692   \renewcommand*{\newacronym}[4][]{%
3693     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3694   }%
3695   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
3696   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
3697   \renewcommand*{\setacronymstyle}[1]{%
3698     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
3699       unavailable.
3700       Use \string\setabbreviationstyle\space instead.
3701       The original acronym interface can be restored with
3702       \string\RestoreAcronyms}{}%
3703   }%
3704   \renewcommand*{\newacronymstyle}[1]{%
3705     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3706       available unless you restore the original acronym interface with
3707       \string\RestoreAcronyms}%
3708     \@glxtr@org@newacronymstyle{##1}%
3709   }%
3710 }

```

Switch acronyms to abbreviations:

```

3711 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3712 \newcommand*{\RestoreAcronyms}{%
3713   \SetGenericNewAcronym
3714   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3715   \renewcommand{\acronymfont}[1]{##1}%
3716   \let\setacronymstyle\@glxtr@org@setacronymstyle
3717   \let\newacronymstyle\@glxtr@org@newacronymstyle

```

Need to restore the original definition of `\@gls@link@checkfirsthyper` but `\glxtrifwasfirstuse` still needs setting for the benefit of the post-link hook.

```

3718 \renewcommand*\@gls@link@checkfirsthyper{%
3719   \ifglsused{\glslabel}%
3720   {\let\glsxtrifwasfirsttuse\@secondoftwo}
3721   {\let\glsxtrifwasfirsttuse\@firstoftwo}%
3722   \@glsxtr@org@checkfirsthyper
3723 }
3724 \glssetcategoryattribute{acronym}{regular}{false}%
3725 \setacronymstyle{long-short}%
3726 }

```

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

```

3727 \renewcommand*\@glsacspace}[1]{%
3728   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3729   \ifdim\dimen@<\glsacspacemax~\else\space\fi
3730 }

```

`\glsacspacemax` Value used in the above.

```

3731 \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`

```

3732 \newcommand*\@glsxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

3733 \let\@glsxtr@org@makeglossaries\makeglossaries

```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with `record`.

`\makeglossaries`

```

3734 \renewcommand*\@makeglossaries[#1][]{%
3735   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
3736     \PackageError{glossaries-extra}{\string\makeglossaries\space
3737       not permitted\MessageBreak with record=only package option}%
3738     {You may only use \string\makeglossaries\space with
3739       record=off or record=alsoindex options}%
3740   \else
3741     \ifblank{#1}%
3742     {\@glsxtr@org@makeglossaries}%
3743     {%
3744       \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex

```



```

3745 \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3746 not permitted\MessageBreak with record=alsoindex package option}%
3747 {You may only use the hybrid \string\makeglossaries[...]\space with
3748 record=off option}%
3749 \else
3750 \edef\@glxtr@reg@glosslist{#1}%
3751 \ifundef{\glswrite}{\newwrite\glswrite}{}%
3752 \protected@write\@auxout{}{\string\providecommand
3753 \string\@glorder[1]{}%
3754 \protected@write\@auxout{}{\string\providecommand
3755 \string\@istfilename[1]{}%
3756 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3757 \protected@write\@auxout{}{\string\@glorder{\glorder}}%
3758 \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}%
3759 \write\@auxout{\string\providecommand\string\@gls@reference[3]{}%

```

Iterate through each supplied glossary type and activate it.

```

3760 \@for\@glo@type:=#1\do{%
3761 \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3762 }%

```

New glossaries must be created before \makeglossaries:

```

3763 \renewcommand*\newglossary[4][]{%
3764 \PackageError{glossaries}{New glossaries
3765 must be created before \string\makeglossaries}{You need
3766 to move \string\makeglossaries\space after all your
3767 \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```

3768 \let\@makeglossary\relax
3769 \let\makeglossary\relax
3770 \renewcommand\makeglossaries[1][]{}%

```

Disable all commands that have no effect after \makeglossaries

```

3771 \@disable@onlypremakeg

```

Allow see key:

```

3772 \let\gls@checkseeallowed\relax

```

Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```

3773 \renewcommand*\@do@seeglossary[2]{%
3774 \glsdoifexists{##1}%
3775 {%
3776 \edef\@gls@label{\glsdetoklabel{##1}}%
3777 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3778 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
3779 {\@glxtr@org@doseeglossary{##1}{##2}}%
3780 {%
3781 \@@glxtrwrglossmark
3782 \protected@write\@auxout{}{%
3783 \string\@gls@reference

```

```

3784             {\gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
3785         }%
3786     }%
3787 }%
3788 }%

Adjust \@do@wrglossary
3789     \let\glsxtr@do@wrglossary\@do@wrglossary
3790     \def\@do@wrglossary{%
3791         \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3792         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3793         {\@glsxtr@do@wrglossary}%
3794         {\gls@noidxglossary}%
3795     }%

Suppress warning about no \makeglossaries
3796     \let\warn@nomakeglossaries\relax
3797     \def\warn@noprintglossary{%
3798         \GlossariesWarningNoLine{No \string\printglossary\space
3799             or \string\printglossaries\space
3800             found.^^J(Remove \string\makeglossaries\space if you don't want
3801             any glossaries.)^^JThis document will not have a glossary}%
3802     }%

Only warn for glossaries not listed.
3803     \renewcommand{\@gls@noref@warn}[1]{%
3804         \edef\@gls@type{##1}%
3805         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3806         {%
3807             \GlossariesExtraWarning{Can't use
3808                 \string\printnoidxglossary[type={\@gls@type}]
3809                 when '\@gls@type' is listed in the optional argument of
3810                 \string\makeglossaries}%
3811         }%
3812         {%
3813             \GlossariesWarning{Empty glossary for
3814                 \string\printnoidxglossary[type={##1}].
3815                 Rerun may be required (or you may have forgotten to use
3816                 commands like \string\gls)}%
3817         }%
3818     }%

Adjust display number list to check for type:
3819     \renewcommand*\@glsdisplaynumberlist[1]{%
3820         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3821         {\@glsxtr@idx@displaynumberlist{##1}}%
3822         {\@glsxtr@noidx@displaynumberlist{##1}}%
3823     }%

Adjust entry list:
3824     \renewcommand*\@glsentrynumberlist[1]{%

```

```

3825         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
3826         {\@glxtr@idx@entrynumberlist{##1}}}%
3827         {\@glxtr@noidx@entrynumberlist{##1}}}%
3828     }%

```

Adjust number list loop

```

3829     \renewcommand*{\glsnumberlistloop}[2]{%
3830         \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
3831         {%
3832             \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3833             not available for glossary ‘##1’}{}%
3834         }%
3835         {\@glxtr@noidx@numberlistloop{##1}{##2}}}%
3836     }%

```

Only sanitize sort for normal indexing glossaries.

```

3837     \renewcommand*{\glsprestandardsort}[3]{%
3838         \expandafter\DTLifinlist\expandafter{##2}{\@glxtr@reg@glosslist}%
3839         {%
3840             \glsdosanitizesort
3841         }%
3842         {%
3843             \ifglssanitizesort
3844                 \@gls@noidx@sanitizesort
3845             \else
3846                 \@gls@noidx@nosanitizesort
3847             \fi
3848         }%
3849     }%

```

Unlike `\makenoidxglossaries` we can’t automatically set `sanitizesort=false`. All entries must be defined in the preamble.

```

3850     \renewcommand*\new@glossaryentry[2]{%
3851         \PackageError{glossaries-extra}{Glossary entries must be defined
3852         in the preamble\MessageBreak when you use the optional argument
3853         of \string\makeglossaries}{Either move your definitions to the
3854         preamble or don’t use the optional argument of
3855         \string\makeglossaries}%
3856     }%

```

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).

```

3857     \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
3858     \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3859         \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
3860         type=\glsdefaulttype,\@end@glxtr@gettype
3861         \def\@glo@sorttype{\@glo@default@sorttype}%
3862     }%

```

Check automake setting:

```

3863     \ifglsautomake
3864     \renewcommand*{\@gls@doautomake}{%
3865         \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
3866             \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3867         }%
3868     }%
3869 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3870     \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3871 \fi
3872 }%
3873 \fi
3874 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\@printglossary` 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.)

```

3875 \newcommand{\@glsxtr@orgprintglossary}[2]{%
3876 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3877 \def\glossarytitle{%
3878     \ifcsdef{\@glo@type\@glo@type @title}%
3879     {\csuse{\@glo@type\@glo@type @title}}%
3880     {\glossaryname}}%
3881 \def\glossarytoctitle{\glossarytitle}%
3882 \let\org@glossarytitle\glossarytitle
3883 \def\@glossarystyle{%
3884     \ifx\@glossary@default@style\relax
3885         \GlossariesWarning{No default glossary style provided \MessageBreak
3886             for the glossary '\@glo@type'. \MessageBreak
3887             Using deprecated fallback. \MessageBreak
3888             To fix this set the style with \MessageBreak
3889             \string\setglossarystyle\space or use the \MessageBreak
3890             style key=value option}%
3891 \fi
3892 }%
3893 \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
3894 \let\@org@glossaryentrynumbers\glossaryentrynumbers
3895 \bgroup
3896     \@printgloss@setsort
3897     \setkeys{printgloss}{#1}%
3898     \ifx\glossarytitle\org@glossarytitle
3899     \else

```

```

3900     \cslet{@glo@type@}@glo@type @title-{\glossarytitle}%
3901     \fi
3902     \let\currentglossary\@glo@type
3903     \let\org@glossaryentrynumbers\glossaryentrynumbers
3904     \let\glsnonextpages\@glsnonextpages
3905     \let\glsnextpages\@glsnextpages

3906     \glsxtractivatenopost
3907     \gls@dotoc@title
3908     \@glossarystyle
3909     \let\gls@org@glossaryentryfield\glossentry
3910     \let\gls@org@glossarysubentryfield\subglossentry
3911     \renewcommand{\glossentry}[1]{%
3912         \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3913         \gls@org@glossaryentryfield{##1}%
3914     }%
3915     \renewcommand{\subglossentry}[2]{%
3916         \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3917         \gls@org@glossarysubentryfield{##1}{##2}%
3918     }%
3919     \@gls@preglossaryhook
3920     #2%
3921     \egroup
3922     \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
3923     \global\let\warn@noprintglossary\relax
3924 }

```

xtractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

3925 \newcommand*{\glsxtractivatenopost}{%
3926     \let\nopostdesc\@nopostdesc
3927     \let\glsxtrnopostpunc\@glsxtr@nopostpunc
3928 }

```

lsxtrnopostpunc

```

3929 \newrobustcmd*{\glsxtrnopostpunc}{%

```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

3930 \newcommand{\@glsxtr@nopostpunc}{%
3931     \let\@glsxtr@org@postdescription\glspostdescription
3932     \ifglsnopostdot
3933         \renewcommand{\glspostdescription}{%
3934             \glsnopostdottrue
3935             \let\glspostdescription\@glsxtr@org@postdescription
3936             \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
3937             \glsxtrpostdescription
3938             \@glsxtr@nopostpunc@postdesc}%
3939     \else
3940         \renewcommand{\glspostdescription}{%

```

```

3941 \let\glspostdescription\@glsxtr@org@postdescription
3942 \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
3943 \glsxtrpostdescription
3944 \@glsxtr@nopostpunc@postdesc}%
3945 \fi
3946 \glsnopostdotfalse
3947 }

```

stpunc@postdesc

```

3948 \newcommand*{\@glsxtr@nopostpunc@postdesc}{}

```

estore@postpunc

```

3949 \newcommand*{\@glsxtr@restore@postpunc}{%
3950 \def\@glsxtr@nopostpunc@postdesc{%
3951 \@glsxtr@org@postdescription
3952 \let\@glsxtr@nopostpunc@postdesc\@empty
3953 \let\glsxtrrestorepostpunc\@empty
3954 }%
3955 }

```

restorepostpunc Does nothing outside of glossary.

```

3956 \newcommand*{\glsxtrrestorepostpunc}{}

```

\@printglossary Redefine.

```

3957 \renewcommand{\@printglossary}[2]{%
3958 \def\@glsxtr@printglossopts{#1}%
3959 \@glsxtr@orgprintglossary{#1}{#2}%
3960 }

```

Add a key that switches off the entry targets:

```

3961 \define@choicekey{printgloss}{target}
3962 [\@glsxtr@printglossval\@glsxtr@printglossnr]%
3963 {true,false}[true]%
3964 {%
3965 \ifcase\@glsxtr@printglossnr
3966 \def\@glstarget{\glsdohypertarget}%
3967 \else
3968 \let\@glstarget\@secondoftwo
3969 \fi
3970 }

```

hypernameprefix

```

3971 \newcommand{\@glsxtrhypernameprefix}{}

```

New to v1.20:

```

3972 \define@key{printgloss}{targetnameprefix}{%
3973 \renewcommand{\@glsxtrhypernameprefix}{#1}%
3974 }

```

```

3975 \define@key{printgloss}{prefix}{%
3976   \renewcommand{\glo@linkprefix}{#1}%
3977 }

```

`\lsdohypertarget` Redefine to insert `\@glxtrhypernameprefix` before the target name.

```

3978 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
3979 \renewcommand{\glsdohypertarget}[2]{%
3980   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
3981 }

```

Update `\@glstarget` to use `\def` instead being assigned with `\let` so that it can pick up the new definition and allow any further redefinitions:

```

3982 \ifx\@glstarget\@glxtr@org@glsdohypertarget
3983   \def\@glstarget{\glsdohypertarget}%
3984 \fi
3985 %\end{macro}

```

`@makeglossaries` For the benefit of `makeglossaries`

```

3986 \newcommand*{\@glxtr@makeglossaries}[1]{%

```

`@glxtr@gettype` Get just the type.

```

3987 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
3988   \def\@glo@type{#2}%
3989 }

```

`@assign@sortkey` Assign the sort key.

```

3990 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
3991   \edef\@glo@type{\@glo@type}%
3992   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
3993   {%
3994     \@glo@no@assign@sortkey{#1}%
3995   }%
3996   {%
3997     \@glo@assign@sortkey{#1}%
3998   }%
3999 }%

```

Display number list for the regular version:

`splaynumberlist`

```

4000 \let\@glxtr@idx@displaynumberlist\glisplaynumberlist

```

Display number list for the “noidx” version:

`splaynumberlist`

```

4001 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4002   \letcs{\@gls@loclist}{glo@\glsetoklabel{#1}@loclist}%
4003   \ifdef\@gls@loclist
4004   {%

```

```

4005 \def\@gls@noidxloclist@sep{%
4006 \def\@gls@noidxloclist@sep{%
4007 \def\@gls@noidxloclist@sep{%
4008 \glsnumlistsep
4009 }%
4010 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4011 }%
4012 }%
4013 \def\@gls@noidxloclist@finalsep{}%
4014 \def\@gls@noidxloclist@prev{}%
4015 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4016 \@gls@noidxloclist@finalsep
4017 \@gls@noidxloclist@prev
4018 }%
4019 {%

4020 \glsxtrundeftag
4021 \glsdoifexists{#1}%
4022 {%
4023 \GlossariesWarning{Missing location list for ‘#1’. Either
4024 a rerun is required or you haven’t referenced the entry.}%
4025 }%
4026 }%
4027 }%
4028

```

And for the number list loop:

@numberlistloop

```

4029 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
4030 \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4031 \let\@gls@org\@glsnoidxdisplayloc\@glsnoidxdisplayloc
4032 \let\@gls@org\@glsseeformat\@glsseeformat
4033 \let\@glsnoidxdisplayloc#2\relax
4034 \let\@glsseeformat#3\relax
4035 \ifdef\@gls@loclist
4036 {%
4037 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4038 }%
4039 {%

4040 \glsxtrundeftag
4041 \glsdoifexists{#1}%
4042 {%
4043 \GlossariesWarning{Missing location list for ‘##1’. Either
4044 a rerun is required or you haven’t referenced the entry.}%
4045 }%
4046 }%
4047 \let\@glsnoidxdisplayloc\@gls@org\@glsnoidxdisplayloc
4048 \let\@glsseeformat\@gls@org\@glsseeformat
4049 }%

```


Same for entry number list.

entrynumberlist

```

4050 \newcommand*{\@glxtr@noidx@entrynumberlist}[1]{%
4051   \letcs{\@glxtr@loclist}{\glo@glxtr@detoklabel{#1}@loclist}%
4052   \ifdef\@glxtr@loclist
4053   {%
4054     \glxtr@noidx@loclist{\@glxtr@loclist}%
4055   }%
4056   {%

4057   \glxtr@undefrag
4058   \glxtr@doifexists{#1}%
4059   {%
4060     \GlossariesWarning{Missing location list for ‘#1’. Either
4061       a rerun is required or you haven’t referenced the entry.}%
4062   }%
4063 }%
4064 }%
```

entrynumberlist

```

4065 \newcommand*{\@glxtr@idx@entrynumberlist}[1]{\glxtr@entrynumberlist{#1}}
```

x@getgrouptitle Patch.

```

4066 \renewcommand*{\@glxtr@noidx@getgrouptitle}[2]{%
4067   \protected@edef\@glxtr@titlelabel{#1}%
4068   \ifdefvoid\@glxtr@titlelabel
4069   {}%
4070   {%
4071     \protected@edef\@glxtr@titlelabel{\csuse{\glxtr@grouptitle@#1}}%
4072   }%
4073   \ifdefvoid{\@glxtr@titlelabel}%
4074   {%
4075     \DTLifint{#1}%
4076     {%
4077       \ifnum#1<256\relax
4078       \edef#2{\char#1\relax}%
4079     \else
4080       \edef#2{#1}%
4081     \fi
4082   }%
4083   {%
4084     \ifcsundef{#1groupname}%
4085     {\def#2{#1}}%
4086     {\letcs#2{#1groupname}}%
4087   }%
4088 }%
4089 {%
4090   \let#2\@glxtr@titlelabel
```

```

4091 }%
4092 }

g@getgrouptitle Save original definition of \@gls@getgrouptitle
4093 \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.
4094 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4095   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4096   \@onelevel@sanitize\@glsxtr@titlelabel
4097   \ifcsdef{\@glsxtr@titlelabel}
4098     {\letcs{#2}{\@glsxtr@titlelabel}}%
4099     {\glsxtr@org@getgrouptitle{#1}{#2}}%
4100 }
4101 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

trsetgrouptitle Sets the title for the given group label.
4102 \newcommand{\glsxtrsetgrouptitle}[2]{%
4103   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4104   \@onelevel@sanitize\@glsxtr@titlelabel
4105   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4106 }

alsetgrouptitle As above put only locally defines the title.
4107 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4108   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4109   \@onelevel@sanitize\@glsxtr@titlelabel
4110   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4111 }

\glsnavigation Redefine to use new user-level command.
4112 \renewcommand*{\glsnavigation}{%
4113   \def\@gls@between{}%
4114   \ifcsundef{\@gls@hypergroupelist@\@glo@type}%
4115     {%
4116       \def\@gls@list{}%
4117     }%
4118     {%
4119       \expandafter\let\expandafter\@gls@list
4120         \csname @gls@hypergroupelist@\@glo@type\endcsname
4121     }%
4122     \@for\@gls@tmp:=\@gls@list\do{%
4123       \@gls@between
4124       \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4125       \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4126       \let\@gls@between\glshypernavsep
4127     }%
4128 }

```

@noidx@glossary

```
4129 \renewcommand*{\@print@noidx@glossary}{%
4130   \ifcsdef{@glsref@{\@glo@type}}%
4131   {%
4132     \ifcsdef{@glo@sortmacro@{\@glo@sorttype}}%
4133     {%
4134       \csuse{@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
4135     }%
4136   }%
4137   \PackageError{glossaries}{Unknown sort handler '@glo@sorttype'}{ }%
4138 }%
4139 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4140 \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
4141 \def{@gls@currentlettergroup}{%
4142 \begin{theglossary}%
4143 \glossaryheader
4144 \glsresetentrylist
4145 \forlistcsloop{@gls@noidx@do}{@glsref@{\@glo@type}}%
4146 \end{theglossary}%
4147 \glossarypostamble
4148 }%
4149 {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
4150 \glstrifemptyglossary{\@glo@type}%
4151 }%
4152 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4153 \@gls@noref@warn{\@glo@type}%
4154 }%
4155 }
```

noidxdisplayloc Patch to check for range formations.

```
4156 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4157 \setentrycounter[#1]{#2}%
4158 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4159 }
```

xtr@display@loc Patch to check for range formations.

```
4160 \def{@glsxtr@display@loc#1#2\end@glxtr@display@loc#3}{%
4161 \ifx#1(\relax
4162 \glxtrdisplaystartloc{#2}{#3}%
4163 \else
4164 \ifx#1)\relax
4165 \glxtrdisplayendloc{#2}{#3}%
4166 \else
```

```

4167      \glxtrdisplaysingleloc{#1#2}{#3}%
4168      \fi
4169      \fi
4170 }

```

`\displaysingleloc` Single location.

```

4171 \newcommand*{\glxtrdisplaysingleloc}[2]{%
4172   \csuse{#1}{#2}%
4173 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glxtrlocrangefmt`.

`\displaystartloc` Start of a location range.

```

4174 \newcommand*{\glxtrdisplaystartloc}[2]{%
4175   \edef\glxtrlocrangefmt{#1}%
4176   \ifx\glxtrlocrangefmt\empty
4177     \def\glxtrlocrangefmt{glsnumberformat}%
4178   \fi
4179   \expandafter\glxtrdisplaysingleloc
4180   \expandafter{\glxtrlocrangefmt}{#2}%
4181 }

```

`\trdisplayendloc` End of a location range.

```

4182 \newcommand*{\glxtrdisplayendloc}[2]{%
4183   \edef\@glxtr@tmp{#1}%
4184   \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{glsnumberformat}}{}%
4185   \ifx\glxtrlocrangefmt\@glxtr@tmp
4186     \else
4187       \GlossariesExtraWarning{Mismatched end location range
4188         (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
4189     \fi
4190     \expandafter\glxtrdisplayendloohook\expandafter{\@glxtr@tmp}{#2}%
4191     \expandafter\glxtrdisplaysingleloc
4192     \expandafter{\glxtrlocrangefmt}{#2}%
4193     \def\glxtrlocrangefmt{}%
4194 }

```

`\splayendloohook` Allow the user to hook into the end of range command.

```

4195 \newcommand*{\glxtrdisplayendloohook}[2]{%

```

`\sxtrlocrangefmt` Current range format. Empty if not in a range.

```

4196 \newcommand*{\glxtrlocrangefmt}{}

```

`\setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4197 \renewcommand*{\setentrycounter}[2][{}]{%
4198   \def\glxtrcounterprefix{#1}%
4199   \ifx\glxtrcounterprefix\@empty

```

```

4200 \def\@glo@counterprefix{.}%
4201 \else
4202 \def\@glo@counterprefix{.#1.}%
4203 \fi
4204 \def\glstrycounter{#2}%
4205 }

```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4206 \def\@gls@removespaces#1 #2\@nil{%
4207 \toks@=\expandafter{\the\toks@#1}%
4208 \ifx\#2\%
4209 \edef\x{\the\toks@}%
4210 \ifx\x\empty
4211 \else
4212 \expandafter\glstrylocationhyperlink\expandafter
4213 \glstrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4214 \fi
4215 \else
4216 \@gls@ReturnAfterFi{%
4217 \@gls@removespaces#2\@nil
4218 }%
4219 \fi
4220 }

```

`cationhyperlink`

```

4221 \newcommand*\glstrylocationhyperlink}[3]{%
4222 \ifdefined\glstrylocationurl
4223 {%
4224 \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4225 }%
4226 {%
4227 \hyperref{\glstrylocationurl}{#1#2#3}{#3}%
4228 }%
4229 }

```

`supphypernumber`

```

4230 \newcommand*\glstryhypernumber}[1]{%
4231 {%
4232 \glshasattribute{\glstrycurrententrylabel}{externallocation}%
4233 {%
4234 \def\glstrylocationurl{%
4235 \glshasattribute{\glstrycurrententrylabel}{externallocation}}%
4236 }%
4237 {%
4238 \def\glstrylocationurl{}%
4239 }%
4240 \glshypernumber{#1}%

```

```

4241 }%
4242 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

```
@print@glossary
```

```

4243 \renewcommand{\@print@glossary}{%
4244   \makeatletter
4245   \@input@{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
4246   \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
4247   {}%
4248   {\glstrNoGlossaryWarning{\@glo@type}}%
4249   \ifglxindy
4250     \ifcsundef{@xdy@\@glo@type @language}%
4251     {%
4252       \edef\@do@auxoutstuff{%
4253         \noexpand\AtEndDocument{%
4254           \noexpand\immediate\noexpand\write\@auxout{%
4255             \string\providecommand\string\@xdylanguage[2]{}}%
4256           \noexpand\immediate\noexpand\write\@auxout{%
4257             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4258         }%
4259       }%
4260     }%
4261     {%
4262       \edef\@do@auxoutstuff{%
4263         \noexpand\AtEndDocument{%
4264           \noexpand\immediate\noexpand\write\@auxout{%
4265             \string\providecommand\string\@xdylanguage[2]{}}%
4266           \noexpand\immediate\noexpand\write\@auxout{%
4267             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4268               @language\endcsname}}%
4269         }%
4270       }%
4271     }%
4272     \@do@auxoutstuff
4273     \edef\@do@auxoutstuff{%
4274       \noexpand\AtEndDocument{%
4275         \noexpand\immediate\noexpand\write\@auxout{%
4276           \string\providecommand\string\@gls@codepage[2]{}}%
4277         \noexpand\immediate\noexpand\write\@auxout{%
4278           \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4279       }%
4280     }%
4281     \@do@auxoutstuff
4282   \fi
4283   \renewcommand*{\@warn@nomakeglossaries}{%
4284     \GlossariesWarningNoLine{\string\makeglossaries\space
4285       hasn't been used,^^Jthe glossaries will not be updated}%

```

```

4286 }%
4287 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4288 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4289   This document is incomplete. The external file associated with
4290   the glossary ‘#1’ (which should be called \texttt{#2})
4291   hasn’t been created.%
4292 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4293 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4294   This has probably happened because there are no entries defined
4295   in this glossary.%
4296 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4297 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4298   If you don’t want this glossary,
4299   add \texttt{nomain} to your package option list when you load
4300   \texttt{glossaries-extra.sty}. For example:%
4301 }

```

`\GlsWarningEmptyNotMain` A glossary that isn’t the default “main” glossary is empty.

```

4302 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4303   Did you forget to use \texttt{type=#1} when you defined your
4304   entries? If you tried to load entries into this glossary with
4305   \texttt{\string\loadglsentries} did you remember to use
4306   \texttt{[#1]} as the optional argument? If you did, check that
4307   the definitions in the file you loaded all had the type set
4308   to \texttt{\string\glsdefaulttype}.%
4309 }

```

`\GlsWarningCheckFile` Advisory message to check the file contents.

```

4310 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4311   Check the contents of the file \texttt{#1}. If
4312   it’s empty, that means you haven’t indexed any of your entries in this
4313   glossary (using commands like \texttt{\string\gls} or
4314   \texttt{\string\glsadd}) so this list can’t be generated.
4315   If the file isn’t empty, the document build process hasn’t been
4316   completed.%
4317 }

```

`\GlsWarningAutoMake` Message when automake option has been used.

```

4318 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4319   You may need to rerun \LaTeX. If you already have, it may be that
4320   \TeX’s shell escape doesn’t allow you to run

```

```

4321 \ifglxindy xindy\else makeindex\fi. Check the
4322 transcript file \texttt{\jobname.log}. If the shell escape is
4323 disabled, try one of the following:
4324
4325 \begin{itemize}
4326   \item Run the external (Lua) application:
4327
4328       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4329
4330   \item Run the external (Perl) application:
4331
4332       \texttt{makeglossaries \string"\jobname\string"}
4333 \end{itemize}
4334
4335 Then rerun \LaTeX\ on this document.
4336 \GlossariesExtraWarning{Rerun required to build the
4337 glossary ‘#1’ or check TeX’s shell escape allows
4338 you to run \ifglxindy xindy\else makeindex\fi}%
4339 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4340 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4341   You need to either replace \texttt{\string\makenoidxglossaries}
4342   with \texttt{\string\makeglossaries} or replace
4343   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4344   \texttt{\string\printnoidxglossary}
4345   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4346   this document.%
4347 }

```

WarningBuildInfo Build advice.

```

4348 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4349   Try one of the following:
4350   \begin{itemize}
4351     \item Add \texttt{automake} to your package option list when you load
4352           \texttt{glossaries-extra.sty}. For example:
4353
4354           \texttt{\string\usepackage[automake]%
4355                   \glsopenbrace glossaries-extra\glsclosebrace}
4356
4357     \item Run the external (Lua) application:
4358
4359           \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4360
4361     \item Run the external (Perl) application:
4362
4363           \texttt{makeglossaries \string"\jobname\string"}
4364   \end{itemize}

```



```

4365
4366 Then rerun \LaTeX\ on this document.%
4367 }

```

trRecordWarning Paragraph for record=only.

```

4368 \newcommand{\GlsXtrRecordWarning}[1]{%
4369 \texttt{\string\printglossary} doesn't work
4370 with the \texttt{record=only} package option
4371 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4372 instead (or change the package option).%
4373 }

```

oGlsWarningTail Final paragraph.

```

4374 \newcommand{\GlsXtrNoGlsWarningTail}{%
4375 This message will be removed once the problem has been fixed.%
4376 }

```

GlsWarningNoOut No out file created. Build advice.

```

4377 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4378 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4379 \texttt{\string\makeglossaries} or you have used
4380 \texttt{\string\nofiles}. If this is just a draft version of the
4381 document, you can suppress this message using the
4382 \texttt{nomissingglstext} package option.%
4383 }

```

glossarywarning

```

4384 \newcommand*{@@glsxtr@defaultnoglossarywarning}[1]{%
4385 \glossarysection[\glossarytoctitle]{\glossarytitle}
4386 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glo@type @\@glo@type @in\endcsname}
4387 \par
4388 \glsxtrifemptyglossary{#1}%
4389 {%
4390 \GlsXtrNoGlsWarningEmptyStart\space
4391 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4392 \medskip
4393 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]}%
4394 \glsopenbrace glossaries-extra\glsclosebrace}
4395 \medskip
4396 }%
4397 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4398 }%
4399 {%
4400 \IfFileExists{\jobname.\csname @glo@type @\@glo@type @out\endcsname}
4401 {%
4402 \GlsXtrNoGlsWarningCheckFile
4403 {\jobname.\csname @glo@type @\@glo@type @out\endcsname}
4404
4405 \ifglsautomake

```

```

4406
4407     \GlsXtrNoGlsWarningAutoMake{#1}
4408
4409     \else
4410
4411         \ifthenelse{\equal{#1}{main}}{%
4412             {%
4413                 \GlsXtrNoGlsWarningEmptyMain\par
4414                 \medskip
4415                 \noindent\texttt{\string\usepackage[nomain]%
4416                     \glsopenbrace glossaries-extra\glsclosebrace}
4417                 \medskip
4418             }%
4419         }%
4420
4421         \ifdefequal\makeglossaries\@no@makeglossaries
4422             {%
4423                 \GlsXtrNoGlsWarningMisMatch
4424             }%
4425             {%
4426                 \GlsXtrNoGlsWarningBuildInfo
4427             }%
4428         \fi
4429     }%
4430     {%
4431         \GlsXtrNoGlsWarningNoOut
4432         {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4433     }%
4434 }%
4435 \par
4436 \GlsXtrNoGlsWarningTail
4437 }

```

glossarywarning Warn about using `\printglossary` with `record`

```

4438 \newcommand*{\@glxtr@record@noglossarywarning}[1]{%
4439     \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4440     with record=only package option\MessageBreak(use
4441     \string\printunsrtglossary[type=#1])\MessageBreak
4442     instead (or change the package option)}%
4443     \glossarysection[\glossarytoctitle]{\glossarytitle}
4444     \GlsXtrRecordWarning{#1}
4445     \GlsXtrNoGlsWarningTail
4446 }

```

Provide some commands to accompany the `record` option for use with **bib2gls**.

xtresourcefile 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.

```

4447 \newcommand*{\glxtrresourcefile}[2] [] {%

```

The record option can't be set after this command.

```

4448 \disable@keys{glossaries-extra.sty}{record}%
4449 \glxtr@writefields
4450 \protected@write\auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%
4451 \let\@glxtr@org@see@noindex\@glxtr@see@noindex
4452 \let\@glxtr@see@noindex\relax
4453 \IfFileExists{#2.glstex}%
4454 {%

```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```

4455 \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4456 \makeatletter
4457 \@input{#2.glstex}%
4458 \@bibgls@restreat
4459 }%
4460 {%
4461 \GlossariesExtraWarning{No file '#2.glstex'}%
4462 }%
4463 \let\@glxtr@org@see@noindex\@glxtr@org@see@noindex
4464 }
4465 \@onlypreamble\glxtrresourcefile

```

glxtrresourceinit Code used during the protected write operation.

```

4466 \newcommand*\glxtrresourceinit{}

```

glxtrresourcecount

```

4467 \newcount\glxtrresourcecount

```

glxtrLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```

4468 \newcommand*\GlsXtrLoadResources[1][]{%
4469 \ifnum\glxtrresourcecount=0\relax
4470 \glxtrresourcefile[#1]{\jobname}%
4471 \else
4472 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4473 \fi
4474 \advance\glxtrresourcecount by 1\relax
4475 }

```

glxtr@resource

```

4476 \newcommand*\glxtr@resource[2]{}

```

\glxtr@fields

```

4477 \newcommand*\glxtr@fields[1]{}

```

glxtr@texencoding

```

4478 \newcommand*\glxtr@texencoding[1]{}

```

\glsxtr@langtag

```
4479 \newcommand*{\glsxtr@langtag}[1]{}
```

@pluralsuffixes

```
4480 \newcommand*{\glsxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
4481 \newcommand*{\glsxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
4482 \newcommand*{\glsxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
4483 \newcommand*{\glsxtr@writefields}{%
```

```
4484   \protected@write\@auxout{}%
```

```
4485     {\string\providecommand*{\string\glsxtr@fields}[1]{}}%
```

```
4486   \protected@write\@auxout{}%
```

```
4487     {\string\providecommand*{\string\glsxtr@resource}[2]{}}%
```

```
4488   \protected@write\@auxout{}%
```

```
4489     {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}}%
```

```
4490   \protected@write\@auxout{}%
```

```
4491     {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}}%
```

```
4492   \protected@write\@auxout{}%
```

```
4493     {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}}%
```

```
4494   \protected@write\@auxout{{\string\glsxtr@fields{\@gls@keymap}}}%
```

```
4495   \protected@write\@auxout{}%
```

```
4496     {\string\providecommand*{\string\glsxtr@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 \glsxtrresourcefile.

```
4497   \ifdef\CurrentTrackedLanguageTag
```

```
4498     {%
```

```
4499       \protected@write\@auxout{}{%
```

```
4500         \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
```

```
4501     }%
```

```
4502   }%
```

```
4503   \protected@write\@auxout{{\string\glsxtr@pluralsuffixes
```

```
4504     {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
```

```
4505     {\glsxtrabbrvpluralsuffix}}%
```

```
4506   \ifdef\inputencodingname
```

```
4507     {%
```

```
4508       \protected@write\@auxout{{\string\glsxtr@texencoding{\inputencodingname}}}%
```

```
4509     }%
```

```
4510   }%
```

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.)

```

4511 \ifpackageloaded{fontspec}%
4512 {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}}%
4513 {}%
4514 }%
4515 \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.

```

4516 \AtBeginDocument
4517 {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}}%
4518 \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.

```

4519 \ifglautomake
4520 \IfFileExists{\jobname.aux}%
4521 {\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.

```

4522 \ifx\@glx@doautomake\@glx@doautomake@err
4523 \let\@glx@doautomake\relax
4524 \fi
4525 \fi
4526 }

```

do@automake@err

```

4527 \newcommand*\@glx@doautomake@err{%
4528 \PackageError{glossaries}{You must use
4529 \string\makeglossaries\space with automake=true}
4530 {%
4531 Either remove the automake=true setting or
4532 add \string\makeglossaries\space to your document preamble.%
4533 }%
4534 }

```

Allow locations specific to a particular counter to be recorded.

\glxtr@record

```

4535 \newcommand*\glxtr@record}[5]{%

```

r@counterrecord Aux file command.

```

4536 \newcommand*\glxtr@counterrecord}[3]{%
4537 \glxtrfieldlistgadd{#1}{record.#2}{#3}%
4538 }

```

counterrecordhook Hook used by \@glxtr@dorecord.

```
4539 \newcommand*{\@glxtr@counterrecordhook}{}
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4540 \newcommand*{\GlsXtrRecordCounter}[1]{%
```

```
4541   \@glxtr@recordcounter{#1}%
```

```
4542 }
```

```
4543 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4544 \newcommand*{\@glxtr@docounterrecord}[1]{%
```

```
4545   \protected@write\@auxout{}{\string\glxtr@counterrecord
```

```
4546     {\@glslabel}{#1}{\csuse{the#1}}}%
```

```
4547 }
```

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.

```
4548 \newcommand*{\glxtrglossentry}[1]{%
```

```
4549   \glxtrtitleorpdforheading
```

```
4550   {\@glxtrglossentry{#1}}%
```

```
4551   {\glsentryname{#1}}%
```

```
4552   {\glxtrheadname{#1}}%
```

```
4553 }
```

lsxtrglossentry Another test is needed in case \@glxtrglossentry has been written to the table of contents.

```
4554 \newrobustcmd*{\@glxtrglossentry}[1]{%
```

```
4555   \glxtrtitleorpdforheading
```

```
4556   {%
```

```
4557     \glsoifexists{#1}%
```

```
4558     {%
```

```
4559       \begingroup
```

```
4560         \edef\glscurrententrylabel{\glsetoklabel{#1}}%
```

```
4561         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
```

```
4562         \ifglshasparent{#1}%
```

```
4563         {\GlsXtrStandaloneSubEntryItem{#1}}%
```

```
4564         {\glsentryitem{#1}}%
```

```
4565         \glstarget{#1}{\glossentryname{#1}}%
```

```
4566       \endgroup
```

```
4567     }%
```

```
4568   }%
```

```
4569   {\glsentryname{#1}}%
```

```
4570   {\glxtrheadname{#1}}%
```

```
4571 }
```

`\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.)

```
4572 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsenentrytype{\glscurrententrylabel}}
```

`\oneSubEntryItem` Used for sub-entries in standalone format. The argument is the entry's label.

```
4573 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4574   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
4575 }
```

`\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.

```
4576 \newcommand*{\glxstrglossentryother}[3]{%
4577   \ifstrempy{#1}%
4578   {%
4579     \ifcsdef{glxstrhead#3}%
4580     {%
4581       \glxstrtitleorpdforheading
4582       {\@glxstrglossentryother{#2}{#3}{#1}}%
4583       {\@gls@entry@field{#2}{#3}}%
4584       {\csuse{glxstrhead#3}{#2}}%
4585     }%
4586     {%
4587       \glxstrtitleorpdforheading
4588       {\@glxstrglossentryother{#2}{#3}{#1}}%
4589       {\@gls@entry@field{#2}{#3}}%
4590       {\@gls@entry@field{NoCaseChange{#2}}{#3}}%
4591     }%
4592   }%
4593   {%
4594     \glxstrtitleorpdforheading
4595     {\@glxstrglossentryother{#2}{#3}{#1}}%
4596     {\@gls@entry@field{#2}{#3}}%
4597     {#1}%
4598   }%
4599 }
```

`\glossentryother` As `\@glxstrglossentry` but uses a different field.

```
4600 \newrobustcmd*{\@glxstrglossentryother}[3]{%
4601   \glxstrtitleorpdforheading
4602   {%
4603     \glsdoifexists{#1}%
4604     {%
4605       \begingroup
4606       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4607       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%

```

```

4608     \ifglshasparent{#1}%
4609     {\GlsXtrStandaloneSubEntryItem{#1}}}%
4610     {\glsentryitem{#1}}}%
4611     \glstarget{#1}{\glossentrynameother{#1}{#2}}}%
4612   \endgroup
4613 }%
4614 }%
4615 {\@gls@entry@field{#1}{#2}}}%
4616 {#3}}%
4617 }

```

`\printunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4618 \newcommand*{\printunsrtglossary}{%
4619   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4620 }

```

`\printunsrtglossary` Unstarred version.

```

4621 \newcommand*{\@printunsrtglossary}[1][ ]{%
4622   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4623 }

```

`\printunsrtglossary` Starred version.

```

4624 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4625   \begingroup
4626     #2%
4627     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4628   \endgroup
4629 }

```

`\printunsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```

4630 \newcommand*{\printunsrtglossaries}{%
4631   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4632 }

```

`\@print@glossary`

```

4633 \newcommand*{\@print@glossary}{%
4634   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4635   \glossarypreamble
4636   check for empty list
4636   \glstrifemptyglossary{\@glo@type}%
4637   {%
4638     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4639   }%
4640   {%

```



```

4641 \key@ifundefined{glossentry}{group}%
4642 {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4643 {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4644 \def\@gls@currentlettergroup{}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

4645 \def\@glsxtr@doglossary{%
4646   \begin{theglossary}%
4647   \glossaryheader
4648   \glsresetentrylist
4649 }%
4650 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4651   :\expandafter=\csname glo@list@\@glo@type\endcsname\do{%
4652   \ifdefempty{\glscurrententrylabel}
4653   }{%
4654   {%

```

Provide a hook (for example to measure width).

```

4655 \let\glsxtr@process\@firstofone
4656 \let\printunsrtglossaryskipentry
4657 \let\glsxtr@printunsrtglossaryskipentry
4658 \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

4659 \glsxtr@process
4660 {%
4661   \ifglshasparent{\glscurrententrylabel}{}%
4662   {%
4663     \@glsxtr@checkgroup\glscurrententrylabel
4664     \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
4665       {\@glsxtr@groupheading}%
4666   }%
4667   \eappto\@glsxtr@doglossary{%
4668     \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4669   }%
4670 }%
4671 }%
4672 \appto\@glsxtr@doglossary{\end{theglossary}}%
4673 \printunsrtglossarypredoglossary
4674 \@glsxtr@doglossary
4675 }%
4676 \glossarypostamble
4677 }

```

entryprocesshook

```

4678 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{%

```

ossaryskipentry

```

4679 \newcommand*{\printunsrtglossaryskipentry}{%
4680 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space

```

```

4681 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4682 }

```

entryprocesshook

```

4683 \newcommand*{\@glstr@printunsrtglossaryskipentry}{%
4684   \let\glstr@process\@gobble
4685 }

```

rypredoglossary

```

4686 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```

4687 \newcommand{\@printunsrt@glossary@handler}[1]{%
4688   \xdef\glscurrententrylabel{#1}%
4689   \printunsrtglossaryhandler\glscurrententrylabel
4690 }

```

glossaryhandler

```

4691 \newcommand{\printunsrtglossaryhandler}[1]{%
4692   \glstrunsrtdo{#1}%
4693 }

```

striflabelinlist

```
\glstriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of \@gls@ifinlist which ensures the label and list are fully expanded.

```

4694 \newrobustcmd*{\glstriflabelinlist}[4]{%
4695   \protected@edef\@glstr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
4696   \@glstr@doiflabelinlist{#3}{#4}%
4697 }

```

srtglossaryunit

```

4698 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
4699   \s@printunsrtglossary[type=\glstypetype,#1]{%
4700     \printunsrtglossaryunitsetup{#2}%
4701   }%
4702 }

```

ossaryunitsetup

```

4703 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4704   \renewcommand{\printunsrtglossaryhandler}[1]{%
4705     \glstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4706     {\glstrunsrtdo{##1}}%
4707   }%
4708 }%

```

Only the target names should have the prefixes adjusted as \gls etc need the original \glslinkprefix. The \@gobble part discards \glslinkprefix.

```

4709 \ifcsundef{theH#1}%
4710 {%
4711 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4712 }%
4713 {%
4714 \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4715 }%
4716 \renewcommand*{\glossarysection}[2][{}]{%
4717 \appto\glossarypostamble{\glspar\medskip\glspar}%
4718 }

```

unsrtglossaryunit

```

4719 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
4720 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4721 requires the record=only or record=alsoindex package option}{}%
4722 }

```

t@getgrouptitle

```

4723 \newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
4724 \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4725 \@onelevel@sanitize\@glsxtr@titlelabel
4726 \ifcsdef{\@glsxtr@titlelabel}
4727 {\letcs{#2}{\@glsxtr@titlelabel}}%
4728 {\def#2{#1}}%
4729 }

```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```

4730 \newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}

```

lsxtrgroupfield bib2gls provides a supplementary field labelled secondarygroup for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```

4731 \newcommand*{\glsxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glsxtr@doglossary is being constructed rather than in the handler.

lsxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glsxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glsxtr@groupheading, which will be empty if no heading is required.

```

4732 \newcommand*{\@glsxtr@checkgroup}[1]{%
4733 \def\@glsxtr@groupheading{}%
4734 \key@ifundefined{glossentry}{group}%
4735 {%

```

```

4736 \letcs{\@gls@sort}{glo@glstetoklabel{#1}@sort}%
4737 \expandafter\glo@grabfirst\@gls@sort{}\@nil
4738 }%
4739 {%

4740 \protected@edef\@glo@thislettergrp{%
4741 \csuse{glo@glstetoklabel{#1}@glstxtrgroupfield}}%
4742 }%
4743 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4744 {}%
4745 {%
4746 \ifdefempty{\@gls@currentlettergroup}{}%
4747 {\def\@glstxtr@groupheading{\glsgroupskip}}%
4748 \eappto\@glstxtr@groupheading{%
4749 \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
4750 }%
4751 }%
4752 \let\@gls@currentlettergroup\@glo@thislettergrp
4753 }

```

glstxtr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present, but also need to check for the group field.

```

4754 \newcommand{\@glstxtr@noidx@do}[1]{%
4755 \ifglstentryexists{#1}%
4756 {%
4757 \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
4758 \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@location}%
4759 \ifglshasparent{#1}%
4760 {%
4761 \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
4762 \ifdefvoid{\@gls@location}%
4763 {%
4764 \ifdefvoid{\@gls@loclist}%
4765 {%
4766 \subglossentry{\gls@level}{#1}{}%
4767 }%
4768 {%
4769 \subglossentry{\gls@level}{#1}%
4770 {%
4771 \glossaryentrynumbers{\glstnoidxloclist{\@gls@loclist}}%
4772 }%
4773 }%
4774 }%
4775 {%
4776 \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4777 }%
4778 }%
4779 {%
4780 \ifdefvoid{\@gls@location}%

```

```

4781      {%
4782      \ifdefvoid{\@gls@loclist}
4783      {%
4784      \glossentry{#1}{}%
4785      }%
4786      {%
4787      \glossentry{#1}%
4788      {%
4789      \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4790      }%
4791      }%
4792      }%
4793      {%
4794      \glossentry{#1}%
4795      {%
4796      \glossaryentrynumbers{\@gls@location}%
4797      }%
4798      }%
4799      }%
4800      }%
4801      {}%
4802      }

```

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`

```
4803 \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>}
```

```

4804 \newcommand*{\@glsxtrnewgls}[4]{%
4805   \ifdef{#3}%
4806   {%
4807     \PackageError{glossaries-extra}{Command \string#3\space already
4808 defined}{}%
4809   }%
4810   {%
4811     \ifcsdef{@#4like@#2}%
4812     {%
4813       \advance\@glsxtrnewgls@inner by \@ne

```

```

4814 \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4815 }%
4816 {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4817 \expandafter\newrobustcmd\expandafter*\expandafter
4818 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4819 \ifstrempy{#1}%
4820 {%
4821 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2] [] {%
4822 \new@ifnextchar [%
4823 {\csname @#4@\endcsname{##1}{#2##2}}%
4824 {\csname @#4@\endcsname{##1}{#2##2} []}%
4825 }%
4826 }%
4827 {%
4828 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2] [] {%
4829 \new@ifnextchar [%
4830 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4831 {\csname @#4@\endcsname{#1,##1}{#2##2} []}%
4832 }%
4833 }%
4834 }%
4835 }

```

`\glsxtrnewgls` `\glsxtrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4836 \newrobustcmd*{\glsxtrnewgls}[3] [] {%
4837 \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4838 }

```

`\glsxtrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4839 \newrobustcmd*{\glsxtrnewglslike}[6] [] {%
4840 \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4841 \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
4842 \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
4843 \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
4844 }

```

`\glsxtrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4845 \newrobustcmd*{\glsxtrnewGLSlike}[4] [] {%
4846 \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
4847 \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
4848 }

```

`\glxtrnewrgls` As `\glxtrnewrgls` but for `\rgls`.

```
4849 \newrobustcmd*{\glxtrnewrgls}[3] [] {%
4850   \@glxtrnewrgls{#1}{#2}{#3}{rgls}%
4851 }
```

`sxtrnewrglslike` As `\glxtrnewrglslike` but for `\rgls` etc.

```
4852 \newrobustcmd*{\glxtrnewrglslike}[6] [] {%
4853   \@glxtrnewrgls{#1}{#2}{#3}{rgls}%
4854   \@glxtrnewrgls{#1}{#2}{#4}{rglsp1}%
4855   \@glxtrnewrgls{#1}{#2}{#5}{rGls}%
4856   \@glxtrnewrgls{#1}{#2}{#6}{rGlspl}%
4857 }
```

`sxtrnewrGLSlike` As `\glxtrnewrGLSlike` but for `\rGLS` etc.

```
4858 \newrobustcmd*{\glxtrnewrGLSlike}[4] [] {%
4859   \@glxtrnewrgls{#1}{#2}{#3}{rGLS}%
4860   \@glxtrnewrgls{#1}{#2}{#4}{rGLSp1}%
4861 }
```

Provide easy access to record count fields.

`totalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```
4862 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
4863   \ifcsdef{glo@\glsetoklabel{#1}@recordcount}%
4864   {\csname glo@\glsetoklabel{#1}@recordcount\endcsname}%
4865   {0}%
4866 }
```

`sXtrRecordCount` Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4867 \newcommand*{\GlsXtrRecordCount}[2] {%
4868   \ifcsdef{glo@\glsetoklabel{#1}@recordcount.#2}%
4869   {\csname glo@\glsetoklabel{#1}@recordcount.#2\endcsname}%
4870   {0}%
4871 }
```

`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 `\glxtrdetoklocation` can be set to something sensible.

```
4872 \newcommand*{\GlsXtrLocationRecordCount}[3] {%
4873   \ifcsdef{glo@\glsetoklabel{#1}@recordcount.#2.\glxtrdetoklocation{#3}}%
4874   {\csname glo@\glsetoklabel{#1}@recordcount.#2.\glxtrdetoklocation{#3}\endcsname}%
4875   {0}%
4876 }
```

`trdetoklocation`

```
4877 \newcommand*{\glxtrdetoklocation}[1]{#1}
```

ablerecordcount

```

4878 \newcommand*{\glxtrenablerecordcount}{%
4879   \renewcommand*{\gls}{\rgls}%
4880   \renewcommand*{\Gls}{\rGls}%
4881   \renewcommand*{\glspl}{\rglspl}%
4882   \renewcommand*{\Glspl}{\rGlspl}%
4883   \renewcommand*{\GLS}{\rGLS}%
4884   \renewcommand*{\GLSpl}{\rGLSpl}%
4885 }

```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```

4886 \newcommand*{\glxtrrecordtriggervalue}[1]{%
4887   \GlsXtrTotalRecordCount{#1}%
4888 }

```

dCountAttribute

```

4889 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4890   \@for\@glxtr@cat:=#1\do
4891   {%
4892     \ifdefempty{\@glxtr@cat}{}%
4893     {%
4894       \glssetcategoryattribute{\@glxtr@cat}{recordcount}{#2}%
4895     }%
4896   }%
4897 }

```

rifrecordtrigger

<code>\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}</code>

```

4898 \newcommand*{\glxtrifrecordtrigger}[3]{%
4899   \glshasattribute{#1}{recordcount}%
4900   {%
4901     \ifnum\glxtrrecordtriggervalue{#1}>\glsgetattribute{#1}{recordcount}\relax
4902     #3%
4903   \else
4904     #2%
4905   \fi
4906 }%
4907 {#3}%
4908 }

```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```

4909 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
4910   \edef\glslabel{\glsdetoklabel{#2}}%
4911   \let\@gls@link@label\glslabel
4912   \def\@glxtr@thevalue{%

```



```

4913 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4914 \def\@glsnumberformat{glstriggerrecordformat}%
4915 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
4916 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
4917 \def\@glsxtr@thevalue{%
4918 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4919 \glsxtrinitwrgloss
4920 \glslinkpresetkeys
4921 \setkeys{glslink}{#1}%
4922 \glslinkpostsetkeys
4923 \ifdefempty{\@glsxtr@thevalue}%
4924 {%
4925 \@gls@saveentrycounter
4926 }%
4927 {%
4928 \let\theglsentrycounter\@glsxtr@thevalue
4929 \def\theHglentrycounter{\@glsxtr@theHvalue}%
4930 }%
4931 \ifglsxtrinitwrglossbefore
4932 \do@wrglossary{#2}%
4933 \fi
4934 #3%
4935 \ifglsxtrinitwrglossbefore
4936 \else
4937 \do@wrglossary{#2}%
4938 \fi
4939 \ifKV@glslink@local
4940 \glslocalunset{#2}%
4941 \else
4942 \glsunset{#2}%
4943 \fi
4944 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

4945 \newcommand*{\glstriggerrecordformat}[1]{

```

\rgls

```

4946 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

4947 \newcommand*{\@rgls}[2][{}]{%
4948 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}}[{}]}%
4949 }

```

\@rgls@

```

4950 \def\@rgls@#1#2[#3]{%
4951 \glsxtrifrecordtrigger{#2}%
4952 {%

```

```

4953 \@glxstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
4954 }%
4955 {%
4956 \@gls@{#1}{#2}[#3]%
4957 }%
4958 }%

```

\rglspl

```

4959 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

```

\@rglspl

```

4960 \newcommand*{\@rglspl}[2][{}]{%
4961 \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
4962 }

```

\@rglspl@

```

4963 \def\@rglspl@#1#2[#3]{%
4964 \glxtrifrecordtrigger{#2}%
4965 {%
4966 \@glxstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
4967 }%
4968 {%
4969 \@glspl@{#1}{#2}[#3]%
4970 }%
4971 }%

```

\rGls

```

4972 \newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}

```

\@rGls

```

4973 \newcommand*{\@rGls}[2][{}]{%
4974 \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
4975 }

```

\@rGls@

```

4976 \def\@rGls@#1#2[#3]{%
4977 \glxtrifrecordtrigger{#2}%
4978 {%
4979 \@glxstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
4980 }%
4981 {%
4982 \@Gls@{#1}{#2}[#3]%
4983 }%
4984 }%

```

\rGlspl

```

4985 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\@rGlspl}

```

```

\@rGlspl
4986 \newcommand*{\@rGlspl}[2] [] {%
4987   \new@ifnextchar [{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2} []}%
4988 }

\@rGlspl@
4989 \def\@rGlspl@#1#2[#3] {%
4990   \glstrifrecordtrigger{#2}%
4991   {%
4992     \@glstr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
4993   }%
4994   {%
4995     \@Glspl@{#1}{#2}[#3]%
4996   }%
4997 }%

\rGLS
4998 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

\@rGLS
4999 \newcommand*{\@rGLS}[2] [] {%
5000   \new@ifnextchar [{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2} []}%
5001 }

\@rGLS@
5002 \def\@rGLS@#1#2[#3] {%
5003   \glstrifrecordtrigger{#2}%
5004   {%
5005     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
5006   }%
5007   {%
5008     \@GLS@{#1}{#2}[#3]%
5009   }%
5010 }%

\rGLSpl
5011 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

\@rGLSpl
5012 \newcommand*{\@rGLSpl}[2] [] {%
5013   \new@ifnextchar [{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2} []}%
5014 }

\@rGLSpl@
5015 \def\@rGLSpl@#1#2[#3] {%
5016   \glstrifrecordtrigger{#2}%
5017   {%
5018     \@glstr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%

```

```

5019 }%
5020 {%
5021   \@GLSpl@{#1}{#2}[#3]%
5022 }%
5023 }%

```

`\rglsformat`

```

5024 \newcommand*{\rglsformat}[2]{%
5025   \glsifregular{#1}
5026   {\glsentryfirst{#1}}%
5027   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5028 }

```

`\rglsplformat`

```

5029 \newcommand*{\rglsplformat}[2]{%
5030   \glsifregular{#1}
5031   {\glsentryfirstplural{#1}}%
5032   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5033 }

```

`\rGlsformat`

```

5034 \newcommand*{\rGlsformat}[2]{%
5035   \glsifregular{#1}
5036   {\Glsentryfirst{#1}}%
5037   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5038 }

```

`\rGlsplformat`

```

5039 \newcommand*{\rGlsplformat}[2]{%
5040   \glsifregular{#1}
5041   {\Glsentryfirstplural{#1}}%
5042   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5043 }

```

`\rGLSformat`

```

5044 \newcommand*{\rGLSformat}[2]{%
5045   \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
5046 }

```

`\rGLSplformat`

```

5047 \newcommand*{\rGLSplformat}[2]{%
5048   \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
5049 }

```

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

`\glxtr@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`.)

`o@inc@linkcount` This performs the actual incrementing and counter definition. The counter is given by `\c@glxtr@linkcount@<label>` where `<label>` is the entry’s label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```
5050 \newcommand{\@glxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
5051 \glsifattribute{\glslabel}{linkcount}{true}%
```

```
5052 {%
```

Does the counter exist?

```
5053 \ifcsdef{c@glxtr@linkcount@\glslabel}{}%
```

```
5054 {%
```

Counter doesn’t exist, so define it.

```
5055 \newcounter{glxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
5056 \glshasattribute{\glslabel}{linkcountmaster}%
```

```
5057 {%
```

Need to ensure values are fully expanded.

```
5058 \begingroup
```

```
5059 \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@\glslabel}%
```

```
5060 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
```

```
5061 \x
```

```
5062 }%
```

```
5063 {}%
```

```
5064 }%
```

Increment counter:

```
5065 \glxtrinlinkcounter{glxtr@linkcount@\glslabel}%
```

```
5066 }%
```

```
5067 {}%
```

```
5068 }
```

`rlinkcounter` May be redefined to use `\refstepcounter` if required.

```
5069 \newcommand*{\glxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
5070 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
```

```
5071 \ifcsundef{c@glxtr@linkcount@#1}{0}{\csname c@glxtr@linkcount@#1\endcsname}%
```

```
5072 }
```

`rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5073 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
```

```
5074 \ifcsundef{theglsxtr@linkcount@#1}{0}%
```

```
5075 {\csname theglxtr@linkcount@#1\endcsname}%
```

```
5076 }
```

`fLinkCounterDef` Tests if the counter has been defined

```
5077 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5078   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5079 }
```

`LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
5080 \newcommand*{\GlsXtrLinkCounterName}[1]{glsextr@linkcount@#1}
```

`ableLinkCounting` `\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
5081 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5082   \let\glsextr@inc@linkcount\glsextr@do@inc@linkcount
5083   \@for\glsextr@label:=#2\do
5084   {%
5085     \glsssetcategoryattribute{\glsextr@label}{linkcount}{true}%
5086     \ifstrempy{#1}{}%
5087     {%
5088       \ifcsundef{c@#1}%
5089       {\@nocounterr{#1}}%
5090       {\glsssetcategoryattribute{\glsextr@label}{linkcountmaster}{#1}}%
5091     }%
5092   }%
5093 }
5094 \@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.

```
5095 \@ifpackageloaded{glossaries-accsupp}
5096 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5097   \newcommand*{\glsaccessname}[1]{%
5098     \glsnameaccessdisplay
5099     {%
5100       \glsentryname{#1}%
5101     }%
5102     {#1}%
5103   }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

5104 \newcommand*{\Glsaccessname}[1]{%
5105   \glsnameaccessdisplay
5106   {%
5107     \Glsentryname{#1}%
5108   }%
5109   {#1}%
5110 }

```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```

5111 \newcommand*{\GLSaccessname}[1]{%
5112   \glsnameaccessdisplay
5113   {%
5114     \mfirstucMakeUppercase{\glsentryname{#1}}%
5115   }%
5116   {#1}%
5117 }

```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

5118 \newcommand*{\glsaccesstext}[1]{%
5119   \glstextaccessdisplay
5120   {%
5121     \glsentrytext{#1}%
5122   }%
5123   {#1}%
5124 }

```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

5125 \newcommand*{\Glsaccesstext}[1]{%
5126   \glstextaccessdisplay
5127   {%
5128     \Glsentrytext{#1}%
5129   }%
5130   {#1}%
5131 }

```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5132 \newcommand*{\GLSaccesstext}[1]{%
5133   \glstextaccessdisplay
5134   {%
5135     \mfirstucMakeUppercase{\glsentrytext{#1}}%
5136   }%
5137   {#1}%
5138 }

```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

5139 \newcommand*{\glsaccessplural}[1]{%
5140   \glspluralaccessdisplay
5141   {%
5142     \glsentryplural{#1}%
5143   }%
5144   {#1}%
5145 }

```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5146 \newcommand*{\Glsaccessplural}[1]{%
5147   \glspluralaccessdisplay
5148   {%
5149     \Glsentryplural{#1}%
5150   }%
5151   {#1}%
5152 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

5153 \newcommand*{\GLSaccessplural}[1]{%
5154   \glspluralaccessdisplay
5155   {%
5156     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5157   }%
5158   {#1}%
5159 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

5160 \newcommand*{\glsaccessfirst}[1]{%
5161   \glsfirstaccessdisplay
5162   {%
5163     \glsentryfirst{#1}%
5164   }%
5165   {#1}%
5166 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5167 \newcommand*{\Glsaccessfirst}[1]{%
5168   \glsfirstaccessdisplay
5169   {%
5170     \Glsentryfirst{#1}%
5171   }%
5172   {#1}%
5173 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

5174 \newcommand*{\GLSaccessfirst}[1]{%

```



```

5175 \glsfirstaccessdisplay
5176 {%
5177 \mfirstucMakeUppercase{\glstentryfirst{#1}}%
5178 }%
5179 {#1}%
5180 }

```

`\glsfirstplural` Display the firstplural value (no link and no check for existence).

```

5181 \newcommand*{\glsaccessfirstplural}[1]{%
5182 \glsfirstpluralaccessdisplay
5183 {%
5184 \glstentryfirstplural{#1}%
5185 }%
5186 {#1}%
5187 }

```

`\Glsfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5188 \newcommand*{\Glsaccessfirstplural}[1]{%
5189 \glsfirstpluralaccessdisplay
5190 {%
5191 \Glstentryfirstplural{#1}%
5192 }%
5193 {#1}%
5194 }

```

`\GLSfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5195 \newcommand*{\GLSaccessfirstplural}[1]{%
5196 \glsfirstpluralaccessdisplay
5197 {%
5198 \mfirstucMakeUppercase{\glstentryfirstplural{#1}}%
5199 }%
5200 {#1}%
5201 }

```

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

5202 \newcommand*{\glsaccesssymbol}[1]{%
5203 \glssymbolaccessdisplay
5204 {%
5205 \glstentrysymbol{#1}%
5206 }%
5207 {#1}%
5208 }

```

`\Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5209 \newcommand*{\Glsaccesssymbol}[1]{%
5210 \glssymbolaccessdisplay

```

```

5211     {%
5212         \Glsentrysymbol{#1}%
5213     }%
5214     {#1}%
5215 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5216 \newcommand*{\GLSaccesssymbol}[1]{%
5217     \glssymbolaccessdisplay
5218     {%
5219         \mfirstucMakeUppercase{\glentrysymbol{#1}}%
5220     }%
5221     {#1}%
5222 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5223 \newcommand*{\glaccesssymbolplural}[1]{%
5224     \glssymbolpluralaccessdisplay
5225     {%
5226         \glentrysymbolplural{#1}%
5227     }%
5228     {#1}%
5229 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5230 \newcommand*{\Glsaccesssymbolplural}[1]{%
5231     \glssymbolpluralaccessdisplay
5232     {%
5233         \Glsentrysymbolplural{#1}%
5234     }%
5235     {#1}%
5236 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5237 \newcommand*{\GLSaccesssymbolplural}[1]{%
5238     \glssymbolpluralaccessdisplay
5239     {%
5240         \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5241     }%
5242     {#1}%
5243 }

```

\glsaccessdesc Display the desc value (no link and no check for existence).

```

5244 \newcommand*{\glsaccessdesc}[1]{%
5245     \glsdescriptionaccessdisplay
5246     {%
5247         \glentrydesc{#1}%

```

```

5248 }%
5249 {#1}%
5250 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5251 \newcommand*{\Glsaccessdesc}[1]{%
5252   \glsdescriptionaccessdisplay
5253   {%
5254     \Glsentrydesc{#1}%
5255   }%
5256   {#1}%
5257 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5258 \newcommand*{\GLSaccessdesc}[1]{%
5259   \glsdescriptionaccessdisplay
5260   {%
5261     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5262   }%
5263   {#1}%
5264 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

5265 \newcommand*{\glsaccessdescplural}[1]{%
5266   \glsdescriptionpluralaccessdisplay
5267   {%
5268     \glsentrydescplural{#1}%
5269   }%
5270   {#1}%
5271 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5272 \newcommand*{\Glsaccessdescplural}[1]{%
5273   \glsdescriptionpluralaccessdisplay
5274   {%
5275     \Glsentrydescplural{#1}%
5276   }%
5277   {#1}%
5278 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5279 \newcommand*{\GLSaccessdescplural}[1]{%
5280   \glsdescriptionpluralaccessdisplay
5281   {%
5282     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5283   }%

```

```

5284     {#1}%
5285 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5286 \newcommand*{\glsaccessshort}[1]{%
5287   \glsshortaccessdisplay
5288   {%
5289     \glentryshort{#1}%
5290   }%
5291   {#1}%
5292 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5293 \newcommand*{\Glsaccessshort}[1]{%
5294   \glsshortaccessdisplay
5295   {%
5296     \Glsentryshort{#1}%
5297   }%
5298   {#1}%
5299 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5300 \newcommand*{\GLSaccessshort}[1]{%
5301   \glsshortaccessdisplay
5302   {%
5303     \mfirstucMakeUppercase{\glentryshort{#1}}%
5304   }%
5305   {#1}%
5306 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5307 \newcommand*{\lsaccessshortpl}[1]{%
5308   \glsshortpluralaccessdisplay
5309   {%
5310     \glentryshortpl{#1}%
5311   }%
5312   {#1}%
5313 }

```

`\Lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5314 \newcommand*{\Lsaccessshortpl}[1]{%
5315   \glsshortpluralaccessdisplay
5316   {%
5317     \Lsentryshortpl{#1}%
5318   }%
5319   {#1}%
5320 }

```

LSaccessshortpl Display the shortplural value (no link and no check for existence) converted to upper case.

```
5321 \newcommand*{\LSaccessshortpl}[1]{%
5322   \glsshortpluralaccessdisplay
5323   {%
5324     \mfirstucMakeUppercase{\glsentryshortpl{#1}}%
5325   }%
5326   {#1}%
5327 }
```

\glsaccesslong Display the long form (no link and no check for existence).

```
5328 \newcommand*{\glsaccesslong}[1]{%
5329   \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
5330 }
```

\Glsaccesslong Display the long form (no link and no check for existence).

```
5331
5332 \newcommand*{\Glsaccesslong}[1]{%
5333   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5334 }
```

\GLSaccesslong Display the long value (no link and no check for existence) converted to upper case.

```
5335 \newcommand*{\GLSaccesslong}[1]{%
5336   \glslongaccessdisplay
5337   {%
5338     \mfirstucMakeUppercase{\glsentrylong{#1}}%
5339   }%
5340   {#1}%
5341 }
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5342 \newcommand*{\glsaccesslongpl}[1]{%
5343   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5344 }
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5345
5346 \newcommand*{\Glsaccesslongpl}[1]{%
5347   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5348 }
```

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.

```
5349 \newcommand*{\GLSaccesslongpl}[1]{%
5350   \glslongpluralaccessdisplay
5351   {%
5352     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5353   }%
5354   {#1}%
5355 }
```

Keys for accessibility support.

```

5356 \define@key{glsxtrabrv}{access}{%
5357   \def\@gls@nameaccess{#1}%
5358 }
5359 \define@key{glsxtrabrv}{textaccess}{%
5360   \def\@gls@textaccess{#1}%
5361 }
5362 \define@key{glsxtrabrv}{firstaccess}{%
5363   \def\@gls@firstaccess{#1}%
5364 }
5365 \define@key{glsxtrabrv}{shortaccess}{%
5366   \def\@gls@shortaccess{#1}%
5367 }
5368 \define@key{glsxtrabrv}{shortpluralaccess}{%
5369   \def\@gls@shortaccesspl{#1}%
5370 }

```

@initaccesskeys

```

5371 \newcommand*\@gls@initaccesskeys{%
5372   \def\@gls@nameaccess{}%
5373   \def\@gls@textaccess{}%
5374   \def\@gls@firstaccess{}%
5375   \def\@gls@shortaccess{}%
5376   \def\@gls@shortaccesspl{}%
5377 }

```

essattribute@set `\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}`

```

5378 \newcommand*\@gls@ifaccessattribute@set}[3]{%
5379   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5380   {#2}%
5381   {%
5382     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5383     {#3}%
5384     {%
5385       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5386       {#2}%
5387       {#3}%
5388     }%
5389   }%
5390 }

```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to `\newabbreviation`.

```

5391 \newcommand{\@gls@setup@default@short@access}[1]{%
    Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.
5392 \ifdefempty\@gls@shortaccess
5393 {%
5394     \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5395     {%
5396         \glsxtr@insertdots\@gls@shortaccess{#1}%
5397         \eappto\ExtraCustomAbbreviationFields{%
5398             shortaccess={\expandonce\@gls@shortaccess},}%
5399     }%
5400 }%
5401 }%
5402 {}%

    If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.
5403 \ifdefempty\@gls@shortaccess
5404 {}%
5405 {%
5406     \ifdefempty\@gls@shortaccesspl
5407     {%
5408         \@gls@ifaccessattribute@set{aposplural}%
5409         {%
5410             \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5411                 \@gls@shortaccess'\abrvpluralsuffix}%
5412             }%
5413             {%
5414                 \@gls@ifaccessattribute@set{noshortplural}%
5415                 {%
5416                     \let\@gls@shortaccesspl\@gls@shortaccess
5417                 }%
5418                 {%
5419                     \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5420                         \@gls@shortaccess\abrvpluralsuffix}%
5421                     }%
5422                 }%
5423                 \eappto\ExtraCustomAbbreviationFields{%
5424                     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5425                 }%
5426             }%
5427         }%

    If access key hasn't been set, check if the nameshortaccess attribute has been set.
5428 \ifdefempty\@gls@nameaccess
5429 {%
5430     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5431     {%

        Do nothing if the shortaccess key hasn't been set.
5432         \ifdefempty\@gls@shortaccess
5433         {}%

```

```

5434      {%
5435          \eappto\ExtraCustomAbbreviationFields{%
5436              access={\expandonce\@gls@shortaccess},%
5437          }%
5438      }%
5439  }%
5440  {%}
5441  }%
5442  {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5443      \ifdefempty\@gls@textaccess
5444      {%
5445          \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5446          {%

```

Do nothing if the shortaccess key hasn't been set.

```

5447          \ifdefempty\@gls@shortaccess
5448          {}%
5449          {%
5450              \eappto\ExtraCustomAbbreviationFields{%
5451                  textaccess={\expandonce\@gls@shortaccess},%
5452              }%
5453          }%
5454      }%
5455      {}%
5456  }%
5457  {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```

5458      \ifdefempty\@gls@firstaccess
5459      {%
5460          \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5461          {%

```

Do nothing if the shortaccess key hasn't been set.

```

5462          \ifdefempty\@gls@shortaccess
5463          {}%
5464          {%
5465              \eappto\ExtraCustomAbbreviationFields{%
5466                  firstaccess={\expandonce\@gls@shortaccess},%
5467              }%
5468          }%
5469      }%
5470      {}%
5471  }%
5472  {}%
5473  }

```

End of if accsupp part

```

5474 }

```


5475 {
 No accessibility support. Just define these commands to do \glsentry<xxx>

\glsaccessname Display the name value (no link and no check for existence).
 5476 \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.
 5477 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
 5478 \newcommand*{\GLSaccessname}[1]{%
 5479 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}

\glsaccesstext Display the text value (no link and no check for existence).
 5480 \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.
 5481 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
 5482 \newcommand*{\GLSaccesstext}[1]{%
 5483 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}

glsaccessplural Display the plural value (no link and no check for existence).
 5484 \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.
 5485 \newcommand*{\GLsaccessplural}[1]{\GLsentryplural{#1}}

GLSaccessplural Display the plural value (no link and no check for existence). converted to upper case.
 5486 \newcommand*{\GLSaccessplural}[1]{%
 5487 \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}

\glsaccessfirst Display the first value (no link and no check for existence).
 5488 \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.
 5489 \newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}

\GLSaccessfirst Display the first value (no link and no check for existence). converted to upper case.
 5490 \newcommand*{\GLSaccessfirst}[1]{%
 5491 \protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}

`\glsaccessfirstplural` Display the firstplural value (no link and no check for existence).
5492 `\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.
5493 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`\GLSaccessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5494 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5495 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).
5496 `\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.
5497 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`\GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
5498 `\newcommand*{\GLSaccesssymbol}[1]{%`
5499 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`\glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence).
5500 `\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}`

`\Glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
5501 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`\GLSaccesssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.
5502 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5503 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).
5504 `\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.
5505 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.
5506 `\newcommand*{\GLSaccessdesc}[1]{%`
5507 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`\glsaccessdescplural` Display the descplural value (no link and no check for existence).
5508 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

ccessdesclplural Display the desclplural value (no link and no check for existence) with the first letter converted to upper case.
5509 \newcommand*{\GLsaccessdesclplural}[1]{\Glsentrydesclplural{#1}}

ccessdesclplural Display the desclplural value (no link and no check for existence). converted to upper case.
5510 \newcommand*{\GLSaccessdesclplural}[1]{%
5511 \protect\mfirstucMakeUppercase{\glsentrydesclplural{#1}}}

\glsaccessshort Display the short form (no link and no check for existence).
5512 \newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}

\GLsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).
5513 \newcommand*{\GLsaccessshort}[1]{\Glsentryshort{#1}}

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.
5514 \newcommand*{\GLSaccessshort}[1]{%
5515 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}

lsaccessshortpl Display the short plural form (no link and no check for existence).
5516 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).
5517 \newcommand*{\GLsaccessshortpl}[1]{\Glsentryshortpl{#1}}

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.
5518 \newcommand*{\GLSaccessshortpl}[1]{%
5519 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}

\glsaccesslong Display the long form (no link and no check for existence).
5520 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}

\GLsaccesslong Display the long form (no link and no check for existence).
5521 \newcommand*{\GLsaccesslong}[1]{\Glsentrylong{#1}}

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.
5522 \newcommand*{\GLSaccesslong}[1]{%
5523 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}

glsaccesslongpl Display the long plural form (no link and no check for existence).
5524 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}

GLsaccesslongpl Display the long plural form (no link and no check for existence).
5525 \newcommand*{\GLsaccesslongpl}[1]{\Glsentrylongpl{#1}}

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

```
5526 \newcommand*{\GLSaccesslongpl}[1]{%
5527 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

`@initaccesskeys` This does nothing if there's no accessibility support.

```
5528 \newcommand*{\@gls@initaccesskeys}{}
```

`lt@short@access` This does nothing if there's no accessibility support.

```
5529 \newcommand{\@gls@setup@default@short@access}[1]{}%
```

End of else part

```
5530 }
```

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
5531 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
5532 \newcommand{\glsifcategory}[4]{%
5533 \ifglstfield{#1}{category}{#2}{#3}{#4}%
5534 }
```

Categories can have attributes.

`categoryattribute` `\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```
5535 \newcommand*{\glissetcategoryattribute}[3]{%
5536 \csdef{@glsxtr@categoryattr@#1@#2}{#3}%
5537 }
```

`categoryattribute` `\glsgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5538 \newcommand*{\glsgetcategoryattribute}[2]{%
5539 \csuse{@glsxtr@categoryattr@#1@#2}%
5540 }
```

categoryattribute `\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
5541 \newcommand*{\glshascategoryattribute}[4]{%
5542   \ifcsvoid{@glstr@categoryattr@#1@#2}{#4}{#3}%
5543 }
```

\glissetattribute `\glissetattribute{<entry label>}{<attribute-label>}{<value>}`

Short cut where the category label is obtained from the entry information.

```
5544 \newcommand*{\glissetattribute}[3]{%
5545   \glsetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5546 }
```

\glsggetattribute `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```
5547 \newcommand*{\glsggetattribute}[2]{%
5548   \glsgcategoryattribute{\glscategory{#1}}{#2}%
5549 }
```

\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.

```
5550 \newcommand*{\glshasattribute}[4]{%
5551   \ifglstryexists{#1}%
5552   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5553   {#4}%
5554 }
```

categoryattribute `\glisifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```
5555 \newcommand{\glisifcategoryattribute}[5]{%
```

```

5556 \ifcsundef{@glxtr@categoryattr@@#1@#2}%
5557 {#5}%
5558 {\ifcsstring{@glxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
5559 }

```

`\glsifattribute` `\glsifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

5560 \newcommand{\glsifattribute}[5]{%
5561   \ifglentryexists{#1}%
5562   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5563   {#5}%
5564 }

```

Set attributes for the default general category:

```
5565 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5566 \glsetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```

5567 \newcommand*{\glsetregularcategory}[1]{%
5568   \glsetcategoryattribute{#1}{regular}{true}%
5569 }

```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```

5570 \newcommand{\glsifregularcategory}[3]{%
5571   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5572 }

```

`notregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```

5573 \newcommand{\glsifnotregularcategory}[3]{%
5574   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5575 }

```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5576 \newcommand{\glsifregular}[3]{%
5577   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
5578 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5579 \newcommand{\glsifnotregular}[3]{%
5580   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5581 }
```

`\foreachincategory` `\glsforeachincategory[<glossary labels>]{<category-label>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
5582 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5583   \foralllglossaries[#1]{#3}%
5584   {%
5585     \forglentries[#3]{#4}%
5586     {%
5587       \glsifcategory{#4}{#2}{#5}{}%
5588     }%
5589   }%
5590 }
```

`\foreachwithattribute` `\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.

```

5591 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5592   \forallglossaries[#1]{#4}%
5593   {%
5594     \forglsentries[#4]{#5}%
5595     {%
5596       \glsifattribute{#5}{#2}{#3}{#6}{}%
5597     }%
5598   }%
5599 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```

5600 \ifdef\newterm
5601 {%

```

`\newterm`

```

5602   \renewcommand*{\newterm}[2][ ]{%
5603     \newglossaryentry{#2}%
5604     {type={index},category=index,name={#2},%
5605      description={\glstrpostdescription\nopostdesc},#1}%
5606   }

```

Indexed terms are regular by default.

```

5607   \glsssetcategoryattribute{index}{regular}{true}

```

`trpostdescindex`

```

5608   \newcommand*{\glstrpostdescindex}{}

5609 }
5610 {}

```

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`.

```

5611 \ifdef\printsymbols
5612 {%

```

`glstrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```

5613   \newcommand*{\glstrnewsymbol}[3][ ]{%
5614     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5615   }

```

Symbols are regular by default.

```

5616   \glsssetcategoryattribute{symbol}{regular}{true}

```

`trpostdescsymbol`

```

5617   \newcommand*{\glstrpostdescsymbol}{}

```



```
5618 }
5619 {}
```

Similar for the numbers option.

```
5620 \ifdef\printnumbers
5621 {%
```

glsxtrnewnumber

```
5622 \ifdef\printnumbers
5623   \newcommand*\glsxtrnewnumber[3][]{%
5624     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5625   }
```

Numbers are regular by default.

```
5626   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
5627   \newcommand*\glsxtrpostdescnumber{}

5628 }
5629 {}
```

sxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5630 \newcommand*\glsxtrsetcategory[2]{%
5631   \@for\@glsxtr@label:=#1\do
5632   {%
5633     \glsfieldxdef{\@glsxtr@label}{category}{#2}%
5634   }%
5635 }
```

tcategoryforall 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.

```
5636 \newcommand*\glsxtrsetcategoryforall[2]{%
5637   \forallglossaries[#1]{\@glsxtr@type}{%
5638     \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
5639     {%
5640       \glsfieldxdef{\@glsxtr@label}{category}{#2}%
5641     }%
5642   }%
5643 }
```

trfieldtitlecase

```
\glsxtrfieldtitlecase{\label}{\field}
```

Apply title casing to the contents of the given field.

```
5644 \newcommand*\glsxtrfieldtitlecase[2]{%
```

```

5645 \expandafter\glxtrfieldtitlecasecs\expandafter
5646   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
5647 }

```

fieldtitlecasecs The command used by \glxtrfieldtitlecase. May be redefined to use a different command, for example, \xcapitalisefmtwords.

```

5648 \newcommand*{\glxtrfieldtitlecasecs}[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.

```

5649 \@ifpackageloaded{glossaries-accsupp}
5650 {
5651   \renewcommand*{\glossentrydesc}[1]{%
5652     \glsdoifexistsorwarn{#1}%
5653     {%
5654       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

5655   \glshasattribute{#1}{glossdescfont}%
5656   {%
5657     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5658     \ifcsdef{\@glxtr@attrval}%
5659     {%
5660       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5661     }%
5662     {%
5663       \GlossariesExtraWarning{Unknown control sequence name
5664         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
5665         for entry ‘#1’. Ignoring}%
5666       \let\@glxtr@glossdescfont\@firstofone
5667     }%
5668   }%
5669   {\let\@glxtr@glossdescfont\@firstofone}%
5670   \glsifattribute{#1}{glossdesc}{firstuc}%
5671   {%
5672     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5673   }%
5674   {%
5675     \glsifattribute{#1}{glossdesc}{title}%
5676     {%
5677       \@glxtr@do@titlecaps@warn
5678       \glsdescriptionaccessdisplay
5679       {%
5680         \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5681       }%
5682       {#1}%

```

```

5683     }%
5684     {%
5685         \@glxtr@glossdescfont{\glsaccessdesc{#1}}%
5686     }%
5687 }%
5688 }%
5689 }
5690 }
5691 {
5692 \renewcommand*{\glossentrydesc}[1]{%
5693     \glsdoifexistsorwarn{#1}%
5694     {%
5695         \glssetabbrvfmt{\glscategory{#1}}%
5696         \glsattribute{#1}{glossdescfont}%
5697         {%
5698             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5699             \ifcsdef{\@glxtr@attrval}%
5700             {%
5701                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5702             }%
5703             {%
5704                 \GlossariesExtraWarning{Unknown control sequence name
5705                     '\@glxtr@attrval' supplied in glossdescfont attribute
5706                     for entry '#1'. Ignoring}%
5707                 \let\@glxtr@glossdescfont\@firstofone
5708             }%
5709         }%
5710         {\let\@glxtr@glossdescfont\@firstofone}%
5711         \glsattribute{#1}{glossdesc}{firstuc}%
5712         {%
5713             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
5714         }%
5715         {%
5716             \glsattribute{#1}{glossdesc}{title}%
5717             {%
5718                 \@glxtr@do@titlecaps@warn
5719                 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5720             }%
5721             {%
5722                 \@glxtr@glossdescfont{\glsentrydesc{#1}}%
5723             }%
5724         }%
5725     }%
5726 }
5727 }

```

`\glossentryname` If the `glossname` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5728 \ifpackageloaded{glossaries-accsupp}

```

```

5729 {
5730   \renewcommand*{\glossentryname}[1]{%
5731     \@glsdoifexistsorwarn{#1}%
5732     {%
5733       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5734     \glshasattribute{#1}{glossnamefont}%
5735     {%
5736       \edef\@glsextr@attrval{\glsetattribute{#1}{glossnamefont}}%
5737       \ifcsdef{\@glsextr@attrval}%
5738       {%
5739         \letcs{\@glsextr@glossnamefont}{\@glsextr@attrval}%
5740       }%
5741       {%
5742         \GlossariesExtraWarning{Unknown control sequence name
5743           '\@glsextr@attrval' supplied in glossnamefont attribute
5744           for entry '#1'. Reverting to default \string\glssnamefont}%
5745         \let\@glsextr@glossnamefont\glssnamefont
5746       }%
5747     }%
5748     {\let\@glsextr@glossnamefont\glssnamefont}%
5749     \glsifattribute{#1}{glossname}{firstuc}%
5750     {%
5751       \glsnameaccessdisplay
5752       {%
5753         \@glsextr@glossnamefont{\Glsentryname{#1}}%
5754       }%
5755       {#1}%
5756     }%
5757     {%
5758       \glsifattribute{#1}{glossname}{title}%
5759       {%
5760         \@glsextr@do@titlecaps@warn
5761         \glsnameaccessdisplay
5762         {%
5763           \@glsextr@glossnamefont{\glsextrfieldtitlecase{#1}{name}}%
5764         }%
5765         {#1}%
5766       }%
5767       {%
5768         \glsifattribute{#1}{glossname}{uc}%
5769         {%
5770           \glsnameaccessdisplay
5771           {%

```

Hide the label from the upper-casing command.

```

5772         \letcs{\glo@name}{glo\glsetoklabel{#1}@name}%
5773         \@glsextr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5774       }%

```

```

5775         {#1}%
5776     }%
5777     {%
5778         \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
5779         \glsnameaccessdisplay
5780     {%
5781         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5782     }%
5783     {#1}%
5784 }%
5785 }%
5786 }%

```

Do post-name hook:

```

5787     \glxtrpostnamehook{#1}%
5788 }%
5789 }
5790 }
5791 {
5792     \renewcommand*{\glossentryname}[1]{%
5793         \@glsdoifexistsorwarn{#1}%
5794     {%
5795         \glsetabbrvfmt{\glscategory{#1}}%
5796         \glshasattribute{#1}{\glossnamefont}%
5797     {%
5798         \edef\@glxtr@attrval{\glsetattribute{#1}{\glossnamefont}}%
5799         \ifcsdef{\@glxtr@attrval}%
5800         {%
5801             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5802         }%
5803         {%
5804             \GlossariesExtraWarning{Unknown control sequence name
5805             '\@glxtr@attrval' supplied in glossnamefont attribute
5806             for entry '#1'. Reverting to default \string\glsnamefont}%
5807             \let\@glxtr@glossnamefont\glsnamefont
5808         }%
5809     }%
5810     {\let\@glxtr@glossnamefont\glsnamefont}%
5811     \gl@ifattribute{#1}{\glossname}{\firstuc}%
5812     {%
5813         \@glxtr@glossnamefont{\Glsentryname{#1}}%
5814     }%
5815     {%
5816         \gl@ifattribute{#1}{\glossname}{\title}%
5817     {%
5818         \@glxtr@do@titlecaps@warn
5819         \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5820     }%
5821     {%
5822         \gl@ifattribute{#1}{\glossname}{\uc}%

```

```
5823      {%
```

Hide the label from the upper-casing command.

```
5824      \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5825      \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5826      }%
5827      {%
```

This little trick is used by glossaries to allow the user to redefine `\glsnamefont` to use `\makefirstuc`. Support it even though they can now use the `firstuc` attribute.

```
5828      \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5829      \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5830      }%
5831      }%
5832      }%
```

Do post-name hook.

```
5833      \glxtrpostnamehook{#1}%
5834      }%
5835  }
5836 }
```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
5837 \ifpackageloaded{glossaries-accsupp}
5838 {
5839   \renewcommand*{\Glossentryname}[1]{%
5840     \@glstdoifexistsorwarn{#1}%
5841     {%
5842       \glsetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```
5843     \glshasattribute{#1}{glossnamefont}%
5844     {%
5845       \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5846       \ifcsdef{\@glxtr@attrval}%
5847       {%
5848         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5849         }%
5850       {%
5851         \GlossariesExtraWarning{Unknown control sequence name
5852           ‘\@glxtr@attrval’ supplied in glossnamefont attribute
5853           for entry ‘#1’. Reverting to default \string\glsnamefont}%
5854         \let\@glxtr@glossnamefont\glsnamefont
5855         }%
5856       }%
5857       {\let\@glxtr@glossnamefont\glsnamefont}%
5858       \glsnameaccessdisplay
5859       {%
5860         \@glxtr@glossnamefont{\Glsentryname{#1}}%
5861         }%
5862       {#1}%
```

Do post-name hook:

```
5863 \glxtrpostnamehook{#1}%
5864 }%
5865 }
5866 }
5867 {
5868 \renewcommand*{\Glossentryname}[1]{%
5869 \@glsdoifexistsorwarn{#1}%
5870 {%
5871 \glsetabbrvfmt{\glscategory{#1}}%
5872 \glshasattribute{#1}{glossnamefont}%
5873 {%
5874 \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5875 \ifcsdef{\@glxtr@attrval}%
5876 {%
5877 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5878 }%
5879 {%
5880 \GlossariesExtraWarning{Unknown control sequence name
5881 '\@glxtr@attrval' supplied in glossnamefont attribute
5882 for entry '#1'. Reverting to default \string\glnamefont}%
5883 \let\@glxtr@glossnamefont\glnamefont
5884 }%
5885 }%
5886 {\let\@glxtr@glossnamefont\glnamefont}%
5887 \@glxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
5888 \glxtrpostnamehook{#1}%
5889 }%
5890 }
5891 }
```

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.

`\glxtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
5892 \newcommand*{\glxtrpostnamehook}[1]{%
5893 \let\@glsnumberformat\@glxtr@defaultnumberformat
5894 \glxtrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
5895 \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
5896 \csuse{glxtrpostname\glscategory{#1}}%
5897 }
```

`\glsextrapostnamehook`

```
5898 \newcommand*{\glsextrapostnamehook}[1]{%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
5899 \newcommand*{\glsdefpostname}[2]{%
5900   \csdef{glsxtrpostname#1}{#2}%
5901 }
```

`etaccessdisplay`

```
5902 \@ifpackageloaded{glossaries-accsupp}
5903 {
5904   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
5905     \ifcsdef{gls#1accessdisplay}%
5906       {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
5907       {%
```

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.

```
5908     \edef\@gls@thisval{#1}%
5909     \@for\@gls@map:=\@gls@keymap\do{%
5910       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
5911       \ifdefequal{\@this@key}{\@gls@thisval}%
5912       {%
5913         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
5914         \@endfortrue
5915       }%
5916     }%
5917   }%
5918   \ifcsdef{gls\@gls@thisval accessdisplay}%
5919     {\letcs\@glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
5920     {\let\@glsxtr@accessdisplay\@firstoftwo}%
5921   }%
5922 }
5923 }
5924 {%
5925   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
5926     \let\@glsxtr@accessdisplay\@firstoftwo}
5927 }
```

`sentrynameother` Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```
5928 \newrobustcmd*{\glossentrynameother}[2]{%
5929   \@glsdoifexistsorwarn{#1}%
5930   {%
```

Accessibility support:

```
5931   \glsxtr@setaccessdisplay{#2}%
```

Set the abbreviation format:

```
5932   \glssetabbrvfmt{\glscategory{#1}}%
5933   \glsattribute{#1}{glossnamefont}%
5934   {%
```



```

5935 \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}}%
5936 \ifcsdef{\@glsxtr@attrval}%
5937 {%
5938 \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5939 }%
5940 {%
5941 \GlossariesExtraWarning{Unknown control sequence name
5942 '\@glsxtr@attrval' supplied in glossnamefont attribute
5943 for entry '#1'. Reverting to default \string\glsnamefont}%
5944 \let\@glsxtr@glossnamefont\glsnamefont
5945 }%
5946 }%
5947 {\let\@glsxtr@glossnamefont\glsnamefont}%
5948 \glsifattribute{#1}{glossname}{firstuc}%
5949 {%
5950 \@glsxtr@accessdisplay
5951 {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
5952 {#1}%
5953 }%
5954 {%
5955 \glsifattribute{#1}{glossname}{title}%
5956 {%
5957 \@glsxtr@do@titlecaps@warn
5958 \@glsxtr@accessdisplay
5959 {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
5960 {#1}%
5961 }%
5962 {%
5963 \glsifattribute{#1}{glossname}{uc}%
5964 {%
5965 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
5966 \@glsxtr@accessdisplay
5967 {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
5968 {#1}%
5969 }%
5970 {%
5971 \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
5972 \@glsxtr@accessdisplay
5973 {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}}%
5974 {#1}%
5975 }%
5976 }%
5977 }%

```

Do post-name hook.

```

5978 \glsxtrpostnamehook{#1}%
5979 }%
5980 }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

5981 \newif\if@glxtr@format@override
5982 \@glxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

5983 \@ifpackageloaded{hyperref}
5984 {
    If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
    don't add it.
5985   \ifHy@hyperindex
5986     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5987       \@glxtr@format@override=true
5988       \appto\theindex{\let\glshypernumber\@firstofone}%
5989     }
5990   \else
5991     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5992       \@glxtr@format@override=true
5993       \appto\theindex{\let\glshypernumber\hyperpage}%
5994     }
5995   \fi
5996 }
5997 {
5998   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5999     \@glxtr@format@override=true
6000   }
6001 }
6002 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

`doautoindexname`

```

6003 \newcommand*{\glxtrdoautoindexname}[2]{%
6004   \glshasattribute{#1}{#2}%
6005   {%
    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.
6006     \@glxtr@autoindex@setname{#1}%
    If the attribute value is simply "true" don't add an encap, otherwise use the value as the encap.
6007     \protected@edef\@glxtr@attrval{\glsggetattribute{#1}{#2}}%
6008     \if@glxtr@format@override
6009       \ifx\@glsnnumberformat\@glxtr@defaultnumberformat
6010       \else
6011         \let\@glxtr@attrval\@glsnnumberformat
6012       \fi
6013     \fi
6014     \ifdefstring{\@glxtr@attrval}{true}%

```

```

6015     {}%
6016     {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}}%
6017     \expandafter\glsxtr@autoindex\expandafter{\@glo@name}%
6018 }%
6019 {}%
6020 }

```

glsxtrautoindex

```

6021 \newcommand*{\glsxtrautoindex}{\index}

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6022 \newcommand*{\@glsxtr@autoindex@setname}[1]{%
6023   \protected@edef\@glo@name{\glsxtrautoindexentry{#1}}%
6024   \glsxtrautoindexassignsort{\@glo@sort}{#1}%
6025   \@gls@checkmkidxchars\@glo@sort
6026   \@glsxtr@autoindex@doextra@esc\@glo@sort
6027   \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
6028 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6029 \newcommand*{\glsxtrautoindexentry}[1]{\string\glsentryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```

6030 \newcommand*{\glsxtrautoindexassignsort}[2]{%
6031   \glsletentryfield{#1}{#2}{sort}%
6032 }

```

dex@doextra@esc

```

6033 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
    Escape the escape character unless it has already been escaped.
6034   \ifx\@glsxtr@autoindex@esc\@gls@quotechar
6035   \else
6036     \def\@gls@checkedmkidx{}%
6037     \edef\@glsxtr@checkspch{%
6038       \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
6039       \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
6040       \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6041     \@glsxtr@checkspch
6042     \let#1\@gls@checkedmkidx\relax
6043   \fi

```

Escape actual character unless it has already been escaped.

```

6044   \ifx\@glsxtr@autoindex@at\@gls@actualchar
6045   \else
6046     \def\@gls@checkedmkidx{}%
6047     \edef\@glsxtr@checkspch{%
6048       \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
6049       \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil

```

```

6050      \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6051      \@glsxtr@checkspch
6052      \let#1\@gls@checkedmkidx\relax
6053      \fi

  Escape level character unless it has already been escaped.
6054      \ifx\@glsxtr@autoindex@level\@gls@levelchar
6055      \else
6056      \def\@gls@checkedmkidx{}%
6057      \edef\@glsxtr@checkspch{%
6058        \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
6059        \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
6060        \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6061      \@glsxtr@checkspch
6062      \let#1\@gls@checkedmkidx\relax
6063      \fi

```

Escape encap character unless it has already been escaped.

```

6064      \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6065      \else
6066      \def\@gls@checkedmkidx{}%
6067      \edef\@glsxtr@checkspch{%
6068        \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6069        \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6070        \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6071      \@glsxtr@checkspch
6072      \let#1\@gls@checkedmkidx\relax
6073      \fi
6074 }

```

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.

```

6075 \newcommand*{\@glsxtr@autoindex@at}{}

```

trSetActualChar Set the actual character.

```

6076 \newcommand*{\GlsXtrSetActualChar}[1]{%
6077   \gdef\@glsxtr@autoindex@at{#1}%
6078   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6079     \@glsxtr@autoindex@escspch{#1}\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6080   }%
6081 }
6082 \@onlypreamble\GlsXtrSetActualChar
6083 \makeatother
6084 \GlsXtrSetActualChar{@}
6085 \makeatletter

```

autoindex@encap Encap character for use with \index.

```

6086 \newcommand*{\@glsxtr@autoindex@encap}{}

```

XtrSetEncapChar Set the encap character.

```
6087 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6088   \gdef\@glstr@autoindex@encap{#1}%
6089   \def\@glstr@autoindex@escencap##1##2##3\@glstr@endescspch{%
6090     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escencap}{##1}{##2}{##3}%
6091   }%
6092 }
6093 \GlsXtrSetEncapChar{||}
6094 \@onlypreamble\GlsXtrSetEncapChar
```

autoindex@level Level character for use with \index.

```
6095 \newcommand*{\@glstr@autoindex@level}{}
```

XtrSetLevelChar Set the encap character.

```
6096 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6097   \gdef\@glstr@autoindex@level{#1}%
6098   \def\@glstr@autoindex@esclevel##1##2##3\@glstr@endescspch{%
6099     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@esclevel}{##1}{##2}{##3}%
6100   }%
6101 }
6102 \GlsXtrSetLevelChar{!}
6103 \@onlypreamble\GlsXtrSetLevelChar
```

r@autoindex@esc Escape character for use with \index.

```
6104 \newcommand*{\@glstr@autoindex@esc}{"}
```

lsXtrSetEscChar Set the escape character.

```
6105 \newcommand*{\GlsXtrSetEscChar}[1]{%
6106   \gdef\@glstr@autoindex@esc{#1}%
6107   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
6108     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escquote}{##1}{##2}{##3}%
6109   }%
6110 }
6111 \GlsXtrSetEscChar{"}
6112 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
6113 \ifdef\actualchar
6114   {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6115 {}
```

Quote character \quotechar:

```
6116 \ifdef\quotechar
6117   {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6118 {}
```

Level character \levelchar:

```
6119 \ifdef\levelchar
6120   {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6121 {}
```

Encap character \encapchar:

```
6122 \ifdef\encapchar
6123 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6124 {}}
```

leto@endescspch

```
6125 \def\@glxtr@gobbleto@endescspch#1\@glxtr@endescspch{}
```

toindex@esc@spch

```
\@glxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
6126 \newcommand*{\@glxtr@autoindex@escspch}[5]{%
6127   \@glstmpb=\expandafter{\@glstmpb\the\@glstmpb}%
6128   \toks@={#3}%
6129   \ifx\@nnil#3\relax
6130     \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch#5\@glxtr@endescspch}%
6131   \else
6132     \ifx\@nnil#4\relax
6133       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
6134       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
6135         #4#5\@glxtr@endescspch}%
6136     \else
6137       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
6138       \def\@glxtr@checkspch{\@glxtr@autoindex@esc#1}%
6139       \def\@glxtr@checkspch{\@glxtr@checkspch#2#5#1\@nnil#1\@glxtr@endescspch}%
6140     \fi
6141   \fi
6142   \@glxtr@checkspch
6143 }
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
6144 \renewcommand*{\Glossentrydesc}[1]{%
6145   \glsoifexistsorwarn{#1}%
6146   {%
6147     \glsetabbrvfmt{\glscategory{#1}}%
6148     \Glsaccessdesc{#1}%
6149   }%
6150 }
```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
6151 \renewcommand*{\Glossentrysymbol}[1]{%
6152   \glsoifexistsorwarn{#1}%
6153   {%
6154     \glsetabbrvfmt{\glscategory{#1}}%
6155     \Glsaccesssymbol{#1}%
6156   }%
6157 }
```

`\glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```
6158 \renewcommand*{\Glossentrysymbol}[1]{%
6159   \glsdoifexistsorwarn{#1}%
6160   {%
6161     \glssetabbrvfmt{\glscategory{#1}}%
6162     \Glsaccesssymbol{#1}%
6163   }%
6164 }
```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\enableInitialTagging` 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.

```
6165 \newcommand*{\GlsXtrEnableInitialTagging}{%
6166   \@ifstar\s@glsextr@enabletagging\@glsextr@enabletagging
6167 }
6168 \@onlypreamble\GlsXtrEnableInitialTagging
```

`\r@enabletagging` Starred version undefines command.

```
6169 \newcommand*{\s@glsextr@enabletagging}[2]{%
6170   \undef#2%
6171   \@glsextr@enabletagging{#1}{#2}%
6172 }
```

`\r@enabletagging` Internal command.

```
6173 \newcommand*{\@glsextr@enabletagging}[2]{%
  Set attributes for categories given in the first argument.
6174   \@for\@glsextr@cat:=#1\do
6175   {%
6176     \ifdefempty\@glsextr@cat
6177     {}%
6178     {\glssetcategoryattribute{\@glsextr@cat}{tagging}{true}}%
6179   }%
6180   \newrobustcmd*#2[1]{##1}%
6181   \def\@glsextr@taggingcs{#2}%
6182   \renewcommand*\@glsextr@activate@initialtagging{%
6183     \let#2\@glsextr@tag
6184   }%
6185   \ifundef\@gls@preglossaryhook
6186   {\GlossariesExtraWarning{Initial tagging requires at least
6187     glossaries.sty v4.19 to work correctly}}%
6188   {}%
6189 }
```

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`

```
6190 \ifundef\mfu@checkword@do
6191 {
6192   \newcommand*{\mfu@checkword@do}[1]{%
6193     \ifdefstring{\mfu@checkword@arg}{#1}%
6194     {%
6195       \let\@mfu@domakefirstuc\@firstofone
6196       \listbreak
6197     }%
6198   }%
6199 }
```

`\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.

```
6200 \ifundef\mfu@checkword
6201 {
6202   \newcommand{\@glstr@do@titlecaps@warn}{%
6203     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6204       support not available}%
6205     \let\@glstr@do@titlecaps@warn\relax
6206   }
6207 }
6208 {
6209   \renewcommand*{\mfu@checkword}[1]{%
6210     \def\mfu@checkword@arg{#1}%
6211     \let\@mfu@domakefirstuc\makefirstuc
6212     \forlistloop\mfu@checkword@do\@mfu@nocaplist
6213   }
6214 }
6215 }
6216 {}% no patch required
```

`@titlecaps@warn` Do warning if title case not supported.

```
6217 \newcommand*{\@glstr@do@titlecaps@warn}{}
```

`@initialtagging` Used in `\printglossary` but at least v4.19 of `glossaries` required.

```
6218 \newcommand*\@glstr@activate@initialtagging{}
```

`\@glstr@tag` Definition of tagging command when used in glossary.

```
6219 \newrobustcmd*{\@glstr@tag}[1]{%
6220   \gl@ifattribute{\gl@currententrylabel}{tagging}{true}%
6221   {\glstr@tagfont{#1}}{#1}%
6222 }
```

`\glstrtagfont` Used in the glossary.

```
6223 \newcommand*{\glstrtagfont}[1]{\underline{#1}}
```


`preglossaryhook` This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```
6224 \ifdef \@gls@preglossaryhook
```

```
6225 {
```

```
6226   \renewcommand*{\@gls@preglossaryhook}{%
```

```
6227     \@glsxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, `\@glsxtr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

```
6228   \ifundef \@glsxtr@org@postdescription
```

```
6229   {%
```

```
6230     \let \@glsxtr@org@postdescription \glspostdescription
```

```
6231     \renewcommand*{\glspostdescription}{%
```

```
6232       \ifglstryexists{\glscurrententrylabel}%
```

```
6233       {%
```

```
6234         \glsxtrpostdescription
```

```
6235         \@glsxtr@org@postdescription
```

```
6236       }%
```

```
6237     {}}%
```

```
6238   }%
```

```
6239 }%
```

```
6240 {}%
```

Enable the options used by `\@glsxtrp`:

```
6241   \glossxtrsetpopts
```

```
6242 }%
```

```
6243 }
```

```
6244 {}
```

`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.

```
6245 \newcommand*{\glsxtrpostdescription}{%
```

```
6246   \csuse{\glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
```

```
6247 }
```

`postdescgeneral`

```
6248 \newcommand*{\glsxtrpostdescgeneral}{{}
```

`xtrpostdescterm`

```
6249 \newcommand*{\glsxtrpostdescterm}{{}
```

`postdescacronym`

```
6250 \newcommand*{\glsxtrpostdescacronym}{{}
```

escabbreviation

```
6251 \newcommand*{\glstrpostdescabbreviation}{}
```

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.

```
6252 \newcommand*{\glsdefpostdesc}[2]{%
6253   \csdef{glstrpostdesc#1}{#2}%
6254 }
```

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.

```
6255 \renewcommand*{\glspostlinkhook}{%
6256   \ifglstryexists{\glslabel}{\glstrpostlinkhook}{}%
6257 }
```

trpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.

```
6258 \newcommand*{\glstrpostlinkhook}{%
6259   \glstrdiscardperiod{\glslabel}%
6260   {\glstrpostlinkendsentence}%
6261   {\glstrifcustomdiscardperiod
6262     {\glstrifperiod{\glstrpostlinkendsentence}{\glstrpostlink}}}%
6263   {\glstrpostlink}%
6264   }%
6265 }
```

omdiscardperiod Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6266 \newcommand*{\glstrifcustomdiscardperiod}[2]{#2}
```

\glstrpostlink

```
6267 \newcommand*{\glstrpostlink}{%
6268   \csuse{glstrpostlink\glscategory{\glslabel}}%
6269 }
```

\glsdefpostlink Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite \glstrpostlink.

```
6270 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
6271   \ifthenelse{\equal{#1}{}}{%
6272     {\PackageError{glossaries-extra}
6273       {Invalid empty category label in \string\glsdefpostlink}{}}%
6274     {\csdef{glstrpostlink#1}{#2}}%
6275 }
```

linkendsentence Done by \glxtrpostlinkhook if a full stop is discarded.

```
6276 \newcommand*{\glxtrpostlinkendsentence}{%
6277   \ifcsdef{glxtrpostlink\glscategory{\glslabel}}
6278   {%
6279     \csuse{glxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
6280   .\spacefactor\sfcode'\. \relax
6281 }%
6282 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
6283   \spacefactor\sfcode'\. \relax
6284 }%
6285 }
```

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.

```
6286 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6287   \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}%
6288 }
```

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.

```
6289 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6290   \glxtrifwasfirstuse
6291   {%
6292     \ifglshassymbol{\glslabel}%
6293     {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
6294     {}%
6295   }%
6296   {}%
6297 }
```

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.

```
6298 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6299   \glxtrifwasfirstuse
6300   {%
6301     \space\glxtrparen
6302     {%
6303       \ifglshassymbol{\glslabel}%
6304       {\glssaccesssymbol{\glslabel}, }%
6305       {}%
6306       \glssaccessdesc{\glslabel}%
6307     }%
6308   }%
6309   {}%
6310 }
```

`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).

```

6311 \newcommand*{\glxtrdiscardperiod}[3]{%
6312   \glxtrifwasfirstuse
6313   {%
6314     \glsifattribute{#1}{retainfirstuseperiod}{true}%
6315     {#3}%
6316     {%
6317       \glsifattribute{#1}{discardperiod}{true}%
6318       {%
6319         \glsifplural
6320         {%
6321           \glsifattribute{#1}{pluraldiscardperiod}{true}%
6322           {\glxtrifperiod{#2}{#3}}%
6323           {#3}%
6324         }%
6325       }%
6326       \glxtrifperiod{#2}{#3}%
6327     }%
6328   }%
6329   {#3}%
6330 }%
6331 }%
6332 {%
6333   \glsifattribute{#1}{discardperiod}{true}%
6334   {%
6335     \glsifplural
6336     {%
6337       \glsifattribute{#1}{pluraldiscardperiod}{true}%
6338       {\glxtrifperiod{#2}{#3}}%
6339       {#3}%
6340     }%
6341   }%
6342   \glxtrifperiod{#2}{#3}%
6343 }%
6344 }%
6345 {#3}%
6346 }%
6347 }

```

`\glxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

6348 \newcommand*{\glxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`glxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This

doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```
6349 \newcommand*{\glxtr@punclist}{.,:;!}
```

punctuationmark Add character to punctuation list.

```
6350 \newcommand*{\glxtraddpunctuationmark}[1]{\appto\glxtr@punclist{#1}}
```

punctuationmarks Reset the punctuation list.

```
6351 \newcommand*{\glxtrsetpunctuationmarks}[1]{\def\glxtr@punclist{#1}}
```

```
\glxtrifpunc \glxtrifnextpunc{<true part>}{<false part>}
```

Test if this is followed by a punctuation mark. (Adapted from \new@ifnextchar.)

```
6352 \newcommand*{\glxtrifnextpunc}[2]{%
6353   \def\reserved@a{#1}%
6354   \def\reserved@b{#2}%
6355   \futurelet\@glspunc@token\glxtr@ifnextpunc
6356 }
```

glxtr@ifnextpunc

```
6357 \newcommand*{\glxtr@ifnextpunc}{%
6358   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6359   \reserved@b
6360 }
```

glxtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
6361 \newcommand*{\glxtr@ifpunctoken}[1]{%
6362   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6363 }
```

glxtr@ifpunctoken

```
6364 \def\@glxtr@ifpunctoken#1#2{%
6365   \let\reserved@d=#2%
6366   \ifx\reserved@d\@nnil
6367     \let\glxtr@next\@glxtr@notfoundinlist
6368   \else
6369     \ifx#1\reserved@d
6370       \let\glxtr@next\@glxtr@foundinlist
6371     \else
6372       \let\glxtr@next\@glxtr@ifpunctoken
6373     \fi
6374   \fi
6375   \glxtr@next#1%
6376 }
```

glxtr@foundinlist

```
6377 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6378 \def\@glxstr@notfoundinlist#1{\@secondoftwo}
```

glxstrdopostpunc

```
\glxstrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
6379 \newcommand{\glxstrdopostpunc}[1]{%
6380   \glxstrifnextpunc{\@glxstr@swaptwo{#1}}{#1}%
6381 }
```

@glxstr@swaptwo

```
6382 \newcommand{\@glxstr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
6383 \define@key{glxstrabbrv}{category}{%
6384   \edef\glscategorylabel{#1}%
6385   \ifcsdef{@glxstr@current@#1}%
6386   {%
```

Warning should already have been issued.

```
6387   \let\@glxstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6388   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6389   \glxstr@applyabbrvstyle{\csglscategorylabel @glxstr@current@#1\endcsname}%
6390   \let\GlsXtrWarnDeprecatedAbbrStyle\@glxstr@orgwarndep
6391 }%
6392 }%
6393 }
```

Save the short plural form. This may be needed before the entry is defined.

```
6394 \define@key{glxstrabbrv}{shortplural}{%
6395   \def\@glxstr@shortpl{#1}%
6396 }
```

Similarly for the long plural form.

```
6397 \define@key{glxstrabbrv}{longplural}{%
6398   \def\@glxstr@longpl{#1}%
6399 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

`\glsshortpltok`

```
6400 \newtoks\glsshortpltok
```

`\glslongpltok`

```
6401 \newtoks\glslongpltok
```

`\glsxtr@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.

```
6402 \newcommand*{\@glsxtr@insertdots}[2]{%
6403   \def#1{}%
6404   \@glsxtr@insert@dots#1#2\@nnil
6405 }
```

`\glsxtr@insert@dots`

```
6406 \newcommand*{\@glsxtr@insert@dots}[2]{%
6407   \ifx\@nnil#2\relax
6408   \let\@glsxtr@insert@dots@next\@gobble
6409   \else
6410   \ifx\relax#2\relax
6411   \else
6412     \appto#1{#2.}%
6413   \fi
6414   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6415   \fi
6416   \@glsxtr@insert@dots@next#1%
6417 }
```

Similarly provide a way of replacing spaces with `\glsxtrwordsep`, which first needs to be defined:

`\glsxtrwordsep`

```
6418 \newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

`\glsxtrword`

```
6419 \newcommand*{\glsxtrword}[1]{#1}
```

`\glsxtr@markwordseps`

```
6420 \newcommand*{\@glsxtr@markwordseps}[2]{%
6421   \def#1{}%
6422   \@glsxtr@mark@wordseps#1#2 \@nnil
6423 }
```

r@mark@wordseps

```
6424 \def\@glxtr@mark@wordseps#1#2 #3{%
6425   \ifdefempty{#1}%
6426   {\def#1{\protect\glxtrword{#2}}}%
6427   {\appto#1{\protect\glxtrwordsep\protect\glxtrword{#2}}}%
6428   \ifx\@nnil#3\relax
6429   \let\@glxtr@mark@wordseps@next\relax
6430   \else
6431   \def\@glxtr@mark@wordseps@next{%
6432     \@glxtr@mark@wordseps#1#3}%
6433   \fi
6434   \@glxtr@mark@wordseps@next
6435 }
```

newabbreviation Define a new generic abbreviation.

```
6436 \newcommand*{\newabbreviation}[4] [] {%
6437   \glxtr@newabbreviation{#1}{#2}{#3}{#4}%
6438 }
```

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.)

```
6439 \newcommand*{\glxtr@newabbreviation}[4] {%
6440   \glskeylisttok{#1}%
6441   \glslabeltok{#2}%
6442   \glsshorttok{#3}%
6443   \glslongtok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```
6444 \def\glxtrorgshort{#3}%
6445 \def\glxtrorglong{#4}%

```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
6446 \def\ExtraCustomAbbreviationFields{}%
```

Initialise accessibility settings if required.

```
6447 \@gls@initaccesskeys
```

Get the category.

```
6448 \def\glscategorylabel{abbreviation}%
6449 \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```
6450 \setkeys*{glxtrabbrv}[shortplural,longplural]{#1}%

```

Set the default long plural

```
6451 \def\@gls@longpl{#4\glspluralsuffix}%
6452 \let\@gls@default@longpl\@gls@longpl

```

Has the markwords attribute been set?

```
6453 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6454 {%

```



```

6455 \glsxtr@markwordseps\@gls@long{#4}%
6456 \expandafter\def\expandafter\@gls@longpl\expandafter
6457 {\@gls@long\glspluralsuffix}%
6458 \let\@gls@default@longpl\@gls@longpl

Update \glslongtok.
6459 \expandafter\glslongtok\expandafter{\@gls@long}%
6460 }%
6461 {}%

Has the markshortwords attribute been set? (Not compatible with insertdots.)
6462 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6463 {%
6464 \glsxtr@markwordseps\@gls@short{#3}%
6465 }%
6466 {%

Has the insertdots attribute been set?
6467 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6468 {%
6469 \glsxtr@insertdots\@gls@short{#3}%
6470 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6471 }%
6472 {\def\@gls@short{#3}}%
6473 }%

Has the aposplural attribute been set? (Not compatible with noshortplural.)
6474 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6475 {%
6476 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6477 '\abbrvpluralsuffix}%
6478 }%
6479 {%

Has the noshortplural attribute been set?
6480 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6481 {%
6482 \let\@gls@shortpl\@gls@short
6483 }%
6484 {%
6485 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6486 \abbrvpluralsuffix}%
6487 }%
6488 }%

Update \glsshorttok:
6489 \expandafter\glsshorttok\expandafter{\@gls@short}%

Hook for further customisation if required:
6490 \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).

```

```
6491 \setkeys*{glsxtrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
6492 \ifx\@gls@default@longpl\@gls@longpl
```

```
6493 \else
```

Has the markwords attribute been set?

```
6494 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
```

```
6495 {%
```

```
6496 \expandafter\@glsxtr@markwordseps\expandafter\@gls@longpl\expandafter
```

```
6497 {\@gls@longpl}%
```

```
6498 }%
```

```
6499 {}%
```

```
6500 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
6501 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
```

```
6502 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6503 \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6504 \newabbreviationhook
```

Define this entry:

```
6505 \protected@edef\@do@newglossaryentry{%
```

```
6506 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
6507 {%
```

```
6508 type=\glsxtrabbrvtype,%
```

```
6509 category=abbreviation,%
```

```
6510 short={\the\glsshorttok},%
```

```
6511 shortplural={\the\glsshortpltok},%
```

```
6512 long={\the\glslongtok},%
```

```
6513 longplural={\the\glslongpltok},%
```

```
6514 name={\the\glsshorttok},%
```

```
6515 \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6516 \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6517 \the\glskeylisttok
```

```
6518 }%
```

```
6519 }%
```

```
6520 \@do@newglossaryentry
```

```
6521 \GlsXtrPostNewAbbreviation
```

```
6522 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

```
6523 \newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}%
```

NewAbbreviation Hook used by abbreviation styles.
6524 `\newcommand*{\GlsXtrPostNewAbbreviation}{}`

bbreviationhook Hook for use with `\newabbreviation`.
6525 `\newcommand*{\newabbreviationhook}{}`

reviationFields
6526 `\newcommand*{\CustomAbbreviationFields}{}`

\glstrparen For the parenthetical styles.
6527 `\newcommand*{\glstrparen}[1]{(#1)}`

lsxtrfullformat Full format without case change.
6528 `\newcommand*{\glsxtrfullformat}[2]{%`
6529 `\glsfirstlongfont{\glssaccesslong{#1}}#2\glsxtrfullsep{#1}%`
6530 `\glstrparen{\protect\glsfirstabbrvfont{\glssaccessshort{#1}}}%`
6531 `}`

lsxtrfullformat Full format with case change.
6532 `\newcommand*{\Glsxtrfullformat}[2]{%`
6533 `\glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%`
6534 `\glstrparen{\protect\glsfirstabbrvfont{\glssaccessshort{#1}}}%`
6535 `}`

xtrfullplformat Plural full format without case change.
6536 `\newcommand*{\glsxtrfullplformat}[2]{%`
6537 `\glsfirstlongfont{\glssaccesslongpl{#1}}#2\glsxtrfullsep{#1}%`
6538 `\glstrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{#1}}}%`
6539 `}`

xtrfullplformat Plural full format with case change.
6540 `\newcommand*{\Glsxtrfullplformat}[2]{%`
6541 `\glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%`
6542 `\glstrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{#1}}}%`
6543 `}`

\glsxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.
6544 `\newcommand*{\glsxtrfullsep}[1]{\space}`

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

inlinefullformat Full format without case change.
6545 `\newcommand*{\glxtrininlinefullformat}{\glsxtrfullformat}`

inlinefullformat Full format with case change.
6546 `\newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}`

`\glxtrfullplformat` Plural full format without case change.

```
6547 \newcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}
```

`\glxtrfullplformat` Plural full format with case change.

```
6548 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glxtrfull` set of commands instead.

`\glentryfull`

```
6549 \renewcommand*{\glentryfull}[1]{\glxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
6550 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glentryfullpl`

```
6551 \renewcommand*{\glentryfullpl}[1]{\glxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

```
6552 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\glfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

```
6553 \newcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvdefaultfont{#1}}
```

`\glabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

```
6554 \newcommand*{\glfirstabbrvdefaultfont}[1]{\glabbrvfont{#1}}
```

`\glabbrvfont` Font changing command used for the abbreviation on subsequent use.

```
6555 \newcommand*{\glabbrvfont}[1]{\glabbrvdefaultfont{#1}}
```

`\glabbrvdefaultfont`

```
6556 \newcommand*{\glabbrvdefaultfont}[1]{#1}
```

`\glslongfont` Font changing command used for the long form in commands like `\glxtrlong`.

```
6557 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glxtrlong`.

```
6558 \newcommand*{\glslongdefaultfont}[1]{#1}
```

`\glfirstlongfont` Font changing command used for the long form on first use or in the full format.

```
6559 \newcommand*{\glfirstlongfont}[1]{\glslongfont{#1}}
```

`\glslongdefaultfont`

```
6560 \newcommand*{\glfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

brvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.

```
6561 \newcommand*{\glxtrabbrvpluralsuffix}{\glspluralsuffix}
```

brvpluralsuffix Default plural suffix.

```
6562 \newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}
```

\glxtrfull Full form (no case-change).

```
6563 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\ns@glxtrfull}
```

```
6564 \newcommand*{\ns@glxtrfull}[2][\{%
```

```
6565 \new@ifnextchar[\@glxtr@full{#1}{#2}}%
```

```
6566 \@glxtr@full{#1}{#2}[]}%
```

```
6567 }
```

\@glxtr@full Low-level macro:

```
6568 \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).

```
6569 \@glxtr@record{#1}{#2}{glslink}%
```

```
6570 \glsdoifexists{#2}%
```

```
6571 {%
```

```
6572 \glsetabbrvfmt{\glscategory{#2}}%
```

```
6573 \let\do@glsc@link@checkfirsthyper\@glsc@link@nocheckfirsthyper
```

```
6574 \let\glsc@link@secondoftwo
```

```
6575 \let\glsc@link@firstofthree
```

```
6576 \let\glsc@link@empty
```

```
6577 \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.

```
6578 \glxtrsetupfulldefs
```

```
6579 \@glsc@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
6580 }%
```

```
6581 \glspostlinkhook
```

```
6582 }
```

trsetupfulldefs

```
6583 \newcommand*{\glxtrsetupfulldefs}{%
```

```
6584 \let\glxtrifwasfirstuse\@firstoftwo
```

```
6585 }
```

\Glsxtrfull Full form (first letter uppercase).

```
6586 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
```

```
6587 \newcommand*{\ns@Glsxtrfull}[2][\{%
```

```
6588 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
```

```
6589 \@Glsxtr@full{#1}{#2}[]}%
```

```
6590 }
```

`\@Glsxtr@full` Low-level macro:

```
6591 \def\@Glsxtr@full#1#2[#3]{%
6592   \glsdoifexists{#2}%
6593   {%
6594     \glssetabbrvfmt{\glscategory{#2}}%
6595     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6596     \let\glsifplural\@secondoftwo
6597     \let\glscapscase\@secondofthree
6598     \let\glsinsert\@empty
6599     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6600     \glsxtrsetupfulldefs
6601     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6602   }%
6603   \glspostlinkhook
6604 }
```

`\GLSxtrfull` Full form (all uppercase).

```
6605 \newrobustcmd*{\GLSxtrfull}{\@gl@hyp@opt\@ns@GLSxtrfull}
6606 \newcommand*\ns@GLSxtrfull[2][{}]{%
6607   \new@ifnextchar[\@GLSxtr@full{#1}{#2}}%
6608   {\@GLSxtr@full{#1}{#2}[{}]}%
6609 }
```

`\@GLSxtr@full` Low-level macro:

```
6610 \def\@GLSxtr@full#1#2[#3]{%
6611   \glsdoifexists{#2}%
6612   {%
6613     \glssetabbrvfmt{\glscategory{#2}}%
6614     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6615     \let\glsifplural\@secondoftwo
6616     \let\glscapscase\@thirdofthree
6617     \let\glsinsert\@empty
6618     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6619     \glsxtrsetupfulldefs
6620     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6621   }%
6622   \glspostlinkhook
6623 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
6624 \newrobustcmd*{\glsxtrfullpl}{\@gl@hyp@opt\@ns@glsxtrfullpl}
6625 \newcommand*\ns@glsxtrfullpl[2][{}]{%
6626   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}}%
6627   {\@glsxtr@fullpl{#1}{#2}[{}]}%
6628 }
```

`\@glsxtr@fullpl` Low-level macro:

```
6629 \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).

```

6630 \@glxstr@record{#1}{#2}{glslink}%
6631 \glsoifexists{#2}%
6632 {%
6633 \glsetabbrvfmt{\glscategory{#2}}%
6634 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6635 \let\gl@ifplural\@firstoftwo
6636 \let\gl@scapscase\@firstofthree
6637 \let\gl@insert\@empty
6638 \def\glscustomtext{\glxstrinlinefullplformat{#2}{#3}}%
6639 \glxstrsetupfulldefs
6640 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6641 }%
6642 \glspostlinkhook
6643 }

```

\Glsxtrfullpl Plural full form (first letter uppercase).

```

6644 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\@ns@Glsxtrfullpl}
6645 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
6646 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6647 {\@Glsxtr@fullpl{#1}{#2}[]}%
6648 }

```

\@Glsxtr@fullpl Low-level macro:

```

6649 \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).

```

6650 \@glxstr@record{#1}{#2}{glslink}%
6651 \glsoifexists{#2}%
6652 {%
6653 \glsetabbrvfmt{\glscategory{#2}}%
6654 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6655 \let\gl@ifplural\@firstoftwo
6656 \let\gl@scapscase\@secondofthree
6657 \let\gl@insert\@empty
6658 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
6659 \glxstrsetupfulldefs
6660 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6661 }%
6662 \glspostlinkhook
6663 }

```

\GLSxtrfullpl Plural full form (all upper case).

```

6664 \newrobustcmd*{\GLSxtrfullpl}{\@gl@hyp@opt\@ns@GLSxtrfullpl}
6665 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
6666 \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
6667 {\@GLSxtr@fullpl{#1}{#2}[]}%

```

6668 }

\@GLSxtr@fullpl Low-level macro:

6669 \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).

6670 \@glxtr@record{#1}{#2}{glslink}%

6671 \glstoifexists{#2}%

6672 {%

6673 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper

6674 \let\glslifplural\@firstoftwo

6675 \let\glscapscase\@thirdofthree

6676 \let\glslinsert\@empty

6677 \def\glscustomtext{%

6678 \mfirstucMakeUppercase{\glxtrinlinefullplformat{#2}{#3}}}%

6679 \glxtrsetupfulldefs

6680 \@glsl@link[#1]{#2}{\csname gls\@glstype @entryfmt\endcsname}%

6681 }%

6682 \glspostlinkhook

6683 }

The short and long forms work in a similar way to acronyms.

\glxtrshort

6684 \newrobustcmd*{\glxtrshort}{\@glsl@hyp@opt\@ns@glxtrshort}

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

6685 \newcommand*{\@ns@glxtrshort}[2][]{%

6686 \new@ifnextchar[{\@glxtrshort{#1}{#2}}{\@glxtrshort{#1}{#2}[]}%

6687 }

Read in the final optional argument:

6688 \def\@glxtrshort#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).

6689 \@glxtr@record{#1}{#2}{glslink}%

6690 \glstoifexists{#2}%

6691 {%

Need to make sure \glssabrvfont is set correctly.

6692 \glsssetabrvfmt{\glscategory{#2}}%

6693 \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper

6694 \let\glxtrifwasfirstuse\@secondoftwo

6695 \let\glslifplural\@secondoftwo

6696 \let\glscapscase\@firstofthree

6697 \let\glslinsert\@empty

6698 \def\glscustomtext{%

6699 \glssabrvfont{\glssaccessshort{#2}\ifglxtrinsertinside#3\fi}%

6700 \ifglxtrinsertinside\else#3\fi


```

6701 }%
6702 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6703 }%
6704 \glspostlinkhook
6705 }

```

\Glsxtrshort

```

6706 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
6707 \newcommand*{\ns@Glsxtrshort}[2][{}]{%
6708   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2}[]}%
6709 }

    Read in the final optional argument:
6710 \def\@Glsxtrshort#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).
6711   \@glsxtr@record{#1}{#2}{glslink}%
6712   \glsdoifexists{#2}%
6713   {%
6714     \glssetabbrvfmt{\glscategory{#2}}%
6715     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6716     \let\glsxtrifwasfirstuse\@secondoftwo
6717     \let\glsifplural\@secondoftwo
6718     \let\glsapscase\@secondofthree
6719     \let\glsinsert\@empty
6720     \def\glscustomtext{%
6721       \glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
6722       \ifglsxtrininsertinside\else#3\fi
6723     }%
6724     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6725   }%
6726   \glspostlinkhook
6727 }

```

\GLSxtrshort

```

6728 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
6729 \newcommand*{\ns@GLSxtrshort}[2][{}]{%
6730   \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2}[]}%
6731 }

    Read in the final optional argument:
6732 \def\@GLSxtrshort#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).
6733   \@glsxtr@record{#1}{#2}{glslink}%

```

```

6734 \glsdoifexists{#2}%
6735 {%
6736   \glssetabbrvfmt{\glscategory{#2}}%
6737   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6738   \let\glxtrifwasfirstuse\@secondoftwo
6739   \let\gl@ifplural\@secondoftwo
6740   \let\glscapscase\@thirdofthree
6741   \let\glinsert\@empty
6742   \def\glscustomtext{%
6743     \mfirstucMakeUppercase
6744     {\glabbrvfont{\glaccessshort{#2}\ifglxtrininsertinside#3\fi}%
6745     \ifglxtrininsertinside\else#3\fi
6746   }%
6747   }%
6748   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6749 }%
6750 \glspostlinkhook
6751 }

```

\glxtrlong

```

6752 \newrobustcmd*{\glxtrlong}{\@gl@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6753 \newcommand*{\ns@glxtrlong}[2] [] {%
6754   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
6755 }

```

Read in the final optional argument:

```

6756 \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).
6757   \@glxtr@record{#1}{#2}{glslink}%
6758   \glsdoifexists{#2}%
6759   {%
6760     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6761     \let\glxtrifwasfirstuse\@secondoftwo
6762     \let\gl@ifplural\@secondoftwo
6763     \let\glscapscase\@firstofthree
6764     \let\glinsert\@empty
6765     \def\glscustomtext{%
6766       \glslongfont{\glaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6767       \ifglxtrininsertinside\else#3\fi
6768     }%
6769     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6770   }%
6771   \glspostlinkhook
6772 }

```

\Glsxtrlong

```

6773 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\@ns@Glsxtrlong}

```

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

```
6774 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6775   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6776 }
```

Read in the final optional argument:

```
6777 \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).

```
6778   \@glsxtr@record{#1}{#2}{glslink}%
6779   \glsdoifexists{#2}%
6780   {%
6781     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6782     \let\glsxtrifwasfirstuse\@secondoftwo
6783     \let\glsifplural\@secondoftwo
6784     \let\gls caps case\@secondofthree
6785     \let\glsinsert\@empty
6786     \def\gls custom text{%
6787       \gls long font{\Gls access long{#2}\ifglsxtrininsertinside#3\fi}%
6788       \ifglsxtrininsertinside\else#3\fi
6789     }%
6790     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6791   }%
6792   \gls post link hook
6793 }
```

\GLSxtrlong

```
6794 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}
```

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

```
6795 \newcommand*{\ns@GLSxtrlong}[2] [] {%
6796   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
6797 }
```

Read in the final optional argument:

```
6798 \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).

```
6799   \@glsxtr@record{#1}{#2}{glslink}%
6800   \glsdoifexists{#2}%
6801   {%
6802     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6803     \let\glsxtrifwasfirstuse\@secondoftwo
6804     \let\glsifplural\@secondoftwo
6805     \let\gls caps case\@thirdofthree
6806     \let\glsinsert\@empty
6807     \def\gls custom text{%
6808       \mfirstucMakeUppercase
6809       {\gls long font{\Gls access long{#2}\ifglsxtrininsertinside#3\fi}%

```

```

6810      \ifglstrinsertinside\else#3\fi
6811    }%
6812  }%
6813    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6814  }%
6815  \glspostlinkhook
6816 }

```

Plural short forms:

\glstrshortpl

```

6817 \newrobustcmd*{\glstrshortpl}{\@gls@hyp@opt\ns@glstrshortpl}

```

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

```

6818 \newcommand*{\ns@glstrshortpl}[2][{}]{%
6819   \new@ifnextchar[{\@glstrshortpl{#1}{#2}}{\@glstrshortpl{#1}{#2}[]}%
6820 }

```

Read in the final optional argument:

```

6821 \def\@glstrshortpl#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).

```

6822   \@glstr@record{#1}{#2}{glslink}%
6823   \glsoifexists{#2}%
6824   {%
6825     \glsetabbrvfmt{\glscategory{#2}}%
6826     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6827     \let\glstrifwasfirstuse\@secondoftwo
6828     \let\glssifplural\@firstoftwo
6829     \let\glscapscase\@firstofthree
6830     \let\glinsert\@empty
6831     \def\glscustomtext{%
6832       \glsabbrvfont{\glssaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6833       \ifglstrinsertinside\else#3\fi
6834     }%
6835     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6836   }%
6837   \glspostlinkhook
6838 }

```

\Glsxtrshortpl

```

6839 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}

```

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

```

6840 \newcommand*{\ns@Glsxtrshortpl}[2][{}]{%
6841   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
6842 }

```

Read in the final optional argument:

```

6843 \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).

```

6844 \@glstr@record{#1}{#2}{glslink}%
6845 \glsoifexists{#2}%
6846 {%
6847 \glsetabbrvfmt{\glscategory{#2}}%
6848 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6849 \let\glstrifwasfirstuse\@secondoftwo
6850 \let\gl@ifplural\@firstoftwo
6851 \let\glscapscase\@secondofthree
6852 \let\glinsert\@empty
6853 \def\glscustomtext{%
6854 \glabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6855 \ifglstrinsertinside\else#3\fi
6856 }%
6857 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6858 }%
6859 \glspostlinkhook
6860 }

```

\GLSxtrshortpl

```

6861 \newrobustcmd*{\GLSxtrshortpl}{\@gl@hyp@opt\ns@GLSxtrshortpl}

```

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

```

6862 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
6863 \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6864 }

```

Read in the final optional argument:

```

6865 \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).

```

6866 \@glstr@record{#1}{#2}{glslink}%
6867 \glsoifexists{#2}%
6868 {%
6869 \glsetabbrvfmt{\glscategory{#2}}%
6870 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6871 \let\glstrifwasfirstuse\@secondoftwo
6872 \let\gl@ifplural\@firstoftwo
6873 \let\glscapscase\@thirdofthree
6874 \let\glinsert\@empty
6875 \def\glscustomtext{%
6876 \mfirstucMakeUppercase
6877 {\glabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6878 \ifglstrinsertinside\else#3\fi
6879 }%
6880 }%
6881 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6882 }%

```

```
6883 \glspostlinkhook
6884 }
```

Plural long forms:

\glxstrlongpl

```
6885 \newrobustcmd*{\glxstrlongpl}{\@gls@hyp@opt\@ns@glxstrlongpl}
```

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

```
6886 \newcommand*{\ns@glxstrlongpl}[2] [] {%
6887 \new@ifnextchar[{\@glxstrlongpl{#1}{#2}}{\@glxstrlongpl{#1}{#2} []}%
6888 }
```

Read in the final optional argument:

```
6889 \def\@glxstrlongpl#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).

```
6890 \@glxstr@record{#1}{#2}{glslink}%
6891 \glsdoifexists{#2}%
6892 {%
6893 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6894 \let\glxstrifwasfirstuse\@secondoftwo
6895 \let\gl@ifplural\@firstoftwo
6896 \let\glscapscase\@firstofthree
6897 \let\glinsert\@empty
6898 \def\glscustomtext{%
6899 \glslongfont{\gl@saccesslongpl{#2}\ifglxstrinsertinside#3\fi}%
6900 \ifglxstrinsertinside\else#3\fi
6901 }%
6902 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6903 }%
6904 \glspostlinkhook
6905 }
```

\Glsxtrlongpl

```
6906 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
```

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

```
6907 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
6908 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
6909 }
```

Read in the final optional argument:

```
6910 \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).

```
6911 \@glxstr@record{#1}{#2}{glslink}%
6912 \glsdoifexists{#2}%
6913 {%
```

```

6914 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6915 \let\glxtrifwasfirstuse\@secondoftwo
6916 \let\gl@ifplural\@firstoftwo
6917 \let\glscapscase\@secondofthree
6918 \let\glinsert\@empty
6919 \def\glscustomtext{%
6920   \glslongfont{\gl@saccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
6921   \ifglxtrinsertinside\else#3\fi
6922 }%
6923 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6924 }%
6925 \glspostlinkhook
6926 }

```

`\GLSxtrlongpl`

```

6927 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6928 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
6929   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
6930 }

```

Read in the final optional argument:

```

6931 \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).
6932 \@glxtr@record{#1}{#2}{gl@link}%
6933 \gl@doifexists{#2}%
6934 {%
6935   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6936   \let\glxtrifwasfirstuse\@secondoftwo
6937   \let\gl@ifplural\@firstoftwo
6938   \let\glscapscase\@thirdofthree
6939   \let\glinsert\@empty
6940   \def\glscustomtext{%
6941     \mfirstucMakeUppercase
6942     {\glslongfont{\gl@saccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
6943     \ifglxtrinsertinside\else#3\fi
6944   }%
6945   }%
6946   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6947 }%
6948 \glspostlinkhook
6949 }

```

`\glsetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```

6950 \newcommand*{\glsetabbrvfmt}[1]{%
6951   \ifcsdef{\gl@abbrv@current@#1}%
6952   {\glxtr@applyabbrvfmt{\csname @gl@abbrv@current@#1\endcsname}}%

```

```

6953 {\glxstr@applyabbrvfmt{\@glabbrv@current@abbreviation}}%
6954 }

```

`\glseuseabbrvfont` Provide a way to use the abbreviation font for a given category for arbitrary text.

```

6955 \newrobustcmd*{\glseuseabbrvfont}[2]{\glsetabbrvfmt{#2}\glabbrvfont{#1}}

```

`\glsuselongfont` Provide a way to use the long font for a given category for arbitrary text.

```

6956 \newrobustcmd*{\glsuselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}

```

`\glxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```

6957 \newcommand*{\glxtrgenabbrvfmt}{%
6958   \ifdefempty\glscustomtext
6959   {%
6960     \ifglused\glslabel
6961     {%

```

Subsequent use:

```

6962     \glcifplural
6963     {%

```

Subsequent plural form:

```

6964     \glscapscase
6965     {%

```

Subsequent plural form, don't adjust case:

```

6966     \glxtrsubsequentplfmt{\glslabel}{\glinsert}%
6967     }%
6968     {%

```

Subsequent plural form, make first letter upper case:

```

6969     \Glxtrsubsequentplfmt{\glslabel}{\glinsert}%
6970     }%
6971     {%

```

Subsequent plural form, all caps:

```

6972     \mfirstucMakeUppercase
6973     {\glxtrsubsequentplfmt{\glslabel}{\glinsert}}%
6974     }%
6975     }%
6976     {%

```

Subsequent singular form

```

6977     \glscapscase
6978     {%

```

Subsequent singular form, don't adjust case:

```

6979     \glxtrsubsequentfmt{\glslabel}{\glinsert}%
6980     }%
6981     {%

```


Subsequent singular form, make first letter upper case:

```
6982      \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%  
6983      }%  
6984      {%
```

Subsequent singular form, all caps:

```
6985      \mfirstucMakeUppercase  
6986      {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%  
6987      }%  
6988      }%  
6989      }%  
6990      {%
```

First use:

```
6991      \glsifplural  
6992      {%
```

First use plural form:

```
6993      \glscapscase  
6994      {%
```

First use plural form, don't adjust case:

```
6995      \glsxtrfullplformat{\glslabel}{\glsinsert}%  
6996      }%  
6997      {%
```

First use plural form, make first letter upper case:

```
6998      \Glsxtrfullplformat{\glslabel}{\glsinsert}%  
6999      }%  
7000      {%
```

First use plural form, all caps:

```
7001      \mfirstucMakeUppercase  
7002      {\glsxtrfullplformat{\glslabel}{\glsinsert}}%  
7003      }%  
7004      }%  
7005      {%
```

First use singular form

```
7006      \glscapscase  
7007      {%
```

First use singular form, don't adjust case:

```
7008      \glsxtrfullformat{\glslabel}{\glsinsert}%  
7009      }%  
7010      {%
```

First use singular form, make first letter upper case:

```
7011      \Glsxtrfullformat{\glslabel}{\glsinsert}%  
7012      }%  
7013      {%
```

First use singular form, all caps:

```

7014      \mfirstucMakeUppercase
7015      {\glxstrfullformat{\glslabel}{\glinsert}}}%
7016      }%
7017      }%
7018      }%
7019      }%
7020      {%

```

User supplied text.

```

7021      \glscustomtext
7022      }%
7023 }

```

trsubsequentfmt Subsequent use format (singular no case change).

```

7024 \newcommand*{\glxtrsubsequentfmt}[2]{%
7025   \glabbrvfont{\glaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7026   \ifglxtrininsertinside \else#2\fi
7027 }
7028 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural no case change).

```

7029 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7030   \glabbrvfont{\glaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7031   \ifglxtrininsertinside \else#2\fi
7032 }
7033 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```

7034 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7035   \glabbrvfont{\Glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7036   \ifglxtrininsertinside \else#2\fi
7037 }
7038 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```

7039 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7040   \glabbrvfont{\Glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7041   \ifglxtrininsertinside \else#2\fi
7042 }
7043 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

abbreviationstyle

```

7044 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7045   \ifcsundef{@glabbrv@dispstyle@setup@#2}
7046   {%

```

```

7047 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7048 }%
7049 {%

```

Have abbreviations already been defined for this category?

```

7050 \ifcsstring{@glsabbrv@current@#1}{#2}%
7051 {%

```

Style already set.

```

7052 }%
7053 {%
7054 \def\@glsxtr@dostylewarn{%
7055 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7056 {%
7057 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
7058 style has been switched \MessageBreak
7059 for category ‘#1’, \MessageBreak
7060 but there have already been entries \MessageBreak
7061 defined for this category. Unwanted \MessageBreak
7062 side-effects may result}}%
7063 \@endfortrue
7064 }%
7065 \@glsxtr@dostylewarn

```

Set up the style for the given category.

```

7066 \csdef{@glsabbrv@current@#1}{#2}%
7067 \glsxtr@applyabbrvstyle{#2}%
7068 }%
7069 }%
7070 }

```

applyabbrvstyle Apply the abbreviation style without existence check.

```

7071 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7072 \csuse{@glsabbrv@dispstyle@setup@#1}%
7073 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7074 }

```

r@applyabbrvfmt Only apply the style formats.

```

7075 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7076 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7077 }

```

abbreviationstyle 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.

```

7078 \newcommand*{\newabbreviationstyle}[3]{%
7079 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7080 {%
7081 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7082 defined}{}%

```

```

7083 }%
7084 {%
7085   \csdef{@glsabbrv@dispstyle@setup@#1}{%
Initialise hook to do nothing. The style may change this.
7086     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7087     #2}%
7088   \csdef{@glsabbrv@dispstyle@fmts@#1}{%
Assume in-line form is the same as first use. The style may change this.
7089     \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7090     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7091     \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7092     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
Reset \glxtrsubsequentfmt etc in case a style changes this.
7093     \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
7094     \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
7095     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7096     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7097     #3}%
7098   }%
7099 }

```

breivationstyle

```

7100 \newcommand*{\renewabbreviationstyle}[3]{%
7101   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7102   {%
7103     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7104   }%
7105   {%
7106     \csdef{@glsabbrv@dispstyle@setup@#1}{%
Initialise hook to do nothing. The style may change this.
7107       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7108       #2}%
7109     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
Assume in-line form is the same as first use. The style may change this.
7110       \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7111       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7112       \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7113       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7114       #3}%
7115     }%
7116 }

```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```

7117 \newcommand*{\letabbreviationstyle}[2]{%
7118   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%

```

```

7119 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7120 }

```

```

ecated@abbrstyle \@glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}

```

Define a synonym for a deprecated abbreviation style.

```

7121 \newcommand*{@glxtr@deprecated@abbrstyle}[2]{%
7122   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7123     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7124     \csuse{@glsabbrv@dispstyle@setup@#2}%
7125   }%
7126   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7127 }

```

ecatedAbbrStyle Generate warning for deprecated style use.

```

7128 \newcommand*{@GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7129   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7130   use ‘#2’ instead}%
7131 }

```

eAbbrStyleSetup

```

7132 \newcommand*{@GlsXtrUseAbbrStyleSetup}[1]{%
7133   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7134   {%
7135     \PackageError{glossaries-extra}%
7136     {Unknown abbreviation style definitions ‘#1’}{}%
7137   }%
7138   {%
7139     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7140   }%
7141 }

```

seAbbrStyleFmts

```

7142 \newcommand*{@GlsXtrUseAbbrStyleFmts}[1]{%
7143   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7144   {%
7145     \PackageError{glossaries-extra}%
7146     {Unknown abbreviation style formats ‘#1’}{}%
7147   }%
7148   {%
7149     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7150   }%
7151 }

```

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, `\glstrfullformat` 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.

```
7152 \newif\ifglstrinsertinside
7153 \glstrinsertinsidefalse
```

`trlongshortname`

```
7154 \newcommand*{\glstrlongshortname}{%
7155   \protect\glsabbrvfont{\the\glsshorttok}%
7156 }
```

`long-short`

```
7157 \newabbreviationstyle{long-short}%
7158 {%
7159   \renewcommand*{\CustomAbbreviationFields}{%
7160     name={\glstrlongshortname},
7161     sort={\the\glsshorttok},
7162     first={\protect\glsfirstlongfont{\the\glslongtok}%
7163       \protect\glstrfullsep{\the\glslabeltok}%
7164       \glstrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7165     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7166       \protect\glstrfullsep{\the\glslabeltok}%
7167       \glstrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7168     plural={\protect\glsabbrvfont{\the\glsshortpltok}}},%
7169     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
7170 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7171   \glshasattribute{\the\glslabeltok}{regular}%
7172   {%
7173     \glssetattribute{\the\glslabeltok}{regular}{false}%
7174   }%
7175   {}%
7176 }%
7177 }%
7178 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

7179 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7180 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7181 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7182 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7183 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7184 \renewcommand*{\glxtrfullformat}[2]{%
7185   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7186   \ifglxtrinsertinside\else##2\fi
7187   \glxtrfullsep{##1}%
7188   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7189 }%
7190 \renewcommand*{\glxtrfullplformat}[2]{%
7191   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7192   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7193   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7194 }%
7195 \renewcommand*{\Glsxtrfullformat}[2]{%
7196   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7197   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7198   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7199 }%
7200 \renewcommand*{\Glsxtrfullplformat}[2]{%
7201   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7202   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7203   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7204 }%
7205 }

```

Set this as the default style for general abbreviations:

```

7206 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7207 \newcommand*{\glxtrlongshortdescsort}{%
7208   \expandonce\glxtrorglong\space (\expandonce\glxtrorgshort)%
7209 }

```

ngshortdescname

```

7210 \newcommand*{\glxtrlongshortdescname}{%
7211   \protect\glslongfont{\the\glslongtok}
7212   \glxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7213 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7214 \newabbreviationstyle{long-short-desc}%
7215 {%
7216   \renewcommand*{\CustomAbbreviationFields}{%

```

```

7217   name={\glxtrlongshortdescname},
7218   sort={\glxtrlongshortdescsort},%
7219   first={\protect\glsfirstlongfont{\the\glslongtok}%
7220     \protect\glxtrfullsep{\the\glslabeltok}%
7221     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7222   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7223     \protect\glxtrfullsep{\the\glslabeltok}%
7224     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7225   text={\protect\glsabbrvfont{\the\glsshorttok}},%
7226   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7227 }%

```

Unset the regular attribute if it has been set.

```

7228 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7229   \glshasattribute{\the\glslabeltok}{regular}%
7230   {%
7231     \glissetattribute{\the\glslabeltok}{regular}{false}%
7232   }%
7233   {}%
7234 }%
7235 }%
7236 {%
7237   \GlsXtrUseAbbrStyleFmts{long-short}%
7238 }

```

trshortlongname

```

7239 \newcommand*{\glxtrshortlongname}{%
7240   \protect\glsabbrvfont{\the\glsshorttok}%
7241 }

```

short-long Short form followed by long form in parenthesis on first use.

```

7242 \newabbreviationstyle{short-long}%
7243 {%
7244   \renewcommand*{\CustomAbbreviationFields}{%
7245     name={\glxtrshortlongname},
7246     sort={\the\glsshorttok},
7247     description={\the\glslongtok},%
7248     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7249       \protect\glxtrfullsep{\the\glslabeltok}%
7250       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7251     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7252       \protect\glxtrfullsep{\the\glslabeltok}%
7253       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7254     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.


```

7255 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7256   \glshasattribute{\the\glslabeltok}{regular}}%
7257   {%
7258     \glissetattribute{\the\glslabeltok}{regular}{false}}%
7259   }%
7260   {}%
7261 }%
7262 }%
7263 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7264 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7265 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
7266 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
7267 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
7268 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7269 \renewcommand*{\glsxtrfullformat}[2]{%
7270   \glssfirstabbrvfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7271   \ifglsxtrininsertinside\else##2\fi
7272   \glsxtrfullsep{##1}}%
7273   \glsxtrparen{\glssfirstlongfont{\glssaccesslong{##1}}}%
7274 }%
7275 \renewcommand*{\glsxtrfullplformat}[2]{%
7276   \glssfirstabbrvfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7277   \ifglsxtrininsertinside\else##2\fi
7278   \glsxtrfullsep{##1}}%
7279   \glsxtrparen{\glssfirstlongfont{\glssaccesslongpl{##1}}}%
7280 }%
7281 \renewcommand*{\Glsxtrfullformat}[2]{%
7282   \glssfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7283   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
7284   \glsxtrparen{\glssfirstlongfont{\Glsaccesslong{##1}}}%
7285 }%
7286 \renewcommand*{\Glsxtrfullplformat}[2]{%
7287   \glssfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7288   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%
7289   \glsxtrparen{\glssfirstlongfont{\Glsaccesslongpl{##1}}}%
7290 }%
7291 }

```

ortlongdescsort

```

7292 \newcommand*{\glsxtrshortlongdescsort}{\the\glssshorttok}

```

ortlongdescname

```

7293 \newcommand*{\glsxtrshortlongdescname}{%
7294   \protect\glssabrvfont{\the\glssshorttok}
7295   \glsxtrparen{\protect\glsslongfont{\the\glsslongtok}}}%
7296 }

```

short-long-desc User supplies description. The long form is included in the name.

```
7297 \newabbreviationstyle{short-long-desc}%
7298 {%
7299   \renewcommand*{\CustomAbbreviationFields}{%
7300     name={\glstrshortlongdescname},
7301     sort={\glstrshortlongdescsort},
7302     first={\protect\glstrshorttok}%
7303     \protect\glstrfullsep{\the\glslabeltok}%
7304     \glstrparen{\protect\glstrlongtok}%
7305     firstplural={\protect\glstrshorttok}%
7306     \protect\glstrfullsep{\the\glslabeltok}%
7307     \glstrparen{\protect\glstrlongtok}}},%
7308   text={\protect\glstrshorttok},%
7309   plural={\protect\glstrshorttok}}%
7310 }%
```

Unset the regular attribute if it has been set.

```
7311 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7312   \glshasattribute{\the\glslabeltok}{regular}%
7313   {%
7314     \glissetattribute{\the\glslabeltok}{regular}{false}%
7315   }%
7316   {%
7317   }%
7318 }%
7319 {%
7320   \GlsXtrUseAbbrStyleFmts{short-long}%
7321 }
```

ongfootnotefont Only used by the “footnote” styles.

```
7322 \newcommand*{\glstrlongfootnotefont}[1]{\glstrlongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
7323 \newcommand*{\glstrlongfootnotefont}[1]{\glstrlongdefaultfont{#1}}%
```

xtrabbrvfootnote `\glstrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument *<long>* includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\glstr` or `\glstrpl`).

```
7324 \newcommand*{\glstrabbrvfootnote}[2]{\footnote{#2}}
```

xtrfootnotename

```
7325 \newcommand*{\glxtrfootnotename}{%
7326   \protect\glsabbrvfont{\the\glsshorttok}%
7327 }
```

footnote Short form followed by long form in footnote on first use.

```
7328 \newabbreviationstyle{footnote}%
7329 {%
7330   \renewcommand*{\CustomAbbreviationFields}{%
7331     name={\glxtrfootnotename},
7332     sort={\the\glsshorttok},
7333     description={\the\glslongtok},%
7334     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7335       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7336       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7337     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7338       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7339       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
7340     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7341 }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
7341 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7342   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7343   \glsattribute{\the\glslabeltok}{regular}%
7344   {%
7345     \glssetattribute{\the\glslabeltok}{regular}{false}%
7346   }%
7347   {}%
7348 }%
7349 }%
7350 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7351 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7352 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7353 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7354 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7355 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
7356 \renewcommand*{\glsxtrfullformat}[2]{%
7357   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrinsetinside##2\fi}%
7358   \ifglsxtrinsetinside\else##2\fi
7359   \protect\glsxtrabbrvfootnote{##1}%
7360   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7361 }%
7362 \renewcommand*{\glsxtrfullplformat}[2]{%
7363   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrinsetinside##2\fi}%
7364   \ifglsxtrinsetinside\else##2\fi
7365   \protect\glsxtrabbrvfootnote{##1}%
7366   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7367 }
```

```

7363 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7364 \ifglxtrinsertinside\else##2\fi
7365 \protect\glxtrabbrvfootnote{##1}%
7366 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7367 }%
7368 \renewcommand*{\Glsxtrfullformat}[2]{%
7369 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7370 \ifglxtrinsertinside\else##2\fi
7371 \protect\glxtrabbrvfootnote{##1}%
7372 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7373 }%
7374 \renewcommand*{\Glsxtrfullplformat}[2]{%
7375 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7376 \ifglxtrinsertinside\else##2\fi
7377 \protect\glxtrabbrvfootnote{##1}%
7378 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7379 }%

```

The first use full form and the inline full form use the short (long) style.

```

7380 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7381 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7382 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7383 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7384 }%
7385 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7386 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7387 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7388 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7389 }%
7390 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7391 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7392 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7393 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7394 }%
7395 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7396 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7397 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7398 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7399 }%
7400 }

```

short-footnote

```
7401 \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.

```

7402 \newabbreviationstyle{postfootnote}%
7403 {}

```

```

7404 \renewcommand*{\CustomAbbreviationFields}{%
7405     name={\glxtrfootnotename},
7406     sort={\the\glsshorttok},
7407     description={\the\glslongtok},%
7408     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7409     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7410     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

7411 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7412     \csdef{glxtrpostlink\glscategorylabel}{%
7413         \glxtrifwasfirstuse
7414         {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7415             \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
7416             {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
7417         }%
7418     }%
7419 }%
7420 \glsattribute{\the\glslabeltok}{regular}%
7421 {%
7422     \glssetattribute{\the\glslabeltok}{regular}{false}%
7423     }%
7424     }%
7425 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7426 \renewcommand*{\glxtrsetupfulldefs}{%
7427     \let\glxtrifwasfirstuse\@secondoftwo
7428 }%
7429 }%
7430 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7431 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7432 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7433 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7434 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7435 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

7436 \renewcommand*{\glxtrfullformat}[2]{%
7437     \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinertinside##2\fi}%
7438     \ifglxtrinertinside\else##2\fi
7439 }%
7440 \renewcommand*{\glxtrfullplformat}[2]{%

```

```

7441 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7442 \ifglxtrinsertinside\else##2\fi
7443 }%
7444 \renewcommand*{\Glsxtrfullformat}[2]{%
7445 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7446 \ifglxtrinsertinside\else##2\fi
7447 }%
7448 \renewcommand*{\Glsxtrfullplformat}[2]{%
7449 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7450 \ifglxtrinsertinside\else##2\fi
7451 }%

```

The first use full form and the inline full form use the short (long) style.

```

7452 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7453 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7454 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7455 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7456 }%
7457 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7458 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7459 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7460 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7461 }%
7462 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7463 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7464 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7465 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7466 }%
7467 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7468 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7469 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7470 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7471 }%
7472 }

```

rt-postfootnote

```

7473 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

shortnolongname

```

7474 \newcommand*{\glsxtrshortnolongname}{%
7475 \protect\glsabbrvfont{\the\glsshorttok}%
7476 }

```

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.

```

7477 \newabbreviationstyle{short}%
7478 {}

```

```

7479 \renewcommand*{\CustomAbbreviationFields}{%
7480   name={\glxtrshortnolongname},
7481   sort={\the\glsshorttok},
7482   first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7483   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7484   text={\protect\glsabbrvfont{\the\glsshorttok}},
7485   plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7486   description={\the\glslongtok}}%
7487 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7488   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7489 }%
7490 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7491 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7492 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7493 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7494 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7495 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7496 \renewcommand*{\glxtrinlinefullformat}[2]{%
7497   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7498   \ifglxtrininsertinside##2\fi}%
7499   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7500   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7501 }%
7502 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7503   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7504   \ifglxtrininsertinside##2\fi}%
7505   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7506   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7507 }%
7508 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7509   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7510   \ifglxtrininsertinside##2\fi}%
7511   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7512   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7513 }%
7514 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7515   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7516   \ifglxtrininsertinside##2\fi}%
7517   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7518   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7519 }%

```

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.

```

7520 \renewcommand*{\glxtrfullformat}[2]{%
7521   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
7522   \ifglxtrininsertinside\else##2\fi

```

```

7523 }%
7524 \renewcommand*{\glxtrfullplformat}[2]{%
7525   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7526   \ifglxtrininsertinside\else##2\fi
7527 }%
7528 \renewcommand*{\Glsxtrfullformat}[2]{%
7529   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7530   \ifglxtrininsertinside\else##2\fi
7531 }%
7532 \renewcommand*{\Glsxtrfullplformat}[2]{%
7533   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7534   \ifglxtrininsertinside\else##2\fi
7535 }%
7536 }

```

Set this as the default style for acronyms:

```

7537 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7538 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7539 \newabbreviationstyle{short-nolong-noreg}%
7540 {%
7541   \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7542 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7543   \glshasattribute{\the\glslabeltok}{regular}%
7544   {%
7545     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7546   }%
7547   {}}%
7548 }%
7549 }%
7550 {%
7551   \GlsXtrUseAbbrStyleFmts{short-nolong}%
7552 }

```

trshortdescname

```

7553 \newcommand*{\glxtrshortdescname}{%
7554   \protect\glssabbrvfont{\the\glssshorttok}%
7555 }

```

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.

```

7556 \newabbreviationstyle{short-desc}%
7557 {%
7558   \renewcommand*{\CustomAbbreviationFields}{%

```



```

7559     name={\glxtrshortdescname},
7560     sort={\the\glsshorttok},
7561     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7562     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7563     text={\protect\glsabbrvfont{\the\glsshorttok}},
7564     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7565     description={\the\glslongtok}}%
7566 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7567   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7568 }%
7569 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7570 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7571 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7572 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7573 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7574 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7575 \renewcommand*{\glxtrinlinefullformat}[2]{%
7576   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7577   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7578   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7579 }%
7580 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7581   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7582   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7583   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7584 }%
7585 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7586   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7587   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7588   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7589 }%
7590 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7591   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7592   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7593   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7594 }%

```

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.

```

7595 \renewcommand*{\glxtrfullformat}[2]{%
7596   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7597   \ifglxtrininsertinside\else##2\fi
7598 }%
7599 \renewcommand*{\glxtrfullplformat}[2]{%
7600   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7601   \ifglxtrininsertinside\else##2\fi
7602 }%

```

```

7603 \renewcommand*{\Glsxtrfullformat}[2]{%
7604   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7605   \ifglxtrinsertinside\else##2\fi
7606 }%
7607 \renewcommand*{\Glsxtrfullplformat}[2]{%
7608   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7609   \ifglxtrinsertinside\else##2\fi
7610 }%
7611 }

```

ort-nolong-desc

```
7612 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7613 \newabbreviationstyle{short-nolong-desc-noreg}%
7614 {%
7615   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
       Unset the regular attribute if it has been set.
7616   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7617     \glshasattribute{\the\glslabeltok}{regular}%
7618     {%
7619       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7620     }%
7621   }%
7622 }%
7623 }%
7624 {%
7625   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7626 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7627 \newabbreviationstyle{nolong-short}%
7628 {%
7629   \GlsXtrUseAbbrStyleSetup{short-nolong}%
7630 }%
7631 {%
7632   \GlsXtrUseAbbrStyleFmts{short-nolong}%
       The inline full form displays the long form followed by the short form in parentheses.
7633   \renewcommand*{\glxtrinlinefullformat}[2]{%
7634     \protect\glsfirstlongfont{\glsaccesslong{##1}%
7635       \ifglxtrinsertinside##2\fi}%
7636     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7637     \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7638   }%
7639   \renewcommand*{\glxtrinlinefullplformat}[2]{%
7640     \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
7641       \ifglxtrinsertinside##2\fi}%

```

```

7642 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7643 \glxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
7644 }%
7645 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7646 \protect\glsfirstlongfont{\glssaccesslong{##1}%
7647 \ifglxtrinsertinside##2\fi}%
7648 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7649 \glxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}%
7650 }%
7651 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7652 \protect\glsfirstlongfont{\glssaccesslongpl{##1}%
7653 \ifglxtrinsertinside##2\fi}%
7654 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7655 \glxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
7656 }%
7657 }

```

ong-short-noreg Like **nolong-short** but doesn't set the regular attribute.

```

7658 \newabbreviationstyle{nolong-short-noreg}%
7659 {%
7660 \GlsXtrUseAbbrStyleSetup{nolong-short}%
7661 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7662 \glshasattribute{\the\glslabeltok}{regular}%
7663 {%
7664 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7665 }%
7666 {}%
7667 }%
7668 }%
7669 {%
7670 \GlsXtrUseAbbrStyleFmts{nolong-short}%
7671 }

```

noshortdescname

```

7672 \newcommand*{\glxtrlongnoshortdescname}{%
7673 \protect\glslongfont{\the\glslongtok}%
7674 }

```

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.

```

7675 \newabbreviationstyle{long-desc}%
7676 {%
7677 \renewcommand*{\CustomAbbreviationFields}{%
7678 name={\glxtrlongnoshortdescname},
7679 sort={\the\glslongtok},
7680 first={\protect\glsfirstlongfont{\the\glslongtok}},

```

```

7681     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7682     text={\glslongfont{\the\glslongtok}},
7683     plural={\glslongfont{\the\glslongpltok}}}%
7684 }%
7685 \renewcommand*\GlsXtrPostNewAbbreviation{%
7686   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7687 }%
7688 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7689 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7690 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7691 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7692 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7693 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7694 \renewcommand*\glsxtrsubsequentfmt}[2]{%
7695   \glslongfont{\glsaccesslong{##1}}\ifglsxtrininsertinside ##2\fi}%
7696   \ifglsxtrininsertinside \else##2\fi
7697 }%
7698 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
7699   \glslongfont{\glsaccesslongpl{##1}}\ifglsxtrininsertinside ##2\fi}%
7700   \ifglsxtrininsertinside \else##2\fi
7701 }%
7702 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
7703   \glslongfont{\Glsaccesslong{##1}}\ifglsxtrininsertinside ##2\fi}%
7704   \ifglsxtrininsertinside \else##2\fi
7705 }%
7706 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
7707   \glslongfont{\Glsaccesslongpl{##1}}\ifglsxtrininsertinside ##2\fi}%
7708   \ifglsxtrininsertinside \else##2\fi
7709 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7710 \renewcommand*\glsxtrinlinefullformat}[2]{%
7711   \glsfirstlongfont{\glsaccesslong{##1}}\ifglsxtrininsertinside##2\fi}%
7712   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7713   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7714 }%
7715 \renewcommand*\glsxtrinlinefullplformat}[2]{%
7716   \glsfirstlongfont{\glsaccesslongpl{##1}}\ifglsxtrininsertinside##2\fi}%
7717   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7718   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7719 }%
7720 \renewcommand*\Glsxtrinlinefullformat}[2]{%
7721   \glsfirstlongfont{\Glsaccesslong{##1}}\ifglsxtrininsertinside##2\fi}%
7722   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7723   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7724 }%
7725 \renewcommand*\Glsxtrinlinefullplformat}[2]{%

```

```

7726 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7727 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7728 \glxtrparen{\protect\glssfirstabbrfont{\Glsaccessshortpl{##1}}}%
7729 }%

```

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.

```

7730 \renewcommand*{\glxtrfullformat}[2]{%
7731 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7732 \ifglxtrinsertinside\else##2\fi
7733 }%
7734 \renewcommand*{\glxtrfullplformat}[2]{%
7735 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7736 \ifglxtrinsertinside\else##2\fi
7737 }%
7738 \renewcommand*{\Glsxtrfullformat}[2]{%
7739 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7740 \ifglxtrinsertinside\else##2\fi
7741 }%
7742 \renewcommand*{\Glsxtrfullplformat}[2]{%
7743 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7744 \ifglxtrinsertinside\else##2\fi
7745 }%
7746 }

```

`ng-noshort-desc` Provide a synonym that matches similar styles.

```

7747 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

`short-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```

7748 \newabbreviationstyle{long-noshort-desc-noreg}%
7749 {%
7750 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

Unset the regular attribute if it has been set.

```

7751 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7752 \glshasattribute{\the\glslabeltok}{regular}%
7753 {%
7754 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7755 }%
7756 {}%
7757 }%
7758 }%
7759 {%
7760 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7761 }

```

`longnoshortname`

```

7762 \newcommand*{\glxtrlongnoshortname}{%
7763 \protect\glsabbrfont{\the\glssshorttok}%
7764 }

```

`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.

```

7765 \newabbreviationstyle{long}%
7766 {%
7767   \renewcommand*{\CustomAbbreviationFields}{%
7768     name={\glstrlongnoshortname},
7769     sort={\the\glsshorttok},
7770     first={\protect\glfirstlongfont{\the\glslongtok}},
7771     firstplural={\protect\glfirstlongfont{\the\glslongpltok}},
7772     text={\glslongfont{\the\glslongtok}},
7773     plural={\glslongfont{\the\glslongpltok}},%
7774     description={\the\glslongtok}%
7775   }%
7776   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7777     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
7778 }%
7779 {%
7780   \GlsXtrUseAbbrStyleFmts{long-desc}%
7781 }
```

`long-noshort` Provide a synonym that matches similar styles.

```

7782 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```

7783 \newabbreviationstyle{long-noshort-noreg}%
7784 {%
7785   \GlsXtrUseAbbrStyleSetup{long-noshort}%
7786   Unset the regular attribute if it has been set.
7786   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7787     \glshasattribute{\the\glslabeltok}{regular}%
7788     {%
7789       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7790     }%
7791     {}%
7792   }%
7793 }%
7794 {%
7795   \GlsXtrUseAbbrStyleFmts{long-noshort}%
7796 }
```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glstrscfont` Maintained for backward-compatibility.

```

7797 \newcommand*{\glstrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7798 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`\glsxtrfirstscfont` Maintained for backward-compatibility.

```
7799 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{##1}}
```

`\glsfirstabbrvscfont` Added for consistent naming.

```
7800 \newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
7801 \newcommand*{\glsxtrscsuffix}{\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
7802 \newabbreviationstyle{long-short-sc}%
7803 {%
7804   \renewcommand*{\CustomAbbreviationFields}{%
7805     name={\glsxtrlongshortname},
7806     sort={\the\glsshorttok},
7807     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7808       \protect\glsxtrfullsep{\the\glslabeltok}%
7809       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7810     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7811       \protect\glsxtrfullsep{\the\glslabeltok}%
7812       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7813     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
7814     description={\the\glslongtok}}%
7815   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7816     \glsattribute{\the\glslabeltok}{regular}%
7817     {%
7818       \glssetattribute{\the\glslabeltok}{regular}{false}%
7819     }%
7820   }%
7821 }%
7822 }%
7823 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
7824 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
7825 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7826 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
7827 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7828 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7829 \renewcommand*{\glsxtrfullformat}[2]{%
7830   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinertinside##2\fi}%

```

```

7831 \ifglxtrinsertinside\else##2\fi
7832 \glxtrfullsep{##1}%
7833 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7834 }%
7835 \renewcommand*{\glxtrfullplformat}[2]{%
7836 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7837 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7838 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7839 }%
7840 \renewcommand*{\Glsxtrfullformat}[2]{%
7841 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7842 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7843 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7844 }%
7845 \renewcommand*{\Glsxtrfullplformat}[2]{%
7846 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7847 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7848 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7849 }%
7850 }

```

g-short-sc-desc

```

7851 \newabbreviationstyle{long-short-sc-desc}%
7852 {%
7853 \renewcommand*{\CustomAbbreviationFields}{%
7854 name={\glxtrlongshortdescname},
7855 sort={\glxtrlongshortdescsort},%
7856 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7857 \protect\glxtrfullsep{\the\glslabeltok}%
7858 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7859 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7860 \protect\glxtrfullsep{\the\glslabeltok}%
7861 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7862 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7863 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7864 }%

```

Unset the regular attribute if it has been set.

```

7865 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7866 \glshasattribute{\the\glslabeltok}{regular}%
7867 {%
7868 \glssetattribute{\the\glslabeltok}{regular}{false}%
7869 }%
7870 {}%
7871 }%
7872 }%
7873 {%

```

As long-short-sc style:

```

7874 \GlsXtrUseAbbrStyleFmts{long-short-sc}%

```


7875 }

Now the short (long) version

```
7876 \newabbreviationstyle{short-sc-long}%
7877 {%
7878   \renewcommand*{\CustomAbbreviationFields}{%
7879     name={\glxtrshortlongname},
7880     sort={\the\glsshorttok},
7881     description={\the\glslongtok},%
7882     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7883       \protect\glxtrfullsep{\the\glslabeltok}%
7884       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7885     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7886       \protect\glxtrfullsep{\the\glslabeltok}%
7887       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7888     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7889 }
```

Unset the regular attribute if it has been set.

```
7889 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7890   \glshasattribute{\the\glslabeltok}{regular}%
7891   {%
7892     \glissetattribute{\the\glslabeltok}{regular}{false}%
7893   }%
7894   {}%
7895 }%
7896 }%
7897 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
7898 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7899 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7900 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7901 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7902 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7903 \renewcommand*{\glxtrfullformat}[2]{%
7904   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7905   \ifglxtrininsertinside\else##2\fi
7906   \glxtrfullsep{##1}%
7907   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7908 }%
7909 \renewcommand*{\glxtrfullplformat}[2]{%
7910   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7911   \ifglxtrininsertinside\else##2\fi
7912   \glxtrfullsep{##1}%
7913   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7914 }%
7915 \renewcommand*{\Glsxtrfullformat}[2]{%
7916   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7917   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7918 }
```

```

7918 \glstrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
7919 }%
7920 \renewcommand*{\Glsxtrfullplformat}[2]{%
7921 \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
7922 \ifglstrinsertinside\else##2\fi\glxtrfullsep{##1}%
7923 \glstrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
7924 }%
7925 }

```

As before but user provides description

```

7926 \newabbreviationstyle{short-sc-long-desc}%
7927 {%
7928 \renewcommand*{\CustomAbbreviationFields}{%
7929 name={\glstrshortlongdescname},
7930 sort={\glstrshortlongdescsort},
7931 first={\protect\glfirstabbrvscfont{\the\glsshorttok}%
7932 \protect\glxtrfullsep{\the\glslabeltok}%
7933 \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongtok}}},%
7934 firstplural={\protect\glfirstabbrvscfont{\the\glsshortpltok}%
7935 \protect\glxtrfullsep{\the\glslabeltok}%
7936 \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongpltok}}},%
7937 text={\protect\glabbrvscfont{\the\glsshorttok}},%
7938 plural={\protect\glabbrvscfont{\the\glsshortpltok}}}%
7939 }%

```

Unset the regular attribute if it has been set.

```

7940 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7941 \glshasattribute{\the\glslabeltok}{regular}%
7942 {%
7943 \glissetattribute{\the\glslabeltok}{regular}{false}%
7944 }%
7945 {}%
7946 }%
7947 }%
7948 {%

```

As short-sc-long style:

```

7949 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
7950 }

```

short-sc

```

7951 \newabbreviationstyle{short-sc}%
7952 {%
7953 \renewcommand*{\CustomAbbreviationFields}{%
7954 name={\glstrshortnolongname},
7955 sort={\the\glsshorttok},
7956 first={\protect\glfirstabbrvscfont{\the\glsshorttok}},
7957 firstplural={\protect\glfirstabbrvscfont{\the\glsshortpltok}},
7958 text={\protect\glabbrvscfont{\the\glsshorttok}},
7959 plural={\protect\glabbrvscfont{\the\glsshortpltok}},

```

```

7960   description={\the\glslongtok}}%
7961 \renewcommand*\GlsXtrPostNewAbbreviation}{%
7962   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7963 }%
7964 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7965 \renewcommand*\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
7966 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7967 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
7968 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7969 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7970 \renewcommand*\glsxtrinlinefullformat}[2]{%
7971   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
7972   \ifglsxtrininsertinside##2\fi}%
7973   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7974   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7975 }%
7976 \renewcommand*\glsxtrinlinefullplformat}[2]{%
7977   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
7978   \ifglsxtrininsertinside##2\fi}%
7979   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7980   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7981 }%
7982 \renewcommand*\Glsxtrinlinefullformat}[2]{%
7983   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
7984   \ifglsxtrininsertinside##2\fi}%
7985   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7986   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7987 }%
7988 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
7989   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}%
7990   \ifglsxtrininsertinside##2\fi}%
7991   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7992   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7993 }%

```

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.

```

7994 \renewcommand*\glsxtrfullformat}[2]{%
7995   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
7996   \ifglsxtrininsertinside\else##2\fi
7997 }%
7998 \renewcommand*\glsxtrfullplformat}[2]{%
7999   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8000   \ifglsxtrininsertinside\else##2\fi
8001 }%
8002 \renewcommand*\Glsxtrfullformat}[2]{%

```

```

8003 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8004 \ifglxtrinsertinside\else##2\fi
8005 }%
8006 \renewcommand*{\Glsxtrfullplformat}[2]{%
8007 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8008 \ifglxtrinsertinside\else##2\fi
8009 }%
8010 }

```

short-sc-nolong

```
8011 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```

8012 \newabbreviationstyle{short-sc-desc}%
8013 {%
8014 \renewcommand*{\CustomAbbreviationFields}{%
8015 name={\glsxtrshortdescname},
8016 sort={\the\glsshorttok},
8017 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8018 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8019 text={\protect\glsabbrvscfont{\the\glsshorttok}},
8020 plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8021 description={\the\glslongtok}}%
8022 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8023 \glsetattribute{\the\glslabeltok}{regular}{true}}%
8024 }%
8025 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8026 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8027 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8028 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8029 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8030 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8031 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8032 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8033 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8034 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8035 }%
8036 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8037 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8038 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8039 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8040 }%
8041 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8042 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8043 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8044 \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%

```

```

8045 }%
8046 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8047   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8048   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8049   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8050 }%

```

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.

```

8051 \renewcommand*{\glsxtrfullformat}[2]{%
8052   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8053   \ifglxtrinsertinside\else##2\fi
8054 }%
8055 \renewcommand*{\glsxtrfullplformat}[2]{%
8056   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8057   \ifglxtrinsertinside\else##2\fi
8058 }%
8059 \renewcommand*{\Glsxtrfullformat}[2]{%
8060   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8061   \ifglxtrinsertinside\else##2\fi
8062 }%
8063 \renewcommand*{\Glsxtrfullplformat}[2]{%
8064   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8065   \ifglxtrinsertinside\else##2\fi
8066 }%
8067 }

```

-sc-nolong-desc

```

8068 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

8069 \newabbreviationstyle{nolong-short-sc}{%
8070 {%
8071   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8072 }%
8073 {%
8074   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8075 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8076   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8077     \ifglxtrinsertinside##2\fi}%
8078   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8079   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8080 }%
8081 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8082   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8083     \ifglxtrinsertinside##2\fi}%
8084   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8085   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%

```

```

8086 }%
8087 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8088   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8089     \ifglxtrinsertinside##2\fi}%
8090   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8091   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8092 }%
8093 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8094   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8095     \ifglxtrinsertinside##2\fi}%
8096   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8097   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8098 }%
8099 }

```

`long-noshort-sc` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

8100 \newabbreviationstyle{long-noshort-sc}%
8101 {%
8102   \renewcommand*{\CustomAbbreviationFields}{%
8103     name={\glxtrlongnoshortname},
8104     sort={\the\glsshorttok},
8105     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8106     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8107     text={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8108     plural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},%
8109     description={\the\glslongtok}%
8110   }%
8111   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8112     \glsetattribute{\the\glslabeltok}{regular}{true}}%
8113 }%
8114 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8115 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8116 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8117 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8118 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8119 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8120 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8121   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8122   \ifglxtrinsertinside \else##2\fi
8123 }%
8124 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8125   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8126   \ifglxtrinsertinside \else##2\fi
8127 }%
8128 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

8129 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8130 \ifglxtrinsertinside \else##2\fi
8131 }%
8132 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8133 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8134 \ifglxtrinsertinside \else##2\fi
8135 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8136 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8137 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8138 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8139 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8140 }%
8141 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8142 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8143 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8144 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8145 }%
8146 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8147 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8148 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8149 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8150 }%
8151 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8152 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8153 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8154 \glsxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8155 }%

```

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.

```

8156 \renewcommand*{\glsxtrfullformat}[2]{%
8157 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8158 \ifglxtrinsertinside\else##2\fi
8159 }%
8160 \renewcommand*{\glsxtrfullplformat}[2]{%
8161 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8162 \ifglxtrinsertinside\else##2\fi
8163 }%
8164 \renewcommand*{\Glsxtrfullformat}[2]{%
8165 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8166 \ifglxtrinsertinside\else##2\fi
8167 }%
8168 \renewcommand*{\Glsxtrfullplformat}[2]{%
8169 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8170 \ifglxtrinsertinside\else##2\fi
8171 }%
8172 }

```

long-sc Backward compatibility:

```
8173 \@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.

```
8174 \newabbreviationstyle{long-noshort-sc-desc}%
```

```
8175 {%
```

```
8176 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

```
8177 }%
```

```
8178 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8179 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
```

```
8180 \renewcommand*{\glabbrvfont[1]{\glabbrvscfont{##1}}}%
```

```
8181 \renewcommand*{\glfirstabbrvfont[1]{\glfirstabbrvscfont{##1}}}%
```

```
8182 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
```

```
8183 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8184 \renewcommand*{\glxtrsubsequentfmt}[2]{%
```

```
8185 \glslongdefaultfont{\glaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
```

```
8186 \ifglxtrinsertinside \else##2\fi
```

```
8187 }%
```

```
8188 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
```

```
8189 \glslongdefaultfont{\glaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
```

```
8190 \ifglxtrinsertinside \else##2\fi
```

```
8191 }%
```

```
8192 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
```

```
8193 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
```

```
8194 \ifglxtrinsertinside \else##2\fi
```

```
8195 }%
```

```
8196 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
```

```
8197 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
```

```
8198 \ifglxtrinsertinside \else##2\fi
```

```
8199 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
8200 \renewcommand*{\glxtrinlinefullformat}[2]{%
```

```
8201 \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
```

```
8202 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
```

```
8203 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshort{##1}}}%
```

```
8204 }%
```

```
8205 \renewcommand*{\glxtrinlinefullplformat}[2]{%
```

```
8206 \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
```

```
8207 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
```

```
8208 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshortpl{##1}}}%
```

```
8209 }%
```

```
8210 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
```

```
8211 \glfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
```

```
8212 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
```



```

8213 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8214 }%
8215 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8216 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8217 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8218 \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8219 }%

```

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.

```

8220 \renewcommand*{\glsxtrfullformat}[2]{%
8221 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8222 \ifglsxtrinsertinside\else##2\fi
8223 }%
8224 \renewcommand*{\glsxtrfullplformat}[2]{%
8225 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8226 \ifglsxtrinsertinside\else##2\fi
8227 }%
8228 \renewcommand*{\Glsxtrfullformat}[2]{%
8229 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8230 \ifglsxtrinsertinside\else##2\fi
8231 }%
8232 \renewcommand*{\Glsxtrfullplformat}[2]{%
8233 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8234 \ifglsxtrinsertinside\else##2\fi
8235 }%
8236 }

```

long-desc-sc Backward compatibility:

```

8237 \@glsxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

8238 \newabbreviationstyle{short-sc-footnote}%
8239 {%
8240 \renewcommand*{\CustomAbbreviationFields}{%
8241 name={\glsxtrfootnotename},
8242 sort={\the\glsshorttok},
8243 description={\the\glslongtok},%
8244 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8245 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
8246 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8247 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8248 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
8249 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8250 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.

```

8251 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8252 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%

```

```

8253 \glshasattribute{\the\glslabeltok}{regular}%
8254 {%
8255 \glissetattribute{\the\glslabeltok}{regular}{false}%
8256 }%
8257 {}%
8258 }%
8259 }%
8260 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8261 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8262 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8263 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8264 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8265 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8266 \renewcommand*\glsxtrfullformat[2]{%
8267 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8268 \ifglsxtrininsertinside\else##2\fi
8269 \protect\glsxtrabbrvfootnote{##1}%
8270 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8271 }%
8272 \renewcommand*\glsxtrfullplformat[2]{%
8273 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8274 \ifglsxtrininsertinside\else##2\fi
8275 \protect\glsxtrabbrvfootnote{##1}%
8276 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8277 }%
8278 \renewcommand*\Glsxtrfullformat[2]{%
8279 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8280 \ifglsxtrininsertinside\else##2\fi
8281 \protect\glsxtrabbrvfootnote{##1}%
8282 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8283 }%
8284 \renewcommand*\Glsxtrfullplformat[2]{%
8285 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8286 \ifglsxtrininsertinside\else##2\fi
8287 \protect\glsxtrabbrvfootnote{##1}%
8288 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8289 }%

```

The first use full form and the inline full form use the short (long) style.

```

8290 \renewcommand*\glsxtrininlinefullformat[2]{%
8291 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8292 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8293 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8294 }%
8295 \renewcommand*\glsxtrininlinefullplformat[2]{%
8296 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8297 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8298 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8299 }%
8300 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8301 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8302 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8303 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
8304 }%
8305 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8306 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8307 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8308 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8309 }%
8310 }

```

footnote-sc Backward compatibility:

```

8311 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8312 \newabbreviationstyle{short-sc-postfootnote}%
8313 {%
8314 \renewcommand*{\CustomAbbreviationFields}{%
8315 name={\glxtrfootnotename},
8316 sort={\the\glsshorttok},
8317 description={\the\glslongtok},%
8318 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8319 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8320 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.

```

8321 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8322 \csdef{glxtrpostlink\glscategorylabel}{%
8323 \glxtrifwasfirstuse
8324 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8325 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8326 {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
8327 }%
8328 {}%
8329 }%
8330 \glshasattribute{\the\glslabeltok}{regular}%
8331 {%
8332 \glissetattribute{\the\glslabeltok}{regular}{false}%
8333 }%
8334 {}%
8335 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8336 \renewcommand*{\glxtrsetupfulldefs}{%
8337   \let\glxtrifwasfirstuse\@secondoftwo
8338 }%
8339 }%
8340 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8341 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8342 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8343 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8344 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8345 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8346 \renewcommand*{\glxtrfullformat}[2]{%
8347   \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8348   \ifglxtrininsertinside\else##2\fi
8349 }%
8350 \renewcommand*{\glxtrfullplformat}[2]{%
8351   \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8352   \ifglxtrininsertinside\else##2\fi
8353 }%
8354 \renewcommand*{\Glsxtrfullformat}[2]{%
8355   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8356   \ifglxtrininsertinside\else##2\fi
8357 }%
8358 \renewcommand*{\Glsxtrfullplformat}[2]{%
8359   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8360   \ifglxtrininsertinside\else##2\fi
8361 }%
```

The first use full form and the inline full form use the short (long) style.

```
8362 \renewcommand*{\glxtrininlinefullformat}[2]{%
8363   \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8364   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8365   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
8366 }%
8367 \renewcommand*{\glxtrininlinefullplformat}[2]{%
8368   \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8369   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8370   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8371 }%
8372 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
8373   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8374   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8375   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8376 }%
8377 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```

```

8378 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8379 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8380 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8381 }%
8382 }

```

postfootnote-sc Backward compatibility:

```

8383 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the `relese` package, which must be loaded by the user. These styles all use:

`\glxtrsmfont` Maintained for backward compatibility.

```

8384 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

```

`\glsabbrvsmfont` Added for consistent naming.

```

8385 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

```

`sxtrfirstsmfont` Maintained for backward compatibility.

```

8386 \newcommand*{\sxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

```

`irstabbrvsmfont` Added for consistent naming.

```

8387 \newcommand*{\glsfirstabbrvsmfont}{\sxtrfirstsmfont}

```

and for the default short form suffix:

`\glxtrsmsuffix`

```

8388 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}

```

`long-short-sm`

```

8389 \newabbreviationstyle{long-short-sm}%
8390 {%
8391 \renewcommand*{\CustomAbbreviationFields}{%
8392   name={\glxtrlongshortname},
8393   sort={\the\glsshorttok},
8394   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8395     \protect\glxtrfullsep{\the\glslabeltok}%
8396     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8397   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8398     \protect\glxtrfullsep{\the\glslabeltok}%
8399     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8400   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},%
8401   description={\the\glslongtok}}%
8402 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8403   \glshasattribute{\the\glslabeltok}{regular}%
8404   {%

```

```

8405     \glsetattribute{\the\glslabeltok}{regular}{false}%
8406   }%
8407   {}%
8408 }%
8409}%
8410{%
8411  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8412  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8413  \renewcommand*\abbrvpluralsuffix{\protect\glsxtrmsuffix}%

```

Use the default long fonts.

```

8414  \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8415  \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8416  \renewcommand*\glsxtrfullformat[2]{%
8417    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8418    \ifglsxtrininsertinside\else##2\fi
8419    \glsxtrfullsep{##1}%
8420    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8421  }%
8422  \renewcommand*\glsxtrfullplformat[2]{%
8423    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8424    \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8425    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8426  }%
8427  \renewcommand*\Glsxtrfullformat[2]{%
8428    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8429    \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8430    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8431  }%
8432  \renewcommand*\Glsxtrfullplformat[2]{%
8433    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8434    \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8435    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8436  }%
8437}

```

g-short-sm-desc

```

8438 \newabbreviationstyle{long-short-sm-desc}%
8439 {%
8440   \renewcommand*\CustomAbbreviationFields{%
8441     name={\glsxtrlongshortdescname},
8442     sort={\glsxtrlongshortdescsort},%
8443     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8444       \protect\glsxtrfullsep{\the\glslabeltok}%
8445       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8446     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8447       \protect\glsxtrfullsep{\the\glslabeltok}%
8448       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%

```

```

8449     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8450     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
8451 }%

```

Unset the regular attribute if it has been set.

```

8452 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8453   \glshasattribute{\the\glslabeltok}{regular}%
8454   {%
8455     \glissetattribute{\the\glslabeltok}{regular}{false}%
8456   }%
8457   {}}%
8458 }%
8459 }%
8460 {%

```

As long-short-sm style:

```

8461 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8462 }

```

short-sm-long Now the short (long) version

```

8463 \newabbreviationstyle{short-sm-long}%
8464 {%
8465   \renewcommand*{\CustomAbbreviationFields}{%
8466     name={\glsxtrshortlongname},
8467     sort={\the\glsshorttok},
8468     description={\the\glslongtok},%
8469     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}%
8470     \protect\glsxtrfullsep{\the\glslabeltok}}%
8471     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8472     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}%
8473     \protect\glsxtrfullsep{\the\glslabeltok}}%
8474     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8475     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8476 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8477   \glshasattribute{\the\glslabeltok}{regular}%
8478   {%
8479     \glissetattribute{\the\glslabeltok}{regular}{false}%
8480   }%
8481   {}}%
8482 }%
8483 }%
8484 {%
8485   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8486   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8487   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
8488   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8489   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8490 \renewcommand*{\glxtrfullformat}[2]{%
8491   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8492   \ifglxtrininsertinside\else##2\fi
8493   \glxtrfullsep{##1}%
8494   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8495 }%
8496 \renewcommand*{\glxtrfullplformat}[2]{%
8497   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8498   \ifglxtrininsertinside\else##2\fi
8499   \glxtrfullsep{##1}%
8500   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8501 }%
8502 \renewcommand*{\Glsxtrfullformat}[2]{%
8503   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8504   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8505   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8506 }%
8507 \renewcommand*{\Glsxtrfullplformat}[2]{%
8508   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8509   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8510   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8511 }%
8512 }

```

rt-sm-long-desc As before but user provides description

```

8513 \newabbreviationstyle{short-sm-long-desc}%
8514 {%
8515   \renewcommand*{\CustomAbbreviationFields}{%
8516     name={\glxtrshortlongdescname},
8517     sort={\glxtrshortlongdescsort},
8518     first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}%
8519       \protect\glxtrfullsep{\the\glslabeltok}%
8520       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8521     firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}%
8522       \protect\glxtrfullsep{\the\glslabeltok}%
8523       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8524     text={\protect\glssabbrvsmfont{\the\glssshorttok}},%
8525     plural={\protect\glssabbrvsmfont{\the\glssshortpltok}}%
8526   }%

```

Unset the regular attribute if it has been set.

```

8527 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8528   \glshasattribute{\the\glslabeltok}{regular}%
8529   {%
8530     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8531   }%
8532   {}%
8533 }%
8534 }%
8535 {%

```


As short-sm-long style:

```
8536 \GlsXtrUseAbbrStyleFmts{short-sm-long}%  
8537 }
```

short-sm

```
8538 \newabbreviationstyle{short-sm}%  
8539 {%  
8540   \renewcommand*{\CustomAbbreviationFields}{%  
8541     name={\glstrshortnolongname},  
8542     sort={\the\glsshorttok},  
8543     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},  
8544     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},  
8545     text={\protect\glsabbrvsmfont{\the\glsshorttok}},  
8546     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},  
8547     description={\the\glslongtok}}%  
8548   \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
8549     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%  
8550 }%  
8551 {%  
8552   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%  
8553   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%  
8554   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
8555   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%  
8556   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
8557 \renewcommand*{\glxtrinlinefullformat}[2]{%  
8558   \protect\glsfirstabbrvsmfont{\glsaccesssshort{##1}}%  
8559   \ifglxtrininsertinside##2\fi}%  
8560   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%  
8561   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%  
8562 }%  
8563 \renewcommand*{\glxtrinlinefullplformat}[2]{%  
8564   \protect\glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}%  
8565   \ifglxtrininsertinside##2\fi}%  
8566   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%  
8567   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%  
8568 }%  
  
8569 \renewcommand*{\Glsxtrinlinefullformat}[2]{%  
8570   \protect\glsfirstabbrvsmfont{\Glsaccesssshort{##1}}%  
8571   \ifglxtrininsertinside##2\fi}%  
8572   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%  
8573   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%  
8574 }%  
8575 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%  
8576   \protect\glsfirstabbrvsmfont{\Glsaccesssshortpl{##1}}%  
8577   \ifglxtrininsertinside##2\fi}%  
8578   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%  
8579   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%  
8579 }
```

8580 }%

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.

```
8581 \renewcommand*{\glxtrfullformat}[2]{%
8582   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8583   \ifglxtrininsertinside\else##2\fi
8584 }%
8585 \renewcommand*{\glxtrfullplformat}[2]{%
8586   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8587   \ifglxtrininsertinside\else##2\fi
8588 }%
8589 \renewcommand*{\Glsxtrfullformat}[2]{%
8590   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8591   \ifglxtrininsertinside\else##2\fi
8592 }%
8593 \renewcommand*{\Glsxtrfullplformat}[2]{%
8594   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8595   \ifglxtrininsertinside\else##2\fi
8596 }%
8597 }
```

short-sm-nolong

```
8598 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8599 \newabbreviationstyle{short-sm-desc}%
8600 {%
8601   \renewcommand*{\CustomAbbreviationFields}{%
8602     name={\glxtrshortdescname},
8603     sort={\the\glssshorttok},
8604     first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}},
8605     firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}},
8606     text={\protect\glsabbrvsmfont{\the\glssshorttok}},
8607     plural={\protect\glsabbrvsmfont{\the\glssshortpltok}},
8608     description={\the\glslongtok}}%
8609   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8610     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8611 }%
8612 {%
8613   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8614   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8615   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8616   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8617   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8618 \renewcommand*{\glxtrininlinefullformat}[2]{%
8619   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8620   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
```

```

8621 \glstrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8622 }%
8623 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8624 \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8625 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8626 \glstrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8627 }%
8628 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8629 \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8630 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8631 \glstrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8632 }%
8633 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8634 \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8635 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8636 \glstrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8637 }%

```

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.

```

8638 \renewcommand*{\glxtrfullformat}[2]{%
8639 \glfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8640 \ifglxtrininsertinside\else##2\fi
8641 }%
8642 \renewcommand*{\glxtrfullplformat}[2]{%
8643 \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8644 \ifglxtrininsertinside\else##2\fi
8645 }%
8646 \renewcommand*{\Glsxtrfullformat}[2]{%
8647 \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8648 \ifglxtrininsertinside\else##2\fi
8649 }%
8650 \renewcommand*{\Glsxtrfullplformat}[2]{%
8651 \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8652 \ifglxtrininsertinside\else##2\fi
8653 }%
8654 }

```

-sm-nolong-desc

```
8655 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

nolong-short-sm

```

8656 \newabbreviationstyle{nolong-short-sm}%
8657 {%
8658 \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8659 }%
8660 {%
8661 \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8662 \renewcommand*{\glxtrinlinefullformat}[2]{%
8663   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8664     \ifglxtrinsertinside##2\fi}%
8665   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8666   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8667 }%
8668 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8669   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8670     \ifglxtrinsertinside##2\fi}%
8671   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8672   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8673 }%
8674 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8675   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8676     \ifglxtrinsertinside##2\fi}%
8677   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8678   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8679 }%
8680 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8681   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8682     \ifglxtrinsertinside##2\fi}%
8683   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8684   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8685 }%
8686 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8687 \newabbreviationstyle{long-noshort-sm}%
8688 {%
8689   \renewcommand*{\CustomAbbreviationFields}{%
8690     name={\glxtrlongnoshortname},
8691     sort={\the\glsshorttok},
8692     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8693     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8694     text={\protect\glslongdefaultfont{\the\glslongtok}},
8695     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8696     description={\the\glslongtok}%
8697   }%
8698   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8699     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8700 }%
8701 {%
8702   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8703   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8704   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8705   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8706   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8707 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8708   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8709   \ifglxtrinsertinside \else##2\fi
8710 }%
8711 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8712   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8713   \ifglxtrinsertinside \else##2\fi
8714 }%
8715 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8716   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8717   \ifglxtrinsertinside \else##2\fi
8718 }%
8719 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8720   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8721   \ifglxtrinsertinside \else##2\fi
8722 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8723 \renewcommand*{\glxtrinlinefullformat}[2]{%
8724   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8725   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8726   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8727 }%
8728 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8729   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8730   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8731   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8732 }%
8733 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8734   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8735   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8736   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8737 }%
8738 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8739   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8740   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8741   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8742 }%

```

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.

```

8743 \renewcommand*{\glxtrfullformat}[2]{%
8744   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8745   \ifglxtrinsertinside\else##2\fi
8746 }%
8747 \renewcommand*{\glxtrfullplformat}[2]{%
8748   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8749   \ifglxtrinsertinside\else##2\fi
8750 }%
8751 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

8752 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8753 \ifglxtrinsertinside\else##2\fi
8754 }%
8755 \renewcommand*{\Glsxtrfullplformat}[2]{%
8756 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8757 \ifglxtrinsertinside\else##2\fi
8758 }%
8759 }

```

long-sm Backward compatibility:

```

8760 \@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.

```

8761 \newabbreviationstyle{long-noshort-sm-desc}%
8762 {%
8763 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8764 }%
8765 {%
8766 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8767 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8768 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
8769 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8770 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8771 \renewcommand*\glsxtrsubsequentfnt}[2]{%
8772 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8773 \ifglxtrinsertinside \else##2\fi
8774 }%
8775 \renewcommand*\glsxtrsubsequentplfnt}[2]{%
8776 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8777 \ifglxtrinsertinside \else##2\fi
8778 }%
8779 \renewcommand*\Glsxtrsubsequentfnt}[2]{%
8780 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8781 \ifglxtrinsertinside \else##2\fi
8782 }%
8783 \renewcommand*\Glsxtrsubsequentplfnt}[2]{%
8784 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8785 \ifglxtrinsertinside \else##2\fi
8786 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8787 \renewcommand*\glsxtrinlinefullformat}[2]{%
8788 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8789 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8790 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8791 }%
8792 \renewcommand*\glsxtrinlinefullplformat}[2]{%

```

```

8793 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8794 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8795 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8796 }%
8797 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8798 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8799 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8800 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8801 }%
8802 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8803 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8804 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8805 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8806 }%

```

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.

```

8807 \renewcommand*{\glxtrfullformat}[2]{%
8808 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8809 \ifglxtrinsertinside\else##2\fi
8810 }%
8811 \renewcommand*{\glxtrfullplformat}[2]{%
8812 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8813 \ifglxtrinsertinside\else##2\fi
8814 }%
8815 \renewcommand*{\Glsxtrfullformat}[2]{%
8816 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8817 \ifglxtrinsertinside\else##2\fi
8818 }%
8819 \renewcommand*{\Glsxtrfullplformat}[2]{%
8820 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8821 \ifglxtrinsertinside\else##2\fi
8822 }%
8823 }

```

long-desc-sm Backward compatibility:

```

8824 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8825 \newabbreviationstyle{short-sm-footnote}%
8826 {%
8827 \renewcommand*{\CustomAbbreviationFields}{%
8828 name={\glxtrfootnotename},
8829 sort={\the\glsshorttok},
8830 description={\the\glslongtok},%
8831 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}%
8832 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8833 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8834 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}%

```

```

8835 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8836 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8837 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.

```

8838 \renewcommand*\GlsXtrPostNewAbbreviation{%
8839 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8840 \glsattribute{\the\glslabeltok}{regular}%
8841 {%
8842 \glssetattribute{\the\glslabeltok}{regular}{false}%
8843 }%
8844 {}%
8845 }%
8846 }%
8847 {%
8848 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8849 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8850 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8851 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8852 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8853 \renewcommand*\glxtrfullformat[2]{%
8854 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8855 \ifglxtrininsertinside\else##2\fi
8856 \protect\glxtrabbrvfootnote{##1}%
8857 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8858 }%
8859 \renewcommand*\glxtrfullplformat[2]{%
8860 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8861 \ifglxtrininsertinside\else##2\fi
8862 \protect\glxtrabbrvfootnote{##1}%
8863 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8864 }%
8865 \renewcommand*\Glsxtrfullformat[2]{%
8866 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8867 \ifglxtrininsertinside\else##2\fi
8868 \protect\glxtrabbrvfootnote{##1}%
8869 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8870 }%
8871 \renewcommand*\Glsxtrfullplformat[2]{%
8872 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8873 \ifglxtrininsertinside\else##2\fi
8874 \protect\glxtrabbrvfootnote{##1}%
8875 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8876 }%

```

The first use full form and the inline full form use the short (long) style.

```

8877 \renewcommand*\glxtrininlinefullformat[2]{%
8878 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```



```

8879 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8880 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
8881 }%
8882 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8883 \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8884 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8885 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8886 }%
8887 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8888 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8889 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8890 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
8891 }%
8892 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8893 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8894 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8895 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8896 }%
8897 }

```

footnote-sm Backward compatibility:

```

8898 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

8899 \newabbreviationstyle{short-sm-postfootnote}%
8900 {%
8901 \renewcommand*{\CustomAbbreviationFields}{%
8902 name={\glxtrfootnotename},
8903 sort={\the\glssshorttok},
8904 description={\the\glslongtok},%
8905 first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}},%
8906 firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}},%
8907 plural={\protect\glssabbrvsmfont{\the\glssshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

8908 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8909 \csdef{glxtrpostlink\glscategorylabel}{%
8910 \glxtrifwasfirstuse
8911 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8912 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8913 {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}%
8914 }%
8915 {}%
8916 }%
8917 \glshasattribute{\the\glslabeltok}{regular}%
8918 {%

```

```

8919     \glsetattribute{\the\glslabeltok}{regular}{false}%
8920 }%
8921 {}%
8922 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8923 \renewcommand*\glxtrsetupfulldefs{%
8924   \let\glxtrifwasfirstuse\@secondoftwo
8925 }%
8926 }%
8927 {%
8928   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8929   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8930   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8931   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8932   \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8933 \renewcommand*\glxtrfullformat[2]{%
8934   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8935   \ifglxtrininsertinside\else##2\fi
8936 }%
8937 \renewcommand*\glxtrfullplformat[2]{%
8938   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8939   \ifglxtrininsertinside\else##2\fi
8940 }%
8941 \renewcommand*\Glsxtrfullformat[2]{%
8942   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8943   \ifglxtrininsertinside\else##2\fi
8944 }%
8945 \renewcommand*\Glsxtrfullplformat[2]{%
8946   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8947   \ifglxtrininsertinside\else##2\fi
8948 }%

```

The first use full form and the inline full form use the short (long) style.

```

8949 \renewcommand*\glxtrininlinefullformat[2]{%
8950   \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8951   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8952   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
8953 }%
8954 \renewcommand*\glxtrininlinefullplformat[2]{%
8955   \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8956   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8957   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8958 }%
8959 \renewcommand*\Glsxtrininlinefullformat[2]{%
8960   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8961   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8962   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%

```

```

8963 }%
8964 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8965   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8966   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8967   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8968 }%
8969 }

```

postfootnote-sm Backward compatibility:

```

8970 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

8971 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

```

`firstabbrvemfont`

```

8972 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

8973 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`firstlongemfont` Only used by the “long-em” styles.

```

8974 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%

```

`\glslongemfont` Only used by the “long-em” styles.

```

8975 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

```

`long-short-em` The long form is just set in the default long font.

```

8976 \newabbreviationstyle{long-short-em}%
8977 {%
8978   \renewcommand*{\CustomAbbreviationFields}{%
8979     name={\glxtrlongshortname},
8980     sort={\the\glsshorttok},
8981     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8982       \protect\glxtrfullsep{\the\glslabeltok}%
8983       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8984     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8985       \protect\glxtrfullsep{\the\glslabeltok}%
8986       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8987     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
8988     description={\the\glslongtok}}%
8989   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8990     \glshasattribute{\the\glslabeltok}{regular}%

```

```

8991   {%
8992     \glssetattribute{\the\glslabeltok}{regular}{false}%
8993   }%
8994   {}%
8995 }%
8996 }%
8997 {%
8998   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
8999   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9000   \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%

```

Use the default long fonts.

```

9001   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9002   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9003   \renewcommand*\glsxtrfullformat[2]{%
9004     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9005     \ifglsxtrininsertinside\else##2\fi
9006     \glsxtrfullsep{##1}%
9007     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9008   }%
9009   \renewcommand*\glsxtrfullplformat[2]{%
9010     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9011     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9012     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9013   }%
9014   \renewcommand*\Glsxtrfullformat[2]{%
9015     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9016     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9017     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9018   }%
9019   \renewcommand*\Glsxtrfullplformat[2]{%
9020     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9021     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9022     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9023   }%
9024 }

```

g-short-em-desc

```

9025 \newabbreviationstyle{long-short-em-desc}%
9026 {%
9027   \renewcommand*\CustomAbbreviationFields{%
9028     name={\glsxtrlongshortdescname},
9029     sort={\glsxtrlongshortdescsort},%
9030     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9031       \protect\glsxtrfullsep{\the\glslabeltok}%
9032       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9033     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9034       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

9035     \glxstrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9036     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9037     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9038 }%

```

Unset the regular attribute if it has been set.

```

9039 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9040   \glshasattribute{\the\glslabeltok}{regular}%
9041   {%
9042     \glssetattribute{\the\glslabeltok}{regular}{false}%
9043   }%
9044   {}%
9045 }%
9046 }%
9047 {%

```

As long-short-em style:

```

9048 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9049 }

```

ong-em-short-em

```

9050 \newabbreviationstyle{long-em-short-em}%
9051 {%
    \glslongemfont is used in the description since \glsdesc doesn't set the style.
9052 \renewcommand*{\CustomAbbreviationFields}{%
9053   name={\glxtrlongshortname},
9054   sort={\the\glsshorttok},
9055   first={\protect\glsfirstlongemfont{\the\glslongtok}}%
9056   \protect\glxtrfullsep{\the\glslabeltok}%
9057   \glxstrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9058   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}}%
9059   \protect\glxtrfullsep{\the\glslabeltok}%
9060   \glxstrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9061   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9062   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9063 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9064   \glshasattribute{\the\glslabeltok}{regular}%
9065   {%
9066     \glssetattribute{\the\glslabeltok}{regular}{false}%
9067   }%
9068   {}%
9069 }%
9070 }%
9071 {%
9072 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9073 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9074 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%

```

```

9075 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9076 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9077 \renewcommand*{\glsxtrfullformat}[2]{%
9078   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9079   \ifglsxtrinsertinside\else##2\fi
9080   \glsxtrfullsep{##1}%
9081   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9082 }%
9083 \renewcommand*{\glsxtrfullplformat}[2]{%
9084   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9085   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9086   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9087 }%
9088 \renewcommand*{\Glsxtrfullformat}[2]{%
9089   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9090   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9091   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9092 }%
9093 \renewcommand*{\Glsxtrfullplformat}[2]{%
9094   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9095   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9096   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9097 }%
9098 }

```

m-short-em-desc

```

9099 \newabbreviationstyle{long-em-short-em-desc}%
9100 {%
9101   \renewcommand*{\CustomAbbreviationFields}{%
9102     name={\glsxtrlongshortdescname},
9103     sort={\glsxtrlongshortdescsort},%
9104     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9105       \protect\glsxtrfullsep{\the\glslabeltok}%
9106       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9107     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9108       \protect\glsxtrfullsep{\the\glslabeltok}%
9109       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9110     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9111     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9112   }%

```

Unset the regular attribute if it has been set.

```

9113 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9114   \glshasattribute{\the\glslabeltok}{regular}%
9115   {%
9116     \glissetattribute{\the\glslabeltok}{regular}{false}%
9117   }%
9118   {}%

```

```

9119 }%
9120 }%
9121 {%
9122   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9123 }

```

short-em-long Now the short (long) version

```

9124 \newabbreviationstyle{short-em-long}%
9125 {%
9126   \renewcommand*{\CustomAbbreviationFields}{%
9127     name={\glxtrshortlongname},
9128     sort={\the\glsshorttok},
9129     description={\the\gslongtok},%
9130     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9131       \protect\glxtrfullsep{\the\glslabeltok}%
9132       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongtok}}},%
9133     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9134       \protect\glxtrfullsep{\the\glslabeltok}%
9135       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongpltok}}},%
9136     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9137 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9138   \glshasattribute{\the\glslabeltok}{regular}%
9139   {%
9140     \glissetattribute{\the\glslabeltok}{regular}{false}%
9141   }%
9142   {}}%
9143 }%
9144 }%
9145 {%

```

Mostly as short-long style:

```

9146 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9147 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9148 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9149 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9150 \renewcommand*{\gslongfont}[1]{\gslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9151 \renewcommand*{\glxtrfullformat}[2]{%
9152   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9153   \ifglxtrininsertinside\else##2\fi
9154   \glxtrfullsep{##1}%
9155   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9156 }%
9157 \renewcommand*{\glxtrfullplformat}[2]{%
9158   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9159   \ifglxtrininsertinside\else##2\fi
9160   \glxtrfullsep{##1}%

```

```

9161 \glstrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9162 }%
9163 \renewcommand*{\Glsxtrfullformat}[2]{%
9164 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9165 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9166 \glstrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9167 }%
9168 \renewcommand*{\Glsxtrfullplformat}[2]{%
9169 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9170 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9171 \glstrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9172 }%
9173 }

```

rt-em-long-desc As before but user provides description

```

9174 \newabbreviationstyle{short-em-long-desc}%
9175 {%
9176 \renewcommand*{\CustomAbbreviationFields}{%
9177 name={\glstrshortlongdescname},
9178 sort={\glstrshortlongdescsort},
9179 first={\protect\glfirstabbrvemfont{\the\glshorttok}%
9180 \protect\glxtrfullsep{\the\gllabeltok}%
9181 \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongtok}}},%
9182 firstplural={\protect\glfirstabbrvemfont{\the\glshortpltok}%
9183 \protect\glxtrfullsep{\the\gllabeltok}%
9184 \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongpltok}}},%
9185 text={\protect\glabbrvemfont{\the\glshorttok}},%
9186 plural={\protect\glabbrvemfont{\the\glshortpltok}}}%
9187 }%

```

Unset the regular attribute if it has been set.

```

9188 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9189 \glshasattribute{\the\gllabeltok}{regular}%
9190 {%
9191 \glissetattribute{\the\gllabeltok}{regular}{false}%
9192 }%
9193 {}%
9194 }%
9195 }%
9196 {%
9197 \GlsXtrUseAbbrStyleFmts{short-em-long}%
9198 }

```

hort-em-long-em

```

9199 \newabbreviationstyle{short-em-long-em}%
9200 {%

```

\glslongemfont is used in the description since \glsdsc doesn't set the style.

```

9201 \renewcommand*{\CustomAbbreviationFields}{%
9202 name={\glstrshortlongname},

```



```

9203     sort={\the\glsshorttok},
9204     description={\protect\glslongemfont{\the\glslongtok}},%
9205     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9206       \protect\glsxtrfullsep{\the\glslabeltok}%
9207       \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9208     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9209       \protect\glsxtrfullsep{\the\glslabeltok}%
9210       \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9211     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9212 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9213   \glshasattribute{\the\glslabeltok}{regular}%
9214   {%
9215     \glissetattribute{\the\glslabeltok}{regular}{false}%
9216   }%
9217   {}%
9218 }%
9219 }%
9220 {%
9221 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9222 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9223 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9224 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9225 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9226 \renewcommand*{\glsxtrfullformat}[2]{%
9227   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9228   \ifglsxtrininsertinside\else##2\fi
9229   \glsxtrfullsep{##1}%
9230   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9231 }%
9232 \renewcommand*{\glsxtrfullplformat}[2]{%
9233   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9234   \ifglsxtrininsertinside\else##2\fi
9235   \glsxtrfullsep{##1}%
9236   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9237 }%
9238 \renewcommand*{\Glsxtrfullformat}[2]{%
9239   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9240   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9241   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9242 }%
9243 \renewcommand*{\Glsxtrfullplformat}[2]{%
9244   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9245   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9246   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9247 }%
9248 }

```

em-long-em-desc

```
9249 \newabbreviationstyle{short-em-long-em-desc}%
9250 {%
9251   \renewcommand*{\CustomAbbreviationFields}{%
9252     name={\glstrshortlongdescname},%
9253     sort={\glstrshortlongdescsort},%
9254     first={\protect\glfirstabbrvemfont{\the\glsshorttok}%
9255       \protect\glstrfullsep{\the\glslabeltok}%
9256       \glstrparen{\protect\glfirstlongemfont{\the\glslongtok}}},%
9257     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}%
9258       \protect\glstrfullsep{\the\glslabeltok}%
9259       \glstrparen{\protect\glfirstlongemfont{\the\glslongpltok}}},%
9260     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9261     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9262   }%
```

Unset the regular attribute if it has been set.

```
9263   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9264     \glshasattribute{\the\glslabeltok}{regular}%
9265     {%
9266       \glissetattribute{\the\glslabeltok}{regular}{false}%
9267     }%
9268   }%
9269 }%
9270 }%
9271 {%
9272   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9273 }
```

short-em

```
9274 \newabbreviationstyle{short-em}%
9275 {%
9276   \renewcommand*{\CustomAbbreviationFields}{%
9277     name={\glstrshortnolongname},
9278     sort={\the\glsshorttok},
9279     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9280     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9281     text={\protect\glsabbrvemfont{\the\glsshorttok}},
9282     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9283     description={\the\glslongtok}}%
9284   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9285     \glissetattribute{\the\glslabeltok}{regular}{true}}%
9286 }%
9287 {%
9288   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9289   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9290   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9291   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9292   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```

9293 \renewcommand*{\glxtrinlinefullformat}[2]{%
9294   \protect\glsfirstabbrvemfont{\glsaccesssshort{##1}%
9295     \ifglxtrinsertinside##2\fi}%
9296   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9297   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9298 }%
9299 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9300   \protect\glsfirstabbrvemfont{\glsaccesssshortpl{##1}%
9301     \ifglxtrinsertinside##2\fi}%
9302   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9303   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9304 }%

9305 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9306   \protect\glsfirstabbrvemfont{\Glsaccesssshort{##1}%
9307     \ifglxtrinsertinside##2\fi}%
9308   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9309   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9310 }%
9311 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9312   \protect\glsfirstabbrvemfont{\Glsaccesssshortpl{##1}%
9313     \ifglxtrinsertinside##2\fi}%
9314   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9315   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9316 }%

```

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.

```

9317 \renewcommand*{\glxtrfullformat}[2]{%
9318   \glsfirstabbrvemfont{\glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
9319   \ifglxtrinsertinside\else##2\fi
9320 }%
9321 \renewcommand*{\glxtrfullplformat}[2]{%
9322   \glsfirstabbrvemfont{\glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
9323   \ifglxtrinsertinside\else##2\fi
9324 }%
9325 \renewcommand*{\Glsxtrfullformat}[2]{%
9326   \glsfirstabbrvemfont{\Glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
9327   \ifglxtrinsertinside\else##2\fi
9328 }%
9329 \renewcommand*{\Glsxtrfullplformat}[2]{%
9330   \glsfirstabbrvemfont{\Glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
9331   \ifglxtrinsertinside\else##2\fi
9332 }%
9333 }

```

short-em-nolong

```

9334 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```
9335 \newabbreviationstyle{short-em-desc}%
9336 {%
9337   \renewcommand*{\CustomAbbreviationFields}{%
9338     name={\glxtrshortdescname},
9339     sort={\the\glsshorttok},
9340     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9341     firstplural={\protect\glfirstabbrvemfont{\the\glshortpltok}},
9342     text={\protect\glabbrvemfont{\the\glsshorttok}},
9343     plural={\protect\glabbrvemfont{\the\glshortpltok}},
9344     description={\the\glslongtok}}%
9345   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9346     \glsetattribute{\the\glslabeltok}{regular}{true}}%
9347 }%
9348 {%
9349   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9350   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9351   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9352   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9353   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
9354   \renewcommand*{\glxtrinlinefullformat}[2]{%
9355     \glfirstabbrvemfont{\glaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9356     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9357     \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9358   }%
9359   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9360     \glfirstabbrvemfont{\glaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9361     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9362     \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9363   }%
9364   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9365     \glfirstabbrvemfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9366     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9367     \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9368   }%
9369   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9370     \glfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9371     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9372     \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9373   }%
```

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.

```
9374   \renewcommand*{\glxtrfullformat}[2]{%
9375     \glfirstabbrvemfont{\glaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9376     \ifglxtrininsertinside\else##2\fi
9377   }%
9378   \renewcommand*{\glxtrfullplformat}[2]{%
9379     \glfirstabbrvemfont{\glaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9380     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9381     \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9382   }%
```

```

9379 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9380 \ifglxtrinsertinside\else##2\fi
9381 }%
9382 \renewcommand*{\Glsxtrfullformat}[2]{%
9383 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9384 \ifglxtrinsertinside\else##2\fi
9385 }%
9386 \renewcommand*{\Glsxtrfullplformat}[2]{%
9387 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9388 \ifglxtrinsertinside\else##2\fi
9389 }%
9390 }

```

-em-nolong-desc

```

9391 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9392 \newabbreviationstyle{nolong-short-em}%
9393 {%
9394 \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9395 }%
9396 {%
9397 \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9398 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9399 \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9400 \ifglxtrinsertinside##2\fi}%
9401 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9402 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9403 }%
9404 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9405 \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9406 \ifglxtrinsertinside##2\fi}%
9407 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9408 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9409 }%
9410 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9411 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9412 \ifglxtrinsertinside##2\fi}%
9413 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9414 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9415 }%
9416 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9417 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9418 \ifglxtrinsertinside##2\fi}%
9419 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9420 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9421 }%
9422 }

```

long-noshort-em The short form is explicitly invoked through commands like \glssshort.

```
9423 \newabbreviationstyle{long-noshort-em}%
9424 {%
9425   \renewcommand*{\CustomAbbreviationFields}{%
9426     name={\glxtrlongnoshortname},
9427     sort={\the\glssshorttok},
9428     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9429     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9430     text={\protect\glslongdefaultfont{\the\glslongtok}},
9431     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9432     description={\the\glslongtok}%
9433   }%
9434   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9435     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9436 }%
9437 {%
9438   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9439   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9440   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9441   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9442   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9443 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9444   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9445   \ifglxtrininsertinside \else##2\fi
9446 }%
9447 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9448   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9449   \ifglxtrininsertinside \else##2\fi
9450 }%
9451 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9452   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9453   \ifglxtrininsertinside \else##2\fi
9454 }%
9455 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9456   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9457   \ifglxtrininsertinside \else##2\fi
9458 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9459 \renewcommand*{\glxtrinlinefullformat}[2]{%
9460   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9461   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9462   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9463 }%
9464 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9465   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9466   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9467   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%

```

```

9468 }%
9469 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9470   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9471   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9472   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9473 }%
9474 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9475   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9476   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9477   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9478 }%

```

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.

```

9479 \renewcommand*{\glxtrfullformat}[2]{%
9480   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9481   \ifglxtrinsertinside\else##2\fi
9482 }%
9483 \renewcommand*{\glxtrfullplformat}[2]{%
9484   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9485   \ifglxtrinsertinside\else##2\fi
9486 }%
9487 \renewcommand*{\Glsxtrfullformat}[2]{%
9488   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9489   \ifglxtrinsertinside\else##2\fi
9490 }%
9491 \renewcommand*{\Glsxtrfullplformat}[2]{%
9492   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9493   \ifglxtrinsertinside\else##2\fi
9494 }%
9495 }

```

long-em Backward compatibility:

```

9496 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9497 \newabbreviationstyle{long-em-noshort-em}%
9498 {%
9499   \renewcommand*{\CustomAbbreviationFields}{%
9500     name={\glxtrlongnoshortname},
9501     sort={\the\glsshorttok},
9502     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9503     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9504     text={\protect\glslongemfont{\the\glslongtok}},
9505     plural={\protect\glslongemfont{\the\glslongpltok}},%
9506     description={\protect\glslongemfont{\the\glslongtok}}%
9507   }%
9508   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9509     \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

```

9510 }%
9511 {%
9512   \renewcommand*{\abbrvpluralsuffix}{\protect\glstremssuffix}%
9513   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9514   \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9515   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongemfont{##1}}%
9516   \renewcommand*{\glsslongfont}[1]{\glsslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9517   \renewcommand*{\glssxtrsubsequentfmt}[2]{%
9518     \glsslongemfont{\glssaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
9519     \ifglssxtrininsertinside \else##2\fi
9520   }%
9521   \renewcommand*{\glssxtrsubsequentplfmt}[2]{%
9522     \glsslongemfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
9523     \ifglssxtrininsertinside \else##2\fi
9524   }%
9525   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9526     \glsslongemfont{\Glsaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
9527     \ifglssxtrininsertinside \else##2\fi
9528   }%
9529   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9530     \glsslongemfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
9531     \ifglssxtrininsertinside \else##2\fi
9532   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9533   \renewcommand*{\glssxtrininlinefullformat}[2]{%
9534     \glssfirstlongemfont{\glssaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
9535     \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9536     \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9537   }%
9538   \renewcommand*{\glssxtrininlinefullplformat}[2]{%
9539     \glssfirstlongemfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
9540     \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9541     \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9542   }%
9543   \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9544     \glssfirstlongemfont{\Glsaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
9545     \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9546     \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9547   }%
9548   \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9549     \glssfirstlongemfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
9550     \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9551     \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9552   }%

```

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.


```

9553 \renewcommand*{\glstrfullformat}[2]{%
9554   \glstrfirstlongemfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
9555   \ifglstrinsertinside\else##2\fi
9556 }%
9557 \renewcommand*{\glstrfullplformat}[2]{%
9558   \glstrfirstlongemfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9559   \ifglstrinsertinside\else##2\fi
9560 }%
9561 \renewcommand*{\GlsXtrfullformat}[2]{%
9562   \glstrfirstlongemfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
9563   \ifglstrinsertinside\else##2\fi
9564 }%
9565 \renewcommand*{\GlsXtrfullplformat}[2]{%
9566   \glstrfirstlongemfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9567   \ifglstrinsertinside\else##2\fi
9568 }%
9569 }

```

noshort-em-noreg Like long-em-noshort-em but doesn't set the regular attribute.

```

9570 \newabbreviationstyle{long-em-noshort-em-noreg}%
9571 {%
9572   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
9573   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9574     \glshasattribute{\the\glslabeltok}{regular}%
9575     {%
9576       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9577     }%
9578   }%
9579 }%
9580 }%
9581 {%
9582   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
9583 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9584 \newabbreviationstyle{long-noshort-em-desc}%
9585 {%
9586   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9587 }%
9588 {%
9589   \renewcommand*{\abbrvpluralsuffix}{\protect\glstremssuffix}%
9590   \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9591   \renewcommand*{\glstrfirstabrvfont}[1]{\glstrfirstabrvemfont{##1}}%
9592   \renewcommand*{\glstrfirstlongfont}[1]{\glstrfirstlongdefaultfont{##1}}%
9593   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9594 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9595   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9596   \ifglxtrinsertinside \else##2\fi
9597 }%
9598 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9599   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9600   \ifglxtrinsertinside \else##2\fi
9601 }%
9602 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9603   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9604   \ifglxtrinsertinside \else##2\fi
9605 }%
9606 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9607   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9608   \ifglxtrinsertinside \else##2\fi
9609 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9610 \renewcommand*{\glxtrinlinefullformat}[2]{%
9611   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9612   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9613   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9614 }%
9615 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9616   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9617   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9618   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9619 }%
9620 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9621   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9622   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9623   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9624 }%
9625 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9626   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9627   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9628   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9629 }%

```

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.

```

9630 \renewcommand*{\glxtrfullformat}[2]{%
9631   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9632   \ifglxtrinsertinside\else##2\fi
9633 }%
9634 \renewcommand*{\glxtrfullplformat}[2]{%
9635   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9636   \ifglxtrinsertinside\else##2\fi
9637 }%
9638 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9639 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9640 \ifglxtrinsertinside\else##2\fi
9641 }%
9642 \renewcommand*{\Glsxtrfullplformat}[2]{%
9643 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9644 \ifglxtrinsertinside\else##2\fi
9645 }%
9646 }

```

long-desc-em Backward compatibility:

```

9647 \@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.

```

9648 \newabbreviationstyle{long-em-noshort-em-desc}%
9649 {%
9650 \renewcommand*{\CustomAbbreviationFields}{%
9651 name={\glxtrlongnoshortdescname},
9652 sort={\the\glslongtok},
9653 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9654 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9655 text={\glslongemfont{\the\glslongtok}},
9656 plural={\glslongemfont{\the\glslongpltok}}}%
9657 }%
9658 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9659 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9660 }%
9661 {%
9662 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9663 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9664 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9665 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9666 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9667 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9668 \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9669 \ifglxtrinsertinside \else##2\fi
9670 }%
9671 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9672 \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9673 \ifglxtrinsertinside \else##2\fi
9674 }%
9675 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9676 \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9677 \ifglxtrinsertinside \else##2\fi
9678 }%
9679 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9680 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%

```

```

9681 \ifglxtrinsertinside \else##2\fi
9682 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9683 \renewcommand*{\glxtrinlinefullformat}[2]{%
9684   \glsfirslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9685   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9686   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9687 }%
9688 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9689   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9690   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9691   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9692 }%
9693 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9694   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9695   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9696   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9697 }%
9698 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9699   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9700   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9701   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9702 }%

```

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.

```

9703 \renewcommand*{\glxtrfullformat}[2]{%
9704   \glsfirslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9705   \ifglxtrinsertinside\else##2\fi
9706 }%
9707 \renewcommand*{\glxtrfullplformat}[2]{%
9708   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9709   \ifglxtrinsertinside\else##2\fi
9710 }%
9711 \renewcommand*{\Glsxtrfullformat}[2]{%
9712   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9713   \ifglxtrinsertinside\else##2\fi
9714 }%
9715 \renewcommand*{\Glsxtrfullplformat}[2]{%
9716   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9717   \ifglxtrinsertinside\else##2\fi
9718 }%
9719 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

9720 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9721 {%
9722   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

9723 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9724   \glshasattribute{\the\glslabeltok}{regular}%
9725   {%
9726     \glsselattribute{\the\glslabeltok}{regular}{false}%
9727   }%
9728   {}%
9729 }%
9730}%
9731{%
9732 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9733}

```

ort-em-footnote

```

9734 \newabbreviationstyle{short-em-footnote}%
9735 {%
9736 \renewcommand*{\CustomAbbreviationFields}{%
9737   name={\glsxtrfootnotename},
9738   sort={\the\glsshorttok},
9739   description={\the\glslongtok},%
9740   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9741     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
9742     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9743   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9744     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
9745     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9746   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.

```

9747 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9748   \glsselattribute{\the\glslabeltok}{nohyperfirst}{true}%
9749   \glshasattribute{\the\glslabeltok}{regular}%
9750   {%
9751     \glsselattribute{\the\glslabeltok}{regular}{false}%
9752   }%
9753   {}%
9754 }%
9755}%
9756{%
9757 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9758 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9759 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9760 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9761 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9762 \renewcommand*{\glsxtrfullformat}[2]{%
9763   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsetinside##2\fi}%
9764   \ifglsxtrinsetinside\else##2\fi

```

```

9765 \protect\glxtrabbrvfootnote{##1}%
9766 {\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9767 }%
9768 \renewcommand*{\glxtrfullplformat}[2]{%
9769 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9770 \ifglxtrininsertinside\else##2\fi
9771 \protect\glxtrabbrvfootnote{##1}%
9772 {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9773 }%
9774 \renewcommand*{\Glsxtrfullformat}[2]{%
9775 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9776 \ifglxtrininsertinside\else##2\fi
9777 \protect\glxtrabbrvfootnote{##1}%
9778 {\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9779 }%
9780 \renewcommand*{\Glsxtrfullplformat}[2]{%
9781 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9782 \ifglxtrininsertinside\else##2\fi
9783 \protect\glxtrabbrvfootnote{##1}%
9784 {\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9785 }%

```

The first use full form and the inline full form use the short (long) style.

```

9786 \renewcommand*{\glxtrininlinefullformat}[2]{%
9787 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9788 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9789 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9790 }%
9791 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9792 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9793 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9794 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9795 }%
9796 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9797 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9798 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9799 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9800 }%
9801 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9802 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9803 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9804 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9805 }%
9806 }

```

footnote-em Backward compatibility:

```

9807 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9808 \newabbreviationstyle{short-em-postfootnote}%
9809 {%
9810   \renewcommand*{\CustomAbbreviationFields}{%
9811     name={\glxtrfootnotename},
9812     sort={\the\glsshorttok},
9813     description={\the\glslongtok},%
9814     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
9815     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
9816     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.

```

9817   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9818     \csdef{glxtrpostlink\glscategorylabel}{%
9819       \glxtrifwasfirstuse
9820       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9821         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
9822         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
9823       }%
9824     {}%
9825   }%
9826   \glshasattribute{\the\glslabeltok}{regular}%
9827   {%
9828     \glssetattribute{\the\glslabeltok}{regular}{false}%
9829   }%
9830   {}%
9831 }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

9832 \renewcommand*{\glxtrsetupfulldefs}{%
9833   \let\glxtrifwasfirstuse\@secondoftwo
9834 }%
9835 }%
9836 {%
9837   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9838   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9839   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9840   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9841   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9842   \renewcommand*{\glxtrfullformat}[2]{%
9843     \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9844     \ifglxtrininsertinside\else##2\fi
9845   }%
9846   \renewcommand*{\glxtrfullplformat}[2]{%
9847     \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

9848 \ifglxtrinsertinside\else##2\fi
9849 }%
9850 \renewcommand*{\Glsxtrfullformat}[2]{%
9851 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9852 \ifglxtrinsertinside\else##2\fi
9853 }%
9854 \renewcommand*{\Glsxtrfullplformat}[2]{%
9855 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9856 \ifglxtrinsertinside\else##2\fi
9857 }%

```

The first use full form and the inline full form use the short (long) style.

```

9858 \renewcommand*{\glxtrinlinefullformat}[2]{%
9859 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9860 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9861 \glxtrparen{\glfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9862 }%
9863 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9864 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9865 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9866 \glxtrparen{\glfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9867 }%
9868 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9869 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9870 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9871 \glxtrparen{\glfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9872 }%
9873 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9874 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9875 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9876 \glxtrparen{\glfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9877 }%
9878 }

```

postfootnote-em Backward compatibility:

```

9879 \@glxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}

```

1.7.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glxtruserfield Default is the useri field.

```

9880 \newcommand*{\glxtruserfield}{useri}

```

glxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If \glscurrentfieldvalue has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```

9881 \ifdef\glscurrentfieldvalue

```



```

9882 {
9883   \newcommand*{\glxtruserparen}[2]{%
9884     \glxtrfullsep{#2}%
9885     \glxtrparen
9886     {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9887   }
9888 }
9889 {
9890   \newcommand*{\glxtruserparen}[2]{%
9891     \glxtrfullsep{#2}%
9892     \glxtrparen
9893     {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9894   }
9895 }

```

Font used for short form:

lsabbrvuserfont

```
9896 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
9897 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
9898 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
9899 \newcommand*{\glfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
9900 \newcommand*{\glxtrusersuffix}{\glxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
9901 \newcommand*{\gluserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```

9902 \newabbreviationstyle{long-short-user}%
9903 {%
9904   \renewcommand*{\CustomAbbreviationFields}{%
9905     name={\glxtrlongshortname},
9906     sort={\the\glsshorttok},

```

```

9907 first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9908 \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
9909 {\the\glslabeltok}},%
9910 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9911 \protect\glsxtruserparen
9912 {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9913 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9914 description={\protect\glsuserdescription{\the\glslongtok}%
9915 {\the\glslabeltok}}}%

```

Unset the regular attribute if it has been set.

```

9916 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9917 \glsattribute{\the\glslabeltok}{regular}%
9918 {%
9919 \glssetattribute{\the\glslabeltok}{regular}{false}%
9920 }%
9921 {}%
9922 }%
9923 }%
9924 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9925 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
9926 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9927 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9928 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9929 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9930 \renewcommand*{\glsxtrfullformat}[2]{%
9931 \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9932 \ifglsxtrininsertinside\else##2\fi
9933 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9934 }%
9935 \renewcommand*{\glsxtrfullplformat}[2]{%
9936 \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9937 \ifglsxtrininsertinside\else##2\fi
9938 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9939 }%
9940 \renewcommand*{\Glsxtrfullformat}[2]{%
9941 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9942 \ifglsxtrininsertinside\else##2\fi
9943 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9944 }%
9945 \renewcommand*{\Glsxtrfullplformat}[2]{%
9946 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9947 \ifglsxtrininsertinside\else##2\fi
9948 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9949 }%
9950 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```
9951 \newabbreviationstyle{long-postshort-user}%
9952 {%
9953   \renewcommand*{\CustomAbbreviationFields}{%
9954     name={\glxtrlongshortname},
9955     sort={\the\glsshorttok},
9956     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9957     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

9958     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9959     description={\protect\glsuserdescription{\the\glslongtok}%
9960       {\the\glslabeltok}}}%
9961 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9962   \csdef{glxtrpostlink\glscategorylabel}{%
9963     \glxtrifwasfirstuse
9964     {%
9965       \glxtruserparen
9966       {\glsfirstabbrvuserfont{\glstryshort{\glslabel}}}%
9967       {\glslabel}%
9968     }%
9969   }%
9970 }%
9971 \glshasattribute{\the\glslabeltok}{regular}%
9972 {%
9973   \glissetattribute{\the\glslabeltok}{regular}{false}%
9974 }%
9975 {}%
9976 }%
9977 }%
9978 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
9979 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
9980 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9981 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9982 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9983 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
9984 \renewcommand*{\glxtrfullformat}[2]{%
9985   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9986   \ifglxtrininsertinside\else##2\fi
9987 }%
9988 \renewcommand*{\glxtrfullplformat}[2]{%
9989   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9990   \ifglxtrininsertinside\else##2\fi
9991 }%
9992 \renewcommand*{\Glsxtrfullformat}[2]{%
9993   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9994   \ifglxtrininsertinside\else##2\fi
```

```

9995 }%
9996 \renewcommand*{\Glsxtrfullplformat}[2]{%
9997   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9998   \ifglxtrinsertinside\else##2\fi
9999 }%

```

In-line format:

```

10000 \renewcommand*{\glxtrinlinefullformat}[2]{%
10001   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10002   \ifglxtrinsertinside\else##2\fi
10003   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshort{##1}}}{##1}%
10004 }%
10005 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10006   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10007   \ifglxtrinsertinside\else##2\fi
10008   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshortpl{##1}}}{##1}%
10009 }%
10010 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10011   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10012   \ifglxtrinsertinside\else##2\fi
10013   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshort{##1}}}{##1}%
10014 }%
10015 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10016   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10017   \ifglxtrinsertinside\else##2\fi
10018   \glxtruserparen{\glsfirstabbrvuserfont{\Glsaccessshortpl{##1}}}{##1}%
10019 }%
10020 }

```

ortuserdescname

```

10021 \newcommand*{\glxtrlongshortuserdescname}{%
10022   \protect\glslonguserfont{\the\glslongtok}%
10023   \protect\glxtruserparen
10024   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10025 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10026 \newabbreviationstyle{long-postshort-user-desc}%
10027 {%
10028   \renewcommand*{\CustomAbbreviationFields}{%
10029     name={\glxtrlongshortuserdescname},
10030     sort={\the\glslongtok},
10031     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10032     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10033     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10034     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10035 }%
10036 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10037   \csdef{glxtrpostlink\glscategorylabel}{%

```

```

10038     \glxtrifwasfirstuse
10039     {%
10040     \glxtruserparen
10041         {\glsfirstabbrvuserfont{\gl Sentryshort{\glslabel}}}%
10042         {\glslabel}%
10043     }%
10044     {}%
10045 }%
10046 \glshasattribute{\the\glslabeltok}{regular}%
10047 {%
10048     \glissetattribute{\the\glslabeltok}{regular}{false}%
10049 }%
10050 {}%
10051 }%
10052 }%
10053 {%
10054     \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10055 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10056 \newabbreviationstyle{short-postlong-user}%
10057 {%
10058     \renewcommand*{\CustomAbbreviationFields}{%
10059         name={\glxtrshortlongname},
10060         sort={\the\glsshorttok},
10061         first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10062         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10063         plural={\protect\glsabrvuserfont{\the\glsshortpltok}},%
10064         description={\protect\gluserdescription{\the\glslongtok}%
10065             {\the\glslabeltok}}}%
10066     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10067         \csdef{glxtrpostlink\glscategorylabel}{%
10068             \glxtrifwasfirstuse
10069             {%
10070             \glxtruserparen
10071                 {\glsfirstlonguserfont{\gl Sentrylong{\glslabel}}}%
10072                 {\glslabel}%
10073             }%
10074             {}%
10075         }%
10076         \glshasattribute{\the\glslabeltok}{regular}%
10077         {%
10078             \glissetattribute{\the\glslabeltok}{regular}{false}%
10079         }%
10080         {}%
10081     }%
10082 }%
10083 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```
10084 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10085 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10086 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10087 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10088 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
10089 \renewcommand*{\glxtrfullformat}[2]{%
10090   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10091   \ifglxtrininsertinside\else##2\fi
10092 }%
10093 \renewcommand*{\glxtrfullplformat}[2]{%
10094   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10095   \ifglxtrininsertinside\else##2\fi
10096 }%
10097 \renewcommand*{\Glsxtrfullformat}[2]{%
10098   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10099   \ifglxtrininsertinside\else##2\fi
10100 }%
10101 \renewcommand*{\Glsxtrfullplformat}[2]{%
10102   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10103   \ifglxtrininsertinside\else##2\fi
10104 }%
```

In-line format:

```
10105 \renewcommand*{\glxtrinlinefullformat}[2]{%
10106   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10107   \ifglxtrininsertinside\else##2\fi
10108   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10109 }%
10110 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10111   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10112   \ifglxtrininsertinside\else##2\fi
10113   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10114 }%
10115 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10116   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10117   \ifglxtrininsertinside\else##2\fi
10118   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10119 }%
10120 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10121   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10122   \ifglxtrininsertinside\else##2\fi
10123   \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10124 }%
10125 }
```

onguserdesname

```
10126 \newcommand*{\glxtrshortlonguserdesname}{%
```

```

10127 \protect\glsabbrvuserfont{\the\glsshorttok}%
10128 \protect\glxtruserparen
10129 {\protect\glslonguserfont{\the\glslongpltok}}}%
10130 {\the\glslabeltok}%
10131 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10132 \newabbreviationstyle{short-postlong-user-desc}%
10133 {%
10134   \renewcommand*{\CustomAbbreviationFields}{%
10135     name={\glxtrshortlonguserdescname},
10136     sort={\the\glsshorttok},
10137     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10138     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10139     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10140     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10141   }%
10142   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10143     \csdef{glxtrpostlink\glscategorylabel}{%
10144       \glxtrifwasfirstuse
10145       {%
10146         \glxtruserparen
10147         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10148         {\glslabel}%
10149       }%
10150     }%
10151   }%
10152   \glshasattribute{\the\glslabeltok}{regular}%
10153   {%
10154     \glissetattribute{\the\glslabeltok}{regular}{false}%
10155   }%
10156   {}%
10157 }%
10158 }%
10159 {%
10160   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10161 }

```

short-user-desc

```

10162 \newabbreviationstyle{long-short-user-desc}%
10163 {%
10164   \renewcommand*{\CustomAbbreviationFields}{%
10165     name={\glxtrlongshortuserdescname},
10166     sort={\glxtrlongshortdescsort},%

10167     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10168       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10169     {\the\glslabeltok}},%
10170     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}}%

```

```

10171 \protect\glxtruserparen
10172 {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10173 text={\protect\glsabbrvfont{\the\glsshorttok}},%
10174 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10175 }%

```

Unset the regular attribute if it has been set.

```

10176 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10177 \glshasattribute{\the\glslabeltok}{regular}%
10178 {%
10179 \glissetattribute{\the\glslabeltok}{regular}{false}%
10180 }%
10181 {}%
10182 }%
10183 }%
10184 {%
10185 \GlsXtrUseAbbrStyleFmts{long-short-user}%
10186 }

```

short-long-user

```

10187 \newabbreviationstyle{short-long-user}%
10188 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10189 \renewcommand*{\CustomAbbreviationFields}{%
10190 name={\glxtrshortlongname},
10191 sort={\the\glsshorttok},
10192 description={\protect\glsuserdescription{\the\glslongtok}%
10193 {\the\glslabeltok}},%
10194 first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10195 \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
10196 {\the\glslabeltok}},%
10197 firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10198 \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
10199 {\the\glslabeltok}},%
10200 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10201 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10202 \glshasattribute{\the\glslabeltok}{regular}%
10203 {%
10204 \glissetattribute{\the\glslabeltok}{regular}{false}%
10205 }%
10206 {}%
10207 }%
10208 }%
10209 {%

```


In case the user wants to mix and match font styles, these are redefined here.

```
10210 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10211 \renewcommand*{\glssabrvfont[1]}{\glssabrvuserfont{##1}}%
10212 \renewcommand*{\glssfirstabrvfont}[1]{\glssfirstabrvuserfont{##1}}%
10213 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlonguserfont{##1}}%
10214 \renewcommand*{\glsslongfont}[1]{\glsslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
10215 \renewcommand*{\glxtrfullformat}[2]{%
10216   \glssfirstabrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10217   \ifglxtrininsertinside\else##2\fi
10218   \glxtruserparen{\glssfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10219 }%
10220 \renewcommand*{\glxtrfullplformat}[2]{%
10221   \glssfirstabrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10222   \ifglxtrininsertinside\else##2\fi
10223   \glxtruserparen{\glssfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10224 }%
10225 \renewcommand*{\Glsxtrfullformat}[2]{%
10226   \glssfirstabrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10227   \ifglxtrininsertinside\else##2\fi
10228   \glxtruserparen{\glssfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10229 }%
10230 \renewcommand*{\Glsxtrfullplformat}[2]{%
10231   \glssfirstabrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10232   \ifglxtrininsertinside\else##2\fi
10233   \glxtruserparen{\glssfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10234 }%
10235 }
```

-long-user-desc

```
10236 \newabbreviationstyle{short-long-user-desc}%
10237 {%
10238   \renewcommand*{\CustomAbbreviationFields}{%
10239     name={\glxtrshortlonguserdescname},
10240     sort={\glxtrshortlongdescsort},%
10241     first={\protect\glssfirstabrvuserfont{\the\glssshorttok}}%
10242     \protect\glxtruserparen{\protect\glssfirstlonguserfont{\the\glsslongtok}}%
10243     {\the\glsslabeltok}},%
10244     firstplural={\protect\glssfirstabrvuserfont{\the\glssshortpltok}}%
10245     \protect\glxtruserparen{\protect\glssfirstlonguserfont{\the\glsslongpltok}}%
10246     {\the\glsslabeltok}},%
10247     text={\protect\glssabrvfont{\the\glssshorttok}},%
10248     plural={\protect\glssabrvfont{\the\glssshortpltok}}%
10249   }%
```

Unset the regular attribute if it has been set.

```
10250 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10251   \glshasattribute{\the\glsslabeltok}{regular}%
```

```

10252   {%
10253       \glsetattribute{\the\glslabeltok}{regular}{false}%
10254   }%
10255   {}%
10256   }%
10257 }%
10258 {%
10259   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10260 }

```

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 `\glstrlongset` `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

`\glstrifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```

10261 \newrobustcmd*{\glstrifhyphenstart}[3]{%
10262   \ifx\glsinsert#1\relax
10263     \expandafter\@glstrifhyphenstart#1\relax\relax
10264     \@end@glstrifhyphenstart{#2}{#3}%
10265   \else
10266     \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%
10267   \fi
10268 }

```

`\glstrifhyphenstart`

```

10269 \def\@glstrifhyphenstart#1#2\@end@glstrifhyphenstart#3#4{%
10270   \ifx-#1\relax#3\else #4\fi
10271 }

```

`\glstrlonghyphenshort`

```
\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.

```
10272 \newcommand*{\glstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10273   {%
```

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.

```
10274   \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{}%
```

```

10275 \glsfirstlonghyphenfont{#2\ifglxtrinsertinside{#4}\fi}%
10276 \ifglxtrinsertinside\else{#4}\fi
10277 \glxtrfullsep{#1}%
10278 \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrinsertinside{#4}\fi}%
10279 \ifglxtrinsertinside\else{#4}\fi}%
10280 }%
10281 }

```

abbrvhyphenfont

```

10282 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%

```

abbrvhyphenfont

```

10283 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%

```

slonghyphenfont

```

10284 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%

```

tlonghyphenfont

```

10285 \newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%

```

The default short form suffix:

xtrhyphensuffix

```

10286 \newcommand*{\glxtrhyphensuffix}{\glxtrabbrvpluralsuffix}

```

en-short-hyphen Designed for use with the markwords attribute.

```

10287 \newabbreviationstyle{long-hyphen-short-hyphen}%
10288 {%
10289 \renewcommand*{\CustomAbbreviationFields}{%
10290 name={\glxtrlongshortname},
10291 sort={\the\glsshorttok},
10292 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10293 \protect\glxtrfullsep{\the\glslabeltok}%
10294 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10295 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10296 \protect\glxtrfullsep{\the\glslabeltok}%
10297 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10298 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10299 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10300 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10301 \glshasattribute{\the\glslabeltok}{regular}%
10302 {%
10303 \glissetattribute{\the\glslabeltok}{regular}{false}%
10304 }%
10305 {}%
10306 }%
10307 }%

```

```

10308 {%
10309   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10310   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10311   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10312   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10313   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10314   \renewcommand*{\glxtrfullformat}[2]{%
10315     \glxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10316   }%
10317   \renewcommand*{\glxtrfullplformat}[2]{%
10318     \glxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
10319     {\glsaccessshortpl{##1}}{##2}%
10320   }%
10321   \renewcommand*{\Glsxtrfullformat}[2]{%
10322     \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10323   }%
10324   \renewcommand*{\Glsxtrfullplformat}[2]{%
10325     \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10326     {\glsaccessshortpl{##1}}{##2}%
10327   }%
10328 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10329 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10330 {%
10331   \renewcommand*{\CustomAbbreviationFields}{%
10332     name={\glxtrlongshortdescname},
10333     sort={\glxtrlongshortdescsort},
10334     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}}%
10335     \protect\glxtrfullsep{\the\glslabeltok}%
10336     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10337     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}}%
10338     \protect\glxtrfullsep{\the\glslabeltok}%
10339     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10340     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10341     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10342   }%

```

Unset the regular attribute if it has been set.

```

10343   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10344     \glshasattribute{\the\glslabeltok}{regular}%
10345     {%
10346       \glissetattribute{\the\glslabeltok}{regular}{false}%
10347     }%
10348   }%
10349 }%
10350 }%
10351 {%

```

```
10352 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10353 }
```

onghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```
10354 \newcommand*{\glxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
10355 {%
```

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.

```
10356 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
10357 \glfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10358 \ifglxtrininsertinside\else{#3}\fi
10359 }%
10360 }
```

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.

```
10361 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10362 {%
10363 \renewcommand*{\CustomAbbreviationFields}{%
10364 name={\glxtrlongnoshortdescname},
10365 sort={\expandonce\glxtrorglong},
10366 first={\protect\glfirstlonghyphenfont{\the\glslongtok}},%
10367 firstplural={\protect\glfirstlonghyphenfont{\the\glslongpltok}},%
10368 plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10369 }%
```

Unset the regular attribute if it has been set.

```
10370 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10371 \glshasattribute{\the\glslabeltok}{regular}%
10372 {%
10373 \glissetattribute{\the\glslabeltok}{regular}{false}%
10374 }%
10375 {}%
10376 }%
10377 }%
10378 {%
10379 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10380 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
```

```

10381 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
10382 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
10383 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
10384 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10385 \renewcommand*\glsxtrsubsequentfmt[2]{%
10386   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10387 }%
10388 \renewcommand*\glsxtrsubsequentplfmt[2]{%
10389   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10390 }%
10391 \renewcommand*\Glsxtrsubsequentfmt[2]{%
10392   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10393 }%
10394 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
10395   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10396 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10397 \renewcommand*\glsxtrinlinefullformat[2]{%
10398   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10399   \glsxtrfullsep{##1}%
10400   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10401 }%
10402 \renewcommand*\glsxtrinlinefullplformat[2]{%
10403   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10404   \glsxtrfullsep{##1}%
10405   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10406 }%
10407 \renewcommand*\Glsxtrinlinefullformat[2]{%
10408   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10409   \glsxtrfullsep{##1}%
10410   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10411 }%
10412 \renewcommand*\Glsxtrinlinefullplformat[2]{%
10413   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10414   \glsxtrfullsep{##1}%
10415   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10416 }%

```

The first use full form only displays the long form.

```

10417 \renewcommand*\glsxtrfullformat[2]{%
10418   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10419 }%
10420 \renewcommand*\glsxtrfullplformat[2]{%
10421   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10422 }%
10423 \renewcommand*\Glsxtrfullformat[2]{%
10424   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10425 }%

```

```

10426 \renewcommand*{\Glsxtrfullplformat}[2]{%
10427   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10428 }%
10429 }

```

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.

```

10430 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10431 {%
10432   \renewcommand*{\CustomAbbreviationFields}{%
10433     name={\glsxtrlongnoshortname},
10434     sort={\the\glsshorttok},
10435     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10436     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10437     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10438     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10439     description={\the\glslongtok}%
10440   }%

```

Unset the regular attribute if it has been set.

```

10441 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10442   \glshasattribute{\the\glslabeltok}{regular}%
10443   {%
10444     \glissetattribute{\the\glslabeltok}{regular}{false}%
10445   }%
10446   {}%
10447 }%
10448 }%
10449 {%
10450   \GlsXtrUseAbbrStyleFmts{long-desc}%
10451 }

```

glsxtrlonghyphen `\glsxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10452 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10453 {%
10454   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}}%
10455   \glsfirstlonghyphenfont{#1}%
10456 }%
10457 }

```

posthyphenshort

`\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.

```

10458 \newcommand*{\glxtrposthyphenshort}[2]{%
10459   {%
10460     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}}%
10461     \ifglxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10462     \glxtrfullsep{#1}%
10463     \glxtrparen
10464     {\glsfirstabbrvhyphenfont{\glsenentryshort{#1}}\ifglxtrininsertinside{#2}\fi}%
10465     \ifglxtrininsertinside\else{#2}\fi
10466   }%
10467 }%
10468 }

```

hyphensubsequent

`\glxtrposthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10469 \newcommand*{\glxtrposthyphensubsequent}[2]{%
10470   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
10471   \ifglxtrininsertinside \else{#2}\fi
10472 }

```

postshort-hyphen

Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10473 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10474 {%
10475   \renewcommand*{\CustomAbbreviationFields}{%
10476     name={\glxtrlongshortname},
10477     sort={\the\glsshorttok},
10478     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10479     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10480     plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
10481     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10482   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10483     \csdef{glxtrpostlink\glscategorylabel}{%
10484       \glxtrifwasfirstuse
10485       {%
10486         \glxtrposthyphenshort{\glslabel}{\glinsert}%
10487       }%
10488     }%

```

Put the insertion into the post-link:


```

10489     \glxtrposthyphensubsequent{\glslabel}{\glinsert}%
10490 }%
10491 }%
10492 \glshasattribute{\the\glslabeltok}{regular}%
10493 {%
10494     \glissetattribute{\the\glslabeltok}{regular}{false}%
10495 }%
10496 {}%
10497 }%
10498 }%
10499 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10500 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
10501 \renewcommand*\glsabbrvfont[1]{\glsabbrvhyphenfont{##1}}%
10502 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvhyphenfont{##1}}%
10503 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
10504 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10505 \renewcommand*\glxtrsubsequentfmt[2]{%
10506     \glsabbrvfont{\glsaccessshort{##1}}%
10507 }%
10508 \renewcommand*\glxtrsubsequentplfmt[2]{%
10509     \glsabbrvfont{\glsaccessshortpl{##1}}%
10510 }%
10511 \renewcommand*\Glsxtrsubsequentfmt[2]{%
10512     \glsabbrvfont{\Glsaccessshort{##1}}%
10513 }%
10514 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
10515     \glsabbrvfont{\Glsaccessshortpl{##1}}%
10516 }%

```

First use full form:

```

10517 \renewcommand*\glxtrfullformat[2]{%
10518     \glxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10519 }%
10520 \renewcommand*\glxtrfullplformat[2]{%
10521     \glxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10522 }%
10523 \renewcommand*\Glsxtrfullformat[2]{%
10524     \glxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10525 }%
10526 \renewcommand*\Glsxtrfullplformat[2]{%
10527     \glxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10528 }%

```

In-line format.

```

10529 \renewcommand*\glxtrinlinefullformat[2]{%
10530     \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10531     \ifglxtrininsertinside{##2}\fi}%

```

```

10532 \ifglxtrinsertinside \else{##2}\fi
10533 }%
10534 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10535 \glsfirslonghyphenfont{\glssaccesslongpl{##1}%
10536 \ifglxtrinsertinside{##2}\fi}%
10537 \ifglxtrinsertinside \else{##2}\fi
10538 }%
10539 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10540 \glsfirslonghyphenfont{\Glsaccesslong{##1}%
10541 \ifglxtrinsertinside{##2}\fi}%
10542 \ifglxtrinsertinside \else{##2}\fi
10543 }%
10544 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10545 \glsfirslonghyphenfont{\Glsaccesslongpl{##1}%
10546 \ifglxtrinsertinside{##2}\fi}%
10547 \ifglxtrinsertinside \else{##2}\fi
10548 }%
10549 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10550 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10551 {%
10552 \renewcommand*{\CustomAbbreviationFields}{%
10553 name={\glxtrlongshortdescname},
10554 sort={\glxtrlongshortdescsort},%
10555 first={\protect\glsfirslonghyphenfont{\the\glslongtok}},%
10556 firstplural={\protect\glsfirslonghyphenfont{\the\glslongpltok}},%
10557 text={\protect\glssabrvhyphenfont{\the\glsshorttok}},%
10558 plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}}}%
10559 }%
10560 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10561 \csdef{glxtrpostlink\glscategorylabel}{%
10562 \glxtrifwasfirstuse
10563 {%
10564 \glxtrposthyphenshort{\glslabel}{\glssinsert}%
10565 }%
10566 {%

```

Put the insertion into the post-link:

```

10567 \glxtrposthyphensubsequent{\glslabel}{\glssinsert}%
10568 }%
10569 }%
10570 \glshasattribute{\the\glslabeltok}{regular}%
10571 {%
10572 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10573 }%
10574 {}%
10575 }%
10576 }%
10577 {%

```

```

10578 \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10579 }

```

rshorthyphenlong

```
\glstrshorthyphenlong{<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.

```
10580 \newcommand*{\glstrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10581 {%
```

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.)

```

10582 \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{}%
10583 \glsfirstabbrvhyphenfont{#2\ifglstrinsertinside{#4}\fi}%
10584 \ifglstrinsertinside\else{#4}\fi
10585 \glstrfullsep{#1}%
10586 \glstrparen{\glsfirstlonghyphenfont{#3\ifglstrinsertinside{#4}\fi}%
10587 \ifglstrinsertinside\else{#4}\fi}%
10588 }%
10589 }

```

hen-long-hyphen

Designed for use with the `markwords` attribute.

```

10590 \newabbreviationstyle{short-hyphen-long-hyphen}%
10591 {%
10592 \renewcommand*{\CustomAbbreviationFields}{%
10593 name={\glstrshortlongname},
10594 sort={\the\glsshorttok},
10595 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10596 \protect\glstrfullsep{\the\glslabeltok}%
10597 \glstrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10598 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10599 \protect\glstrfullsep{\the\glslabeltok}%
10600 \glstrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10601 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10602 description={\protect\glslonghyphenfont{\the\glslongtok}}}

```

Unset the regular attribute if it has been set.

```

10603 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10604 \glshasattribute{\the\glslabeltok}{regular}%
10605 {%
10606 \glissetattribute{\the\glslabeltok}{regular}{false}%
10607 }%
10608 {}%
10609 }%

```

```

10610 }%
10611 {%
10612   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10613   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10614   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10615   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10616   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10617   \renewcommand*{\glxtrfullformat}[2]{%
10618     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10619   }%
10620   \renewcommand*{\glxtrfullplformat}[2]{%
10621     \glxtrshorthyphenlong{##1}%
10622     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10623   }%
10624   \renewcommand*{\Glsxtrfullformat}[2]{%
10625     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10626   }%
10627   \renewcommand*{\Glsxtrfullplformat}[2]{%
10628     \glxtrshorthyphenlong{##1}%
10629     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10630   }%
10631 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10632 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10633 {%
10634   \renewcommand*{\CustomAbbreviationFields}{%
10635     name={\glxtrshortlongdescname},
10636     sort={\glxtrshortlongdescsort},
10637     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
10638     \protect\glxtrfullsep{\the\glslabeltok}%
10639     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10640     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
10641     \protect\glxtrfullsep{\the\glslabeltok}%
10642     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10643     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10644     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10645   }%

```

Unset the regular attribute if it has been set.

```

10646   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10647     \glshasattribute{\the\glslabeltok}{regular}%
10648     {%
10649       \glissetattribute{\the\glslabeltok}{regular}{false}%
10650     }%
10651   }%
10652 }%
10653 }%

```

```

10654 {%
10655   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10656 }

```

`\glstrshorthyphen`
`\glstrshorthyphen{<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.

```

10657 \newcommand*{\glstrshorthyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10658 {%
10659   \glstrifhyphenstart{#3}{\def\glstrwordsep{-}}}%
10660   \glsfirstabbrvhyphenfont{#1}%
10661 }%
10662 }

```

`\glstrposthyphenlong`
`\glstrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glstrshorthyphenlong` but omits the `<short>` part. This always uses the singular long form.

```

10663 \newcommand*{\glstrposthyphenlong}[2]{%
10664 {%
10665   \glstrifhyphenstart{#2}{\def\glstrwordsep{-}}}%
10666   \ifglstrinsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10667   \glstrfullsep{#1}%
10668   \glstrparen
10669   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglstrinsertinside{#2}\fi}%
10670   \ifglstrinsertinside\else{#2}\fi
10671 }%
10672 }%
10673 }

```

`postlong-hyphen` Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10674 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10675 {%
10676   \renewcommand*{\CustomAbbreviationFields}{%
10677     name={\glstrshortlongname},
10678     sort={\the\glsshorttok},
10679     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10680     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10681     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%

```

```

10682     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10683 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10684   \csdef{glsxtrpostlink\glscategorylabel}{%
10685     \glsxtrifwasfirstuse
10686     {%
10687       \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
10688     }%
10689     {%

```

Put the insertion into the post-link:

```

10690       \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10691     }%
10692   }%
10693   \glsattribute{\the\glslabeltok}{regular}%
10694   {%
10695     \glssetattribute{\the\glslabeltok}{regular}{false}%
10696   }%
10697   {}%
10698 }%
10699 }%
10700 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10701 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
10702 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10703 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10704 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10705 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10706 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10707   \glsabbrvfont{\glsaccessshort{##1}}%
10708 }%
10709 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10710   \glsabbrvfont{\glsaccessshortpl{##1}}%
10711 }%
10712 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10713   \glsabbrvfont{\Glsaccessshort{##1}}%
10714 }%
10715 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10716   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10717 }%

```

First use full form:

```

10718 \renewcommand*{\glsxtrfullformat}[2]{%
10719   \glsxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10720 }%
10721 \renewcommand*{\glsxtrfullplformat}[2]{%
10722   \glsxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10723 }%
10724 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

10725 \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10726 }%
10727 \renewcommand*{\Glsxtrfullplformat}[2]{%
10728 \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10729 }%

```

In-line format. Commands like `\glxtrfull` set `\glinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10730 \renewcommand*{\glxtrinlinefullformat}[2]{%
10731 \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
10732 \ifglxtrininsertinside{##2}\fi}%
10733 \ifglxtrininsertinside \else{##2}\fi
10734 }%
10735 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10736 \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
10737 \ifglxtrininsertinside{##2}\fi}%
10738 \ifglxtrininsertinside \else{##2}\fi
10739 }%
10740 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10741 \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
10742 \ifglxtrininsertinside{##2}\fi}%
10743 \ifglxtrininsertinside \else{##2}\fi
10744 }%
10745 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10746 \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
10747 \ifglxtrininsertinside{##2}\fi}%
10748 \ifglxtrininsertinside \else{##2}\fi
10749 }%
10750 }

```

`ong-hyphen-desc` Like `short-hyphen-postlong-hyphen` but the description must be supplied by the user.

```

10751 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10752 {%
10753 \renewcommand*{\CustomAbbreviationFields}{%
10754 name={\glxtrshortlongdescname},
10755 sort={\glxtrshortlongdescsort},%
10756 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10757 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10758 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10759 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10760 }%
10761 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10762 \csdef{glxtrpostlink\glscategorylabel}{%
10763 \glxtrifwasfirstuse
10764 {%
10765 \glxtrposthyphenlong{\glslabel}{\glinsert}%
10766 }%
10767 }%

```

Put the insertion into the post-link:

```

10768      \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
10769    }%
10770  }%
10771  \glshasattribute{\the\glslabeltok}{regular}%
10772  {%
10773    \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10774    }%
10775    {}%
10776  }%
10777 }%
10778 {%
10779   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10780 }

```

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

```
10781 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
10782 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
10783 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
10784 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10785 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```

10786 \newcommand*{\glsxtronlyname}{%
10787   \protect\glsabbrvonlyfont{\the\glsshorttok}%
10788 }

```

only-short-only

```

10789 \newabbreviationstyle{long-only-short-only}%
10790 {%
10791   \renewcommand*{\CustomAbbreviationFields}{%
10792     name={\glsxtronlyname},
10793     sort={\the\glsshorttok},
10794     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10795     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10796     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
10797     description={\protect\glslongonlyfont{\the\glslongtok}}}%

```


Unset the regular attribute if it has been set.

```

10798 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10799   \glshasattribute{\the\glslabeltok}{regular}%
10800   {%
10801     \glsselattribute{\the\glslabeltok}{regular}{false}%
10802   }%
10803   {}%
10804 }%
10805 }%
10806 {%
10807 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
10808 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
10809 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
10810 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
10811 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%

```

The first use full form doesn't show the short form.

```

10812 \renewcommand*{\glsxtrfullformat}[2]{%
10813   \glsfirstlongonlyfont{\glssaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10814   \ifglsxtrinsertinside\else##2\fi
10815 }%
10816 \renewcommand*{\glsxtrfullplformat}[2]{%
10817   \glsfirstlongonlyfont{\glssaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10818   \ifglsxtrinsertinside\else##2\fi
10819 }%
10820 \renewcommand*{\Glsxtrfullformat}[2]{%
10821   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10822   \ifglsxtrinsertinside\else##2\fi
10823 }%
10824 \renewcommand*{\Glsxtrfullplformat}[2]{%
10825   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10826   \ifglsxtrinsertinside\else##2\fi
10827 }%

```

The inline full form does show the short form.

```

10828 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10829   \glsfirstlongonlyfont{\glssaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10830   \ifglsxtrinsertinside\else##2\fi
10831   \glsxtrfullsep{##1}%
10832   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshort{##1}}}%
10833 }%
10834 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10835   \glsfirstlongonlyfont{\glssaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10836   \ifglsxtrinsertinside\else##2\fi
10837   \glsxtrfullsep{##1}%
10838   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10839 }%
10840 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10841   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10842   \ifglsxtrinsertinside\else##2\fi

```

```

10843 \glxtrfullsep{##1}%
10844 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10845 }%
10846 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10847 \glsfirstlongonlyfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10848 \ifglxtrinsertinside\else##2\fi
10849 \glxtrfullsep{##1}%
10850 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10851 }%
10852 }

```

xtronlydescsort

```

10853 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

10854 \newcommand*{\glxtronlydescname}{%
10855 \protect\glslongfont{\the\glslongtok}%
10856 }

```

short-only-desc

```

10857 \newabbreviationstyle{long-only-short-only-desc}%
10858 {%
10859 \renewcommand*{\CustomAbbreviationFields}{%
10860 name={\glxtronlydescname},
10861 sort={\glxtronlydescsort},%
10862 first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10863 firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10864 text={\protect\glssabbrvonlyfont{\the\glssshorttok}},%
10865 plural={\protect\glssabbrvonlyfont{\the\glssshortpltok}}}%
10866 }%

```

Unset the regular attribute if it has been set.

```

10867 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10868 \glshasattribute{\the\glslabeltok}{regular}%
10869 {%
10870 \glissetattribute{\the\glslabeltok}{regular}{false}%
10871 }%
10872 {}%
10873 }%
10874 }%
10875 {%
10876 \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10877 }

```

1.8 Using Entries in Headings

There are four main problems with using entries in sectioning commands: they can mess with the first use flag if they end up in the table of contents, they can add unwanted numbers to the entry's location list, the label is corrupted if used inside `\MakeUppercase` (which

is used by the default headings style) and they need to be expandable for PDF bookmarks. The glossaries package therefore recommends the use of the expandable commands, such as `\glsentryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glsabbrvfont`. The commands below are an attempt to get around these problems.

The PDF bookmark issue can easily be fixed with `hyperref`'s `\texorpdfstring` which can simply use the expandable command in the PDF string case. The \TeX string case can now use `\glsxtrshort` with the `noindex` key set, which prevents the unwanted additions to the location list, and the `hyper` key set to false, which prevents the problem of nested links. This just leaves one thing left that needs to be dealt with, and that's what to do if the heading style uses `\MakeUppercase`.

Note that `glossaries` automatically loads `textcase`, so the label can be protected from case change with `textcase`'s `\NoCaseChange`. This means that we don't have a problem provided the page style uses `\MakeTextUppercase`, but the default heading page style uses `\MakeUppercase`.

To get around this, save the original definition of `\markboth` and `\markright` and adjust it so that `\MakeUppercase` is temporarily redefined to `\MakeTextUppercase`. Some packages or classes redefine these commands, so we can't just assume they still have the original kernel definition.

`\markright` Save original definition:

```
10878 \let\@glsxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
10879 \renewcommand*{\markright}[1]{%
10880   \glsxtrmarkhook
10881   \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
10882   \glsxtrrestoremarkhook
10883 }
```

`\markboth` Save original definition:

```
10884 \let\@glsxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10885 \renewcommand*{\markboth}[2]{%
10886   \glsxtrmarkhook
10887   \@glsxtr@org@markboth
10888   {\@glsxtrinmark#1\@glsxtrnotinmark}%
10889   {\@glsxtrinmark#2\@glsxtrnotinmark}%
10890   \glsxtrrestoremarkhook
10891 }
```

Also do this for `\@starttoc`

`\@starttoc` Save original definition:

```
10892 \let\@glsxtr@org@@starttoc\@starttoc
```

Redefine:

```
10893 \renewcommand*{\@starttoc}[1]{%
```

```

10894 \glxtrmarkhook
10895 \@glxtrinmark
10896 \@glxtr@org@@starttoc{#1}%
10897 \@glxtrnotinmark
10898 \glxtrrestoremarkhook
10899 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

10900 \newcommand*{\glxtrRevertMarks}{%
10901   \let\markright\@glxtr@org@markright
10902   \let\markboth\@glxtr@org@markboth
10903   \let\@starttoc\@glxtr@org@@starttoc
10904 }

```

rRevertTocMarks Just restores \@starttoc.

```

10905 \newcommand*{\glxtrRevertTocMarks}{%
10906   \let\@starttoc\@glxtr@org@@starttoc
10907 }

```

\glxtrifinmark

```

10908 \newcommand*{\glxtrifinmark}[2]{#2}

```

\@glxtrinmark

```

10909 \newrobustcmd*{\@glxtrinmark}{%
10910   \let\glxtrifinmark\@firstoftwo
10911 }

```

glxtrnotinmark

```

10912 \newrobustcmd*{\@glxtrnotinmark}{%
10913   \let\glxtrifinmark\@secondoftwo
10914 }

```

eorpdforheading

```

10915 \ifdef\texorpdfstring
10916 {
10917   \newcommand*{\glxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
10918 }
10919 {
10920   \newcommand*{\glxtrtitleorpdforheading}[3]{#1}
10921 }

```

\glxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```

10922 \newcommand*{\glxtrmarkhook}{%

```

Save current definitions:

```
10923 \let\@glsxtr@org@MakeUppercase\MakeUppercase
10924 \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
10925 \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
10926 \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
10927 \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
10928 \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
10929 \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
10930 \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
10931 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
10932 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
10933 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
10934 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
10935 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
10936 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
10937 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
10938 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
10939 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
10940 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
10941 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
10942 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
10943 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
10944 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
10945 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
10946 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
10947 \let\glsxtrifinmark\@firstoftwo
10948 \let\MakeUppercase\MakeTextUppercase
10949 \let\glsxtrtitleorpdforheading\@thirdofthree
10950 \let\glsxtrtitleshort\glsxtrheadshort
10951 \let\glsxtrtitleshortpl\glsxtrheadshortpl
10952 \let\Glsxtrtitleshort\Glsxtrheadshort
10953 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
10954 \let\glsxtrtitlename\glsxtrheadname
10955 \let\Glsxtrtitlename\Glsxtrheadname
10956 \let\glsxtrtitletext\glsxtrheadtext
10957 \let\Glsxtrtitletext\Glsxtrheadtext
10958 \let\glsxtrtitleplural\glsxtrheadplural
10959 \let\Glsxtrtitleplural\Glsxtrheadplural
10960 \let\glsxtrtitlefirst\glsxtrheadfirst
10961 \let\Glsxtrtitlefirst\Glsxtrheadfirst
10962 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
10963 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
10964 \let\glsxtrtitlelong\glsxtrheadlong
10965 \let\glsxtrtitlelongpl\glsxtrheadlongpl
10966 \let\Glsxtrtitlelong\Glsxtrheadlong
10967 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
10968 \let\glsxtrtitlefull\glsxtrheadfull
10969 \let\glsxtrtitlefullpl\glsxtrheadfullpl
```

```

10970 \let\Glsxtrtitlefull\Glsxtrheadfull
10971 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
10972 }

```

`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.)

```

10973 \newcommand*{\glxtrrestoremarkhook}{%
10974   \let\glxtrifinmark\@secondoftwo
10975   \let\MakeUppercase\@glxtr@org@MakeUppercase
10976   \let\glxtrtitleorpdforheading\@glxtr@org@glxtrtitleorpdforheading
10977   \let\glxtrtitleshort\@glxtr@org@glxtrtitleshort
10978   \let\glxtrtitleshortpl\@glxtr@org@glxtrtitleshortpl
10979   \let\Glsxtrtitleshort\@glxtr@org@Glsxtrtitleshort
10980   \let\Glsxtrtitleshortpl\@glxtr@org@Glsxtrtitleshortpl
10981   \let\glxtrtitlename\@glxtr@org@glxtrtitlename
10982   \let\Glsxtrtitlename\@glxtr@org@Glsxtrtitlename
10983   \let\glxtrtitletext\@glxtr@org@glxtrtitletext
10984   \let\Glsxtrtitletext\@glxtr@org@Glsxtrtitletext
10985   \let\glxtrtitleplural\@glxtr@org@glxtrtitleplural
10986   \let\Glsxtrtitleplural\@glxtr@org@Glsxtrtitleplural
10987   \let\glxtrtitlefirst\@glxtr@org@glxtrtitlefirst
10988   \let\Glsxtrtitlefirst\@glxtr@org@Glsxtrtitlefirst
10989   \let\glxtrtitlefirstplural\@glxtr@org@glxtrtitlefirstplural
10990   \let\Glsxtrtitlefirstplural\@glxtr@org@Glsxtrtitlefirstplural
10991   \let\glxtrtitlelong\@glxtr@org@glxtrtitlelong
10992   \let\glxtrtitlelongpl\@glxtr@org@glxtrtitlelongpl
10993   \let\Glsxtrtitlelong\@glxtr@org@Glsxtrtitlelong
10994   \let\Glsxtrtitlelongpl\@glxtr@org@Glsxtrtitlelongpl
10995   \let\glxtrtitlefull\@glxtr@org@glxtrtitlefull
10996   \let\glxtrtitlefullpl\@glxtr@org@glxtrtitlefullpl
10997   \let\Glsxtrtitlefull\@glxtr@org@Glsxtrtitlefull
10998   \let\Glsxtrtitlefullpl\@glxtr@org@Glsxtrtitlefullpl
10999 }

```

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.

```

11000 \newcommand*{\glxtrheadshort}[1]{%
11001   \protect\NoCaseChange
11002   {%
11003     \gl@ifattribute{#1}{headuc}{true}%
11004     {%
11005       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11006     }%
11007     {%
11008       \glxtrshort[noindex,hyper=false]{#1}[]%

```

```

11009 }%
11010 }%
11011 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11012 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11013   \glsxtrshort[noindex,hyper=false]{#1}[]%
11014 }

```

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.

```

11015 \newcommand*{\lsxtrheadshortpl}[1]{%
11016   \protect\NoCaseChange
11017   {%
11018     \glsifattribute{#1}{headuc}{true}%
11019     {%
11020       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11021     }%
11022     {%
11023       \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11024     }%
11025   }%
11026 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

11027 \newrobustcmd*{\gxtrtitleshortpl}[1]{%
11028   \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11029 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

11030 \newcommand*{\Glsxtrheadshort}[1]{%
11031   \protect\NoCaseChange
11032   {%
11033     \glsifattribute{#1}{headuc}{true}%
11034     {%
11035       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11036     }%
11037     {%
11038       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11039     }%
11040   }%
11041 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11042 \newrobustcmd*{\Glsxtrtitleshort}[1]{%

```

```

11043 \Glsxtrshort[noindex,hyper=false]{#1}[]%
11044 }

```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```

11045 \newcommand*{\Glsxtrheadshortpl}[1]{%
11046 \protect\NoCaseChange
11047 {%
11048 \glsifattribute{#1}{headuc}{true}%
11049 {%
11050 \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11051 }%
11052 {%
11053 \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11054 }%
11055 }%
11056 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11057 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11058 \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11059 }

```

\glsxtrheadname As above but for the name value.

```

11060 \newcommand*{\glsxtrheadname}[1]{%
11061 \protect\NoCaseChange
11062 {%
11063 \glsifattribute{#1}{headuc}{true}%
11064 {%
11065 \GLSname[noindex,hyper=false]{#1}[]%
11066 }%
11067 {%
11068 \glsname[noindex,hyper=false]{#1}[]%
11069 }%
11070 }%
11071 }

```

glsxtrtitlename Command to display name value in section title and table of contents.

```

11072 \newrobustcmd*{\glsxtrtitlename}[1]{%
11073 \glsname[noindex,hyper=false]{#1}[]%
11074 }

```

\Glsxtrheadname First letter converted to upper case

```

11075 \newcommand*{\Glsxtrheadname}[1]{%
11076 \protect\NoCaseChange
11077 {%
11078 \glsifattribute{#1}{headuc}{true}%

```



```

11079   {%
11080     \GLSname[noindex,hyper=false]{#1}[]%
11081   }%
11082   {%
11083     \Glsname[noindex,hyper=false]{#1}[]%
11084   }%
11085 }%
11086 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11087 %\changes{1.21}{2017-11-03}{new}
11088 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11089   \Glsname[noindex,hyper=false]{#1}[]%
11090 }

```

\glsxtrheadtext As above but for the text value.

```

11091 \newcommand*{\glsxtrheadtext}[1]{%
11092   \protect\NoCaseChange
11093   {%
11094     \glsifattribute{#1}{headuc}{true}%
11095     {%
11096       \GLStext[noindex,hyper=false]{#1}[]%
11097     }%
11098     {%
11099       \glstext[noindex,hyper=false]{#1}[]%
11100     }%
11101   }%
11102 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

11103 \newrobustcmd*{\glsxtrtitletext}[1]{%
11104   \glstext[noindex,hyper=false]{#1}[]%
11105 }

```

\Glsxtrheadtext First letter converted to upper case

```

11106 \newcommand*{\Glsxtrheadtext}[1]{%
11107   \protect\NoCaseChange
11108   {%
11109     \glsifattribute{#1}{headuc}{true}%
11110     {%
11111       \GLStext[noindex,hyper=false]{#1}[]%
11112     }%
11113     {%
11114       \Glstext[noindex,hyper=false]{#1}[]%
11115     }%
11116   }%
11117 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```
11118 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11119   \Gls{text[noindex,hyper=false]}{#1}[]%
11120 }
```

lsxtrheadplural As above but for the plural value.

```
11121 \newcommand*{\glsxtrheadplural}[1]{%
11122   \protect\NoCaseChange
11123   {%
11124     \glsifattribute{#1}{headuc}{true}%
11125     {%
11126       \GLSplural[noindex,hyper=false]{#1}[]%
11127     }%
11128     {%
11129       \glsplural[noindex,hyper=false]{#1}[]%
11130     }%
11131   }%
11132 }
```

sxtrtitleplural Command to display plural value in section title and table of contents.

```
11133 \newrobustcmd*{\glsxtrtitleplural}[1]{%
11134   \glsplural[noindex,hyper=false]{#1}[]%
11135 }
```

lsxtrheadplural Convert first letter to upper case.

```
11136 \newcommand*{\Glsxtrheadplural}[1]{%
11137   \protect\NoCaseChange
11138   {%
11139     \glsifattribute{#1}{headuc}{true}%
11140     {%
11141       \GLSplural[noindex,hyper=false]{#1}[]%
11142     }%
11143     {%
11144       \Glsplural[noindex,hyper=false]{#1}[]%
11145     }%
11146   }%
11147 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11148 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11149   \Glsplural[noindex,hyper=false]{#1}[]%
11150 }
```

glsxtrheadfirst As above but for the first value.

```
11151 \newcommand*{\glsxtrheadfirst}[1]{%
11152   \protect\NoCaseChange
```

```

11153 {%
11154   \glsifattribute{#1}{headuc}{true}%
11155   {%
11156     \GLSfirst[noindex,hyper=false]{#1}[]%
11157   }%
11158   {%
11159     \glsfirst[noindex,hyper=false]{#1}[]%
11160   }%
11161 }%
11162 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```

11163 \newrobustcmd*{\lsxtrtitlefirst}[1]{%
11164   \glsfirst[noindex,hyper=false]{#1}[]%
11165 }

```

Glsxtrheadfirst First letter converted to upper case

```

11166 \newcommand*{\Glsxtrheadfirst}[1]{%
11167   \protect\NoCaseChange
11168   {%
11169     \glsifattribute{#1}{headuc}{true}%
11170     {%
11171       \GLSfirst[noindex,hyper=false]{#1}[]%
11172     }%
11173     {%
11174       \Glsfirst[noindex,hyper=false]{#1}[]%
11175     }%
11176   }%
11177 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11178 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11179   \Glsfirst[noindex,hyper=false]{#1}[]%
11180 }

```

headfirstplural As above but for the firstplural value.

```

11181 \newcommand*{\glsxtrheadfirstplural}[1]{%
11182   \protect\NoCaseChange
11183   {%
11184     \glsifattribute{#1}{headuc}{true}%
11185     {%
11186       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11187     }%
11188     {%
11189       \glsfirstplural[noindex,hyper=false]{#1}[]%
11190     }%
11191   }%
11192 }

```

`titlefirstplural` Command to display firstplural value in section title and table of contents.

```
11193 \newrobustcmd*{\glxtrtitlefirstplural}[1]{%
11194   \glsfirstplural[noindex,hyper=false]{#1}[]%
11195 }
```

`headfirstplural` First letter converted to upper case

```
11196 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11197   \protect\NoCaseChange
11198   {%
11199     \glusifattribute{#1}{headuc}{true}%
11200     {%
11201       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11202     }%
11203     {%
11204       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11205     }%
11206   }%
11207 }
```

`titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11208 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
11209   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11210 }
```

`\glxtrheadlong` Command used to display long form in the page header.

```
11211 \newcommand*{\glxtrheadlong}[1]{%
11212   \protect\NoCaseChange
11213   {%
11214     \glusifattribute{#1}{headuc}{true}%
11215     {%
11216       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11217     }%
11218     {%
11219       \glsxtrlong[noindex,hyper=false]{#1}[]%
11220     }%
11221   }%
11222 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
11223 \newrobustcmd*{\glxstrtitlelong}[1]{%
11224   \glsxtrlong[noindex,hyper=false]{#1}[]%
11225 }
```

`lsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
11226 \newcommand*{\glxtrheadlongpl}[1]{%
```

```

11227 \protect\NoCaseChange
11228 {%
11229   \glsifattribute{#1}{headuc}{true}%
11230   {%
11231     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11232   }%
11233   {%
11234     \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11235   }%
11236 }%
11237 }

```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

11238 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%
11239   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11240 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

11241 \newcommand*{\Glsxtrheadlong}[1]{%
11242   \protect\NoCaseChange
11243   {%
11244     \glsifattribute{#1}{headuc}{true}%
11245     {%
11246       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11247     }%
11248     {%
11249       \Glsxtrlong[noindex,hyper=false]{#1}[]%
11250     }%
11251   }%
11252 }

```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11253 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11254   \Glsxtrlong[noindex,hyper=false]{#1}[]%
11255 }

```

`lsextrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```

11256 \newcommand*{\Glsxtrheadlongpl}[1]{%
11257   \protect\NoCaseChange
11258   {%
11259     \glsifattribute{#1}{headuc}{true}%
11260     {%
11261       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11262     }%
11263     {%

```

```

11264 \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11265 }%
11266 }%
11267 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11268 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11269 \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11270 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11271 \newcommand*{\glsxtrheadfull}[1]{%
11272 \protect\NoCaseChange
11273 {%
11274 \glsifattribute{#1}{headuc}{true}%
11275 {%
11276 \GLSxtrfull[noindex,hyper=false]{#1}[]%
11277 }%
11278 {%
11279 \glsxtrfull[noindex,hyper=false]{#1}[]%
11280 }%
11281 }%
11282 }

```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```

11283 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11284 \glsxtrfull[noindex,hyper=false]{#1}[]%
11285 }

```

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.

```

11286 \newcommand*{\glsxtrheadfullpl}[1]{%
11287 \protect\NoCaseChange
11288 {%
11289 \glsifattribute{#1}{headuc}{true}%
11290 {%
11291 \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11292 }%
11293 {%
11294 \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11295 }%
11296 }%
11297 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

11298 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%

```

```

11299 \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11300 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

11301 \newcommand*{\Glsxtrheadfull}[1]{%
11302 \protect\NoCaseChange
11303 {%
11304 \glsifattribute{#1}{headuc}{true}%
11305 {%
11306 \Glsxtrfull[noindex,hyper=false]{#1}[]%
11307 }%
11308 {%
11309 \Glsxtrfull[noindex,hyper=false]{#1}[]%
11310 }%
11311 }%
11312 }

```

\Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11313 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11314 \Glsxtrfull[noindex,hyper=false]{#1}[]%
11315 }

```

\Glsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

11316 \newcommand*{\Glsxtrheadfullpl}[1]{%
11317 \protect\NoCaseChange
11318 {%
11319 \glsifattribute{#1}{headuc}{true}%
11320 {%
11321 \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11322 }%
11323 {%
11324 \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11325 }%
11326 }%
11327 }

```

\Glsxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11328 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11329 \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11330 }

```

\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.

```

11331 \ifdef\texorpdfstring

```

```

11332 {
11333   \newcommand*{\glsfmtshort}[1]{%
11334     \texorpdfstring
11335       {\glsxtrtitleshort{#1}}%
11336       {\glsentryshort{#1}}%
11337   }
11338 }
11339 {
11340   \newcommand*{\glsfmtshort}[1]{%
11341     \glsxtrtitleshort{#1}}
11342 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11343 \ifdef\texorpdfstring
11344 {
11345   \newcommand*{\glsfmtshortpl}[1]{%
11346     \texorpdfstring
11347       {\glsxtrtitleshortpl{#1}}%
11348       {\glsentryshortpl{#1}}%
11349   }
11350 }
11351 {
11352   \newcommand*{\glsfmtshortpl}[1]{%
11353     \glsxtrtitleshortpl{#1}}
11354 }

```

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).

```

11355 \ifdef\texorpdfstring
11356 {
11357   \newcommand*{\Glsfmtshort}[1]{%
11358     \texorpdfstring
11359       {\Glsxtrtitleshort{#1}}%
11360       {\glsentryshort{#1}}%
11361   }
11362 }
11363 {
11364   \newcommand*{\Glsfmtshort}[1]{%
11365     \Glsxtrtitleshort{#1}}
11366 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11367 \ifdef\texorpdfstring
11368 {
11369   \newcommand*{\Glsfmtshortpl}[1]{%
11370     \texorpdfstring

```



```

11371     {\Glsxtrtitleshortpl{#1}}}%
11372     {\glsentryshortpl{#1}}}%
11373   }
11374 }
11375 {
11376   \newcommand*{\Glsfmtshortpl}[1]{%
11377     \Glsxtrtitleshortpl{#1}}
11378 }

```

`\glsfmtname` As above but for the name value.

```

11379 \ifdef\teorpdfstring
11380 {
11381   \newcommand*{\glsfmtname}[1]{%
11382     \teorpdfstring
11383     {\Glsxtrtitlename{#1}}}%
11384     {\glsentryname{#1}}}%
11385   }
11386 }
11387 {
11388   \newcommand*{\glsfmtname}[1]{%
11389     \Glsxtrtitlename{#1}}
11390 }

```

`\Glsfmtname` First letter converted to upper case.

```

11391 \ifdef\teorpdfstring
11392 {
11393   \newcommand*{\Glsfmtname}[1]{%
11394     \teorpdfstring
11395     {\Glsxtrtitlename{#1}}}%
11396     {\glsentryname{#1}}}%
11397   }
11398 }
11399 {
11400   \newcommand*{\Glsfmtname}[1]{%
11401     \Glsxtrtitlename{#1}}
11402 }

```

`\glsfmttext` As above but for the text value.

```

11403 \ifdef\teorpdfstring
11404 {
11405   \newcommand*{\glsfmttext}[1]{%
11406     \teorpdfstring
11407     {\Glsxtrtitletext{#1}}}%
11408     {\glsentrytext{#1}}}%
11409   }
11410 }
11411 {
11412   \newcommand*{\glsfmttext}[1]{%
11413     \Glsxtrtitletext{#1}}

```

11414 }

`\Glsfmttext` First letter converted to upper case.

```
11415 \ifdef\teorpdfstring
11416 {
11417   \newcommand*\Glsfmttext}[1]{%
11418     \teorpdfstring
11419     {\Glsxtrtitletext{#1}}%
11420     {\glstrytext{#1}}%
11421   }
11422 }
11423 {
11424   \newcommand*\Glsfmttext}[1]{%
11425     \Glsxtrtitletext{#1}}
11426 }
```

`\glsfmtplural` As above but for the plural value.

```
11427 \ifdef\teorpdfstring
11428 {
11429   \newcommand*\glsfmtplural}[1]{%
11430     \teorpdfstring
11431     {\glsxtrtitleplural{#1}}%
11432     {\glstryplural{#1}}%
11433   }
11434 }
11435 {
11436   \newcommand*\glsfmtplural}[1]{%
11437     \glsxtrtitleplural{#1}}
11438 }
```

`\Glsfmtplural` First letter converted to upper case.

```
11439 \ifdef\teorpdfstring
11440 {
11441   \newcommand*\Glsfmtplural}[1]{%
11442     \teorpdfstring
11443     {\Glsxtrtitleplural{#1}}%
11444     {\glstryplural{#1}}%
11445   }
11446 }
11447 {
11448   \newcommand*\Glsfmtplural}[1]{%
11449     \Glsxtrtitleplural{#1}}
11450 }
```

`\glsfmtfirst` As above but for the first value.

```
11451 \ifdef\teorpdfstring
11452 {
11453   \newcommand*\glsfmtfirst}[1]{%
11454     \teorpdfstring
```

```

11455     {\glsxtrtitlefirst{#1}}}%
11456     {\glsentryfirst{#1}}}%
11457   }
11458 }
11459 {
11460   \newcommand*{\Glsfmtfirst}[1]{%
11461     \glsxtrtitlefirst{#1}}
11462 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11463 \ifdef\txorpdfstring
11464 {
11465   \newcommand*{\Glsfmtfirst}[1]{%
11466     \txorpdfstring
11467     {\Glsxtrtitlefirst{#1}}}%
11468     {\glsentryfirst{#1}}}%
11469   }
11470 }
11471 {
11472   \newcommand*{\Glsfmtfirst}[1]{%
11473     \Glsxtrtitlefirst{#1}}
11474 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11475 \ifdef\txorpdfstring
11476 {
11477   \newcommand*{\glsfmtfirstpl}[1]{%
11478     \txorpdfstring
11479     {\glsxtrtitlefirstplural{#1}}}%
11480     {\glsentryfirstplural{#1}}}%
11481   }
11482 }
11483 {
11484   \newcommand*{\glsfmtfirstpl}[1]{%
11485     \glsxtrtitlefirstplural{#1}}
11486 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11487 \ifdef\txorpdfstring
11488 {
11489   \newcommand*{\Glsfmtfirstpl}[1]{%
11490     \txorpdfstring
11491     {\Glsxtrtitlefirstplural{#1}}}%
11492     {\glsentryfirstplural{#1}}}%
11493   }
11494 }
11495 {
11496   \newcommand*{\Glsfmtfirstpl}[1]{%
11497     \Glsxtrtitlefirstplural{#1}}

```

11498 }

`\glsfmtlong` As above but for the long value.

```
11499 \ifdef\txorpdfstring
11500 {
11501   \newcommand*\glsfmtlong[1]{%
11502     \txorpdfstring
11503     {\glsxtrtitlelong{#1}}%
11504     {\glsentrylong{#1}}%
11505   }
11506 }
11507 {
11508   \newcommand*\glsfmtlong[1]{%
11509     \glsxtrtitlelong{#1}}
11510 }
```

`\Glsfmtlong` First letter converted to upper case.

```
11511 \ifdef\txorpdfstring
11512 {
11513   \newcommand*\Glsfmtlong[1]{%
11514     \txorpdfstring
11515     {\Glsxtrtitlelong{#1}}%
11516     {\glsentrylong{#1}}%
11517   }
11518 }
11519 {
11520   \newcommand*\Glsfmtlong[1]{%
11521     \Glsxtrtitlelong{#1}}
11522 }
```

`\glsfmtlongpl` As above but for the longplural value.

```
11523 \ifdef\txorpdfstring
11524 {
11525   \newcommand*\glsfmtlongpl[1]{%
11526     \txorpdfstring
11527     {\glsxtrtitlelongpl{#1}}%
11528     {\glsentrylongpl{#1}}%
11529   }
11530 }
11531 {
11532   \newcommand*\glsfmtlongpl[1]{%
11533     \glsxtrtitlelongpl{#1}}
11534 }
```

`\Glsfmtlongpl` First letter converted to upper case.

```
11535 \ifdef\txorpdfstring
11536 {
11537   \newcommand*\Glsfmtlongpl[1]{%
11538     \txorpdfstring
```

```

11539     {\Glsxtrtitlelongpl{#1}}%
11540     {\glsentrylongpl{#1}}%
11541   }
11542 }
11543 {
11544   \newcommand*{\Glsfmtlongpl}[1]{%
11545     \Glsxtrtitlelongpl{#1}}
11546 }

```

`\glsfmtfull` In-line full format.

```

11547 \ifdef\teorpdfstring
11548 {
11549   \newcommand*{\glsfmtfull}[1]{%
11550     \teorpdfstring
11551     {\glsxtrtitlefull{#1}}%
11552     {\glsxtrinlinefullformat{#1}-{}}%
11553   }
11554 }
11555 {
11556   \newcommand*{\glsfmtfull}[1]{%
11557     \glsxtrtitlefull{#1}}
11558 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11559 \ifdef\teorpdfstring
11560 {
11561   \newcommand*{\Glsfmtfull}[1]{%
11562     \teorpdfstring
11563     {\Glsxtrtitlefull{#1}}%
11564     {\Glsxtrinlinefullformat{#1}-{}}%
11565   }
11566 }
11567 {
11568   \newcommand*{\Glsfmtfull}[1]{%
11569     \Glsxtrtitlefull{#1}}
11570 }

```

`\glsfmtfullpl` In-line full plural format.

```

11571 \ifdef\teorpdfstring
11572 {
11573   \newcommand*{\glsfmtfullpl}[1]{%
11574     \teorpdfstring
11575     {\glsxtrtitlefullpl{#1}}%
11576     {\glsxtrinlinefullplformat{#1}-{}}%
11577   }
11578 }
11579 {
11580   \newcommand*{\glsfmtfullpl}[1]{%
11581     \glsxtrtitlefullpl{#1}}

```

```
11582 }
```

`\Glsfmtfullpl` First letter converted to upper case.

```
11583 \ifdef\teorpdfstring
11584 {
11585   \newcommand*{\Glsfmtfullpl}[1]{%
11586     \teorpdfstring
11587     {\Glsxtrtitlefullpl{#1}}%
11588     {\Glsxtrinlinefullplformat{#1}{}}%
11589   }
11590 }
11591 {
11592   \newcommand*{\Glsfmtfullpl}[1]{%
11593     \Glsxtrtitlefullpl{#1}}
11594 }
```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```
11595 \newcommand*{\RequireGlossariesExtraLang}[1]{%
11596   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11597 }
```

`sariesExtraLang`

```
11598 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11599   \ProvidesFile{glossariesxtr-#1.ldf}%
11600 }
```

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.

```
11601 \newcommand{\glxtr@loaddialect}{%
11602   \IfTrackedLanguageFileExists{\this@dialect}%
11603   {glossariesxtr-}% prefix
11604   {.ldf}%
11605   {%
11606     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11607   }%
11608   {}% not found
```

If glossaries-extra-bib2gls has been loaded, `\@glstrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11609 \@glstrdialecthook
11610 }

11611 \@ifpackageloaded{tracklang}
11612 {%
11613   \AnyTrackedLanguages
11614   {%
11615     \ForEachTrackedDialect{\this@dialect}{\glstr@loaddialect}%
11616   }%
11617   {}%
11618 }
11619 {}

```

Load glossaries-extra-stylemods if required.

```

11620 \@glstr@redefstyles
    and set the style:
11621 \@glstr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for bib2gls and is automatically loaded by the record option.

```

11622 \NeedsTeXFormat{LaTeX2e}
11623 \ProvidesPackage{glossaries-extra-bib2gls}[2018/05/09 v1.31 (NLCT)]

```

These are some convenient macros for use with custom rules.

```

\glshex
11624 \newcommand*{\glshex}{\string\u}

```

```

\glscapturedgroup
11625 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's `save-child-count` resource option.

```

11626 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11627   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11628 }

```

`\glstrprovidecommand` For use in `@preamble`, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11629 \newcommand*{\glstrprovidecommand}{\providecommand}

```

`\glstr@wrglossarylocation` For use with `indexcounter` and bib2gls.

```

11630 \newcommand*{\glstr@wrglossarylocation}[2]{#1}

```

IndexCounterLink `\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```

11631 \ifdef\hyperref
11632 {%
11633   \newcommand*\GlsXtrIndexCounterLink[2]{%
11634     \glstrifhasfield{indexcounter}{#2}%
11635     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11636     {#1}%
11637   }
11638 }
11639 {
11640   \newcommand*\GlsXtrIndexCounterLink[2]{#1}
11641 }

```

\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.

```

11642 \newcommand*\GlsXtrDualField{dual}

```

sXtrDualBackLink `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```

11643 \newcommand*\GlsXtrDualBackLink[2]{%
11644   \glstrifhasfield{\GlsXtrDualField}{#2}%
11645   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11646   {#2}%
11647 }

```

TeXEntryAliases Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```

11648 \newcommand*\GlsXtrBibTeXEntryAliases{%
11649   article=bibtexentry,
11650   book=bibtexentry,
11651   booklet=bibtexentry,
11652   conference=bibtexentry,
11653   inbook=bibtexentry,
11654   incollection=bibtexentry,
11655   inproceedings=bibtexentry,
11656   manual=bibtexentry,

```



```

11657 mastersthesis=bibtexentry,
11658 misc=bibtexentry,
11659 phdthesis=bibtexentry,
11660 proceedings=bibtexentry,
11661 techreport=bibtexentry,
11662 unpublished=bibtexentry
11663 }

```

`\provideBibTeXFields` Convenient shortcut to define the standard BibTeX fields.

```

11664 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11665   \glsaddstoragekey{address}{}{\glsxtrbibaddress}%
11666   \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
11667   \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}%
11668   \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
11669   \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
11670   \glsaddstoragekey{howpublished}{}{\glsxtrbibhowpublished}%
11671   \glsaddstoragekey{institution}{}{\glsxtrbibinstitution}%
11672   \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
11673   \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
11674   \glsaddstoragekey{note}{}{\glsxtrbibnote}%
11675   \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
11676   \glsaddstoragekey{organization}{}{\glsxtrbiborganization}%
11677   \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
11678   \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}%
11679   \glsaddstoragekey{school}{}{\glsxtrbibschooll}%
11680   \glsaddstoragekey{series}{}{\glsxtrbibseries}%
11681   \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
11682   \glsaddstoragekey{bibtextype}{}{\glsxtrbibtype}%
11683   \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
11684 }

```

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 \LaTeX version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

`\Alpha`

```
11685 \providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
11686 \providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
11687 \providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
11688 \providecommand*{\Zeta}{\mathrm{Z}}
```

```

\Eta
11689 \providecommand*\Eta{\mathrm{H}}

\Iota
11690 \providecommand*\Iota{\mathrm{I}}

\Kappa
11691 \providecommand*\Kappa{\mathrm{K}}

\Mu
11692 \providecommand*\Mu{\mathrm{M}}

\Nu
11693 \providecommand*\Nu{\mathrm{N}}

\Omicron
11694 \providecommand*\Omicron{\mathrm{O}}

\Rho
11695 \providecommand*\Rho{\mathrm{P}}

\Tau
11696 \providecommand*\Tau{\mathrm{T}}

\Chi
11697 \providecommand*\Chi{\mathrm{X}}

\Digamma
11698 \providecommand*\Digamma{\mathrm{F}}

\omicron
11699 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11700 \@ifpackageloaded{upgreek}%
11701 {

\Upalpha
11702   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11703   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11704   \providecommand*\Upsilon{\mathrm{E}}

```

```

\Upzeta
11705 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11706 \providecommand*\Upeta{\mathrm{H}}

\Upiota
11707 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11708 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11709 \providecommand*\Upmu{\mathrm{M}}

\Upnu
11710 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11711 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11712 \providecommand*\Uprho{\mathrm{P}}

\Uptau
11713 \providecommand*\Uptau{\mathrm{T}}

\Upchi
11714 \providecommand*\Upchi{\mathrm{X}}

\upomicron
11715 \providecommand*\upomicron{\mathrm{o}}

11716}%
11717}% 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
;\glxtrcombiningspacingrules
,\glxtrhyphenrules
<\glxtrgeneralpuncrules
<\glxtrdigitrules
<\glxtrfractionrules
<\glxtrGeneralLatinIVrules
<\glxtrMathItalicGreekIrules
}
```

glxtrcontrolrules 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.

```
11718 \newcommand*{\glxtrcontrolrules}{%
11719 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
11720 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
11721 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
11722 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
11723 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
11724 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
11725 0010\string'\string=\glshex 0011
11726 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11727 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11728 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11729 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11730 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11731 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11732 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11733 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11734 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11735 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11736 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11737 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11738 }
```

glxtrspacerules These are space characters.

```
11739 \newcommand*{\glxtrspacerules}{%
11740 \string'\string'\string;
11741 \string'\glshex 00A0\string'\string;
11742 \string'\glshex 2000\string'\string;
11743 \string'\glshex 2001\string'\string;
11744 \string'\glshex 2002\string'\string;
11745 \string'\glshex 2003\string'\string;
```

```

11746 \string'\glshex 2004\string'\string;
11747 \string'\glshex 2005\string'\string;
11748 \string'\glshex 2006\string'\string;
11749 \string'\glshex 2007\string'\string;
11750 \string'\glshex 2008\string'\string;
11751 \string'\glshex 2009\string'\string;
11752 \string'\glshex 200A\string'\string;
11753 \string'\glshex 3000\string'
11754 }

```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11755 \newcommand*{\glxtrnonprintablerules}{%
11756 \string'\glshex FEFF\string'\string;
11757 \string'\glshex 000A\string'\string;
11758 \string'\glshex 0009\string'\string;
11759 \string'\glshex 000C\string'\string;
11760 \string'\glshex 000B\string'
11761 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

11762 \newcommand*{\glxtrcombiningdiacriticrules}{%
11763 \glxtrcombiningdiacriticIrules\string;
11764 \glxtrcombiningdiacriticIIrules\string;
11765 \glxtrcombiningdiacriticIIIrules\string;
11766 \glxtrcombiningdiacriticIVrules
11767 }

```

diacriticIrules First set of combining diacritic marks.

```

11768 \newcommand*{\glxtrcombiningdiacriticIrules}{%
11769 \glshex 0301\string;% combining acute
11770 \glshex 0300\string;% combining grave
11771 \glshex 0306\string;% combining breve
11772 \glshex 0302\string;% combining circumflex
11773 \glshex 030C\string;% combining caron
11774 \glshex 030A\string;% combining ring
11775 \glshex 030D\string;% combining vertical line above
11776 \glshex 0308\string;% combining diaeresis
11777 \glshex 030B\string;% combining double acute
11778 \glshex 0303\string;% combining tilde
11779 \glshex 0307\string;% combining dot above
11780 \glshex 0304% combining macron
11781 }

```

iacriticIIrules Second set of combining diacritic marks.

```

11782 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
11783 \glshex 0337\string;% combining short solidus overlay
11784 \glshex 0327\string;% combining cedilla
11785 \glshex 0328\string;% combining ogonek
11786 \glshex 0323\string;% combining dot below

```

```

11787 \glshex 0332\string;% combining low line
11788 \glshex 0305\string;% combining overline
11789 \glshex 0309\string;% combining hook above
11790 \glshex 030E\string;% combining double vertical line above
11791 \glshex 030F\string;% combining double grave accent
11792 \glshex 0310\string;% combining candrabindu
11793 \glshex 0311\string;% combining inverted breve
11794 \glshex 0312\string;% combining turned comma above
11795 \glshex 0313\string;% combining comma above
11796 \glshex 0314\string;% combining reversed comma above
11797 \glshex 0315\string;% combining comma above right
11798 \glshex 0316\string;% combining grave accent below
11799 \glshex 0317% combining acute accent below
11800 }

```

acriticIIIrules Third set of combining diacritic marks.

```

11801 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
11802 \glshex 0318\string;% combining left tack below
11803 \glshex 0319\string;% combining right tack below
11804 \glshex 031A\string;% combining left angle above
11805 \glshex 031B\string;% combining horn
11806 \glshex 031C\string;% combining left half ring below
11807 \glshex 031D\string;% combining up tack below
11808 \glshex 031E\string;% combining down tack below
11809 \glshex 031F\string;% combining plus sign below
11810 \glshex 0320\string;% combining minus sign below
11811 \glshex 0321\string;% combining palatalized hook below
11812 \glshex 0322\string;% combining retroflex hook below
11813 \glshex 0324\string;% combining diaeresis below
11814 \glshex 0325\string;% combining ring below
11815 \glshex 0326\string;% combining comma below
11816 \glshex 0329\string;% combining vertical line below
11817 \glshex 032A\string;% combining bridge below
11818 \glshex 032B\string;% combining inverted double arch below
11819 \glshex 032C\string;% combining caron below
11820 \glshex 032D\string;% combining circumflex accent below
11821 \glshex 032E\string;% combining breve below
11822 \glshex 032F\string;% combining inverted breve below
11823 \glshex 0330\string;% combining tilde below
11824 \glshex 0331\string;% combining macron below
11825 \glshex 0333\string;% combining double low line
11826 \glshex 0334\string;% combining tilde overlay
11827 \glshex 0335\string;% combining short stroke overlay
11828 \glshex 0336\string;% combining long stroke overlay
11829 \glshex 0338\string;% combining long solidus overlay
11830 \glshex 0339\string;% combining combining right half ring below
11831 \glshex 033A\string;% combining inverted bridge below
11832 \glshex 033B\string;% combining square below
11833 \glshex 033C\string;% combining seagull below

```

```

11834 \glshex 033D\string;% combining x above
11835 \glshex 033E\string;% combining vertical tilde
11836 \glshex 033F\string;% combining double overline
11837 \glshex 0342\string;% combining Greek perispomeni
11838 \glshex 0344\string;% combining Greek dialytika tonos
11839 \glshex 0345\string;% combining Greek ypogegrammeni
11840 \glshex 0360\string;% combining double tilde
11841 \glshex 0361\string;% combining double inverted breve
11842 \glshex 0483\string;% combining Cyrillic titlo
11843 \glshex 0484\string;% combining Cyrillic palatalization
11844 \glshex 0485\string;% combining Cyrillic dasia pneumata
11845 \glshex 0486% combining Cyrillic psili pneumata
11846 }

```

iacriticIVrules Fourth set of combining diacritic marks.

```

11847 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
11848 \glshex 20D0\string;% combining left harpoon above
11849 \glshex 20D1\string;% combining right harpoon above
11850 \glshex 20D2\string;% combining long vertical line overlay
11851 \glshex 20D3\string;% combining short vertical line overlay
11852 \glshex 20D4\string;% combining anticlockwise arrow above
11853 \glshex 20D5\string;% combining clockwise arrow above
11854 \glshex 20D6\string;% combining left arrow above
11855 \glshex 20D7\string;% combining right arrow above
11856 \glshex 20D8\string;% combining ring overlay
11857 \glshex 20D9\string;% combining clockwise ring overlay
11858 \glshex 20DA\string;% combining anticlockwise ring overlay
11859 \glshex 20DB\string;% combining three dots above
11860 \glshex 20DC\string;% combining four dots above
11861 \glshex 20DD\string;% combining enclosing circle
11862 \glshex 20DE\string;% combining enclosing square
11863 \glshex 20DF\string;% combining enclosing diamond
11864 \glshex 20E0\string;% combining enclosing circle backslash
11865 \glshex 20E1% combining left right arrow above
11866 }

```

sxtrhyphenrules Hyphens.

```

11867 \newcommand*{\glxtrhyphenrules}{%
11868 \string'\string-\string'\string;% ASCII hyphen
11869 \glshex 00AD\string;% soft hyphen
11870 \glshex 2010\string;% hyphen
11871 \glshex 2011\string;% non-breaking hyphen
11872 \glshex 2012\string;% figure dash
11873 \glshex 2013\string;% en dash
11874 \glshex 2014\string;% em dash
11875 \glshex 2015\string;% horizontal bar
11876 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11877 }

```

generalpuncrules General punctuation.

```
11878 \newcommand*{\glxtrgeneralpuncrules}{%
11879   \glxtrgeneralpuncIrules
11880   \string<\glxtrcurrencyrules
11881   \string<\glxtrgeneralpuncIIrules
11882 }
```

generalpuncIrules First set of general punctuation.

```
11883 \newcommand*{\glxtrgeneralpuncIrules}{%
11884   \string'\glshex 005F\string'% underscore
11885   \string<\glshex 00AF\string'% macron
11886   \string<\string'\glshex 002C\string'% comma
11887   \string<\string'\glshex 003B\string'% semi-colon
11888   \string<\string'\glshex 003A\string'% colon
11889   \string<\string'\glshex 0021\string'% exclamation mark
11890   \string<\glshex 00A1\string'% inverted exclamation mark
11891   \string<\string'\glshex 003F\string'% question mark
11892   \string<\glshex 00BF\string'% inverted question mark
11893   \string<\string'\glshex 002F\string'% solidus
11894   \string<\string'\glshex 002E\string'% full stop
11895   \string<\glshex 00B4\string'% acute accent
11896   \string<\string'\glshex 0060\string'% grave accent
11897   \string<\string'\glshex 005E\string'% circumflex accent
11898   \string<\glshex 00A8\string'% diaeresis
11899   \string<\string'\glshex 007E\string'% tilde
11900   \string<\glshex 00B7\string'% middle dot
11901   \string<\glshex 00B8\string'% cedilla
11902   \string<\string'\glshex 0027\string'% straight apostrophe
11903   \string<\string'\glshex 0022\string'% straight double quote
11904   \string<\glshex 00AB\string'% left guillemet
11905   \string<\glshex 00BB\string'% right guillemet
11906   \string<\string'\glshex 0028\string'% left parenthesis
11907   \string=<\glshex 207D\string=<\glshex 208D\string'% super/subscript left parenthesis
11908   \string<\string'\glshex 0029\string'% right parenthesis
11909   \string=<\glshex 207E\string=<\glshex 208E\string'% super/subscript right parenthesis
11910   \string<\string'\glshex 005B\string'% left square bracket
11911   \string<\string'\glshex 005D\string'% right square bracket
11912   \string<\string'\glshex 007B\string'% left curly bracket
11913   \string<\string'\glshex 007D\string'% right curly bracket
11914   \string<\glshex 00A7\string'% section sign
11915   \string<\glshex 00B6\string'% pilcrow sign
11916   \string<\glshex 00A9\string'% copyright sign
11917   \string<\glshex 00AE\string'% registered sign
11918   \string<\string'\glshex 0040\string'% at sign
11919 }
```

trcurrencyrules General punctuation.

```
11920 \newcommand*{\glxtrcurrencyrules}{%
11921   \glshex 00A4\string'% currency sign
```



```

11922 \string<\glshex 0E3F% Thai currency symbol baht
11923 \string<\glshex 00A2% cent sign
11924 \string<\glshex 20A1% colon sign
11925 \string<\glshex 20A2% cruzeiro sign
11926 \string<\string'\glshex 0024\string'% dollar sign
11927 \string<\glshex 20AB% dong sign
11928 \string<\glshex 20AC% euro sign
11929 \string<\glshex 20A3% French franc sign
11930 \string<\glshex 20A4% lira sign
11931 \string<\glshex 20A5% mill sign
11932 \string<\glshex 20A6% naira sign
11933 \string<\glshex 20A7% peseta sign
11934 \string<\glshex 00A3% pound sign
11935 \string<\glshex 20A8% rupee sign
11936 \string<\glshex 20AA% new sheqel sign
11937 \string<\glshex 20A9% won sign
11938 \string<\glshex 00A5% yen sign
11939 }

```

eralpuncIIrules Second set of general punctuation.

```

11940 \newcommand*{\glxtrgeneralpuncIIrules}{%
11941 \string'\glshex 002A\string'% asterisk
11942 \string<\string'\glshex 005C\string'% backslash
11943 \string<\string'\glshex 0026\string'% ampersand
11944 \string<\string'\glshex 0023\string'% hash sign
11945 \string<\string'\glshex 0025\string'% percent sign
11946 \string<\string'\glshex 002B\string'% plus sign
11947 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
11948 \string<\glshex 00B1% plus-minus sign
11949 \string<\glshex 00F7% division sign
11950 \string<\glshex 00D7% multiplication sign
11951 \string<\string'\glshex 003C\string'% less-than sign
11952 \string<\string'\glshex 003D\string'% equals sign
11953 \string<\string'\glshex 003E\string'% greater-than sign
11954 \string<\glshex 00AC% not sign
11955 \string<\string'\glshex 007C\string'% vertical bar (pipe)
11956 \string<\glshex 00A6% broken bar
11957 \string<\glshex 00B0% degree sign
11958 \string<\glshex 00B5% micron sign
11959 }

```

eralLatinIrules Basic Latin alphabet.

```

11960 \newcommand*{\glxtrGeneralLatinIrules}{%
11961 \glxtrLatinA
11962 \string<b,B%
11963 \string<c,C%
11964 \string<d,D%
11965 \string<\glxtrLatinE
11966 \string<f,F%

```

```

11967 \string<g,G%
11968 \string<\glxtrLatinH
11969 \string<\glxtrLatinI
11970 \string<j,J%
11971 \string<\glxtrLatinK
11972 \string<\glxtrLatinL
11973 \string<\glxtrLatinM
11974 \string<\glxtrLatinN
11975 \string<\glxtrLatinO
11976 \string<\glxtrLatinP
11977 \string<q,Q%
11978 \string<r,R%
11979 \string<\glxtrLatinS
11980 \string<\glxtrLatinT
11981 \string<u,U%
11982 \string<v,V%
11983 \string<w,W%
11984 \string<\glxtrLatinX
11985 \string<y,Y%
11986 \string<z,Z
11987 }

```

raalLatinIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

11988 \newcommand*{\glxtrGeneralLatinIrules}{%
11989 \glxtrLatinA
11990 \string<b,B%
11991 \string<c,C%
11992 \string<d,D%
11993 \string<\glxtrLatinEth
11994 \string<\glxtrLatinE
11995 \string<f,F%
11996 \string<g,G%
11997 \string<\glxtrLatinH
11998 \string<\glxtrLatinI
11999 \string<j,J%
12000 \string<\glxtrLatinK
12001 \string<\glxtrLatinL
12002 \string<\glxtrLatinM
12003 \string<\glxtrLatinN
12004 \string<\glxtrLatinO
12005 \string<\glxtrLatinP
12006 \string<q,Q%
12007 \string<r,R%
12008 \string<\glxtrLatinS
12009 \string& SS \string, \glxtrLatinEszettSs
12010 \string<\glxtrLatinT
12011 \string<u,U%
12012 \string<v,V%
12013 \string<w,W%

```

```

12014 \string<\glxtrLatinX
12015 \string<y,Y%
12016 \string<z,Z%
12017 }

```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12018 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12019 \glxtrLatinA
12020 \string<b,B%
12021 \string<c,C%
12022 \string<d,D%
12023 \string<\glxtrLatinEth
12024 \string<\glxtrLatinE
12025 \string<f,F%
12026 \string<g,G%
12027 \string<\glxtrLatinH
12028 \string<\glxtrLatinI
12029 \string<j,J%
12030 \string<\glxtrLatinK
12031 \string<\glxtrLatinL
12032 \string<\glxtrLatinM
12033 \string<\glxtrLatinN
12034 \string<\glxtrLatinO
12035 \string<\glxtrLatinP
12036 \string<q,Q%
12037 \string<r,R%
12038 \string<\glxtrLatinS
12039 \string& SZ, \glxtrLatinEszettSz
12040 \string<\glxtrLatinT
12041 \string<u,U%
12042 \string<v,V%
12043 \string<w,W%
12044 \string<\glxtrLatinX
12045 \string<y,Y%
12046 \string<z,Z%
12047 }

```

alLatinIVrules General Latin alphabet (Æ treated as AE and Æ treated as OE, Þ treated as TH, ß treated as SS, eth between D and E).

```

12048 \newcommand*{\glxtrGeneralLatinIVrules}{%
12049 \glxtrLatinA
12050 \string& AE , \glxtrLatinAELigature
12051 \string<b,B%
12052 \string<c,C%
12053 \string<d,D%
12054 \string<\glxtrLatinEth
12055 \string<\glxtrLatinE
12056 \string<f,F%
12057 \string<g,G%

```

```

12058 \string<\glxtrLatinH
12059 \string<\glxtrLatinI
12060 \string<j,J%
12061 \string<\glxtrLatinK
12062 \string<\glxtrLatinL
12063 \string<\glxtrLatinM
12064 \string<\glxtrLatinN
12065 \string<\glxtrLatinO
12066 \string& OE , \glxtrLatinOELigature
12067 \string<\glxtrLatinP
12068 \string<q,Q%
12069 \string<r,R%
12070 \string<\glxtrLatinS
12071 \string& SS , \glxtrLatinEszettSs
12072 \string<\glxtrLatinT
12073 \string& th =\glshex 00DE
12074 \string& TH =\glshex 00FE
12075 \string<u,U%
12076 \string<v,V%
12077 \string<w,W%
12078 \string<\glxtrLatinX
12079 \string<y,Y%
12080 \string<z,Z%
12081 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12082 \newcommand*{\glxtrGeneralLatinVrules}{%
12083 \glxtrLatinA
12084 \string<b,B%
12085 \string<c,C%
12086 \string<d,D%
12087 \string<\glxtrLatinEth
12088 \string<\glxtrLatinE
12089 \string<f,F%
12090 \string<g,G%
12091 \string<\glxtrLatinH
12092 \string<\glxtrLatinI
12093 \string<j,J%
12094 \string<\glxtrLatinK
12095 \string<\glxtrLatinL
12096 \string<\glxtrLatinM
12097 \string<\glxtrLatinN
12098 \string<\glxtrLatinO
12099 \string<\glxtrLatinP
12100 \string<q,Q%
12101 \string<r,R%
12102 \string<\glxtrLatinS
12103 \string& SS , \glxtrLatinEszettSs
12104 \string<\glxtrLatinT

```

```

12105 \string& th =\glshex 00DE
12106 \string& TH =\glshex 00FE
12107 \string<u,U%
12108 \string<v,V%
12109 \string<w,W%
12110 \string<\glsxtrLatinX
12111 \string<y,Y%
12112 \string<z,Z%
12113 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12114 \newcommand*{\glsxtrGeneralLatinVIrules}{%
12115 \glsxtrLatinA
12116 \string<b,B%
12117 \string<c,C%
12118 \string<d,D%
12119 \string<\glsxtrLatinEth
12120 \string<\glsxtrLatinE
12121 \string<f,F%
12122 \string<g,G%
12123 \string<\glsxtrLatinH
12124 \string<\glsxtrLatinI
12125 \string<j,J%
12126 \string<\glsxtrLatinK
12127 \string<\glsxtrLatinL
12128 \string<\glsxtrLatinM
12129 \string<\glsxtrLatinN
12130 \string<\glsxtrLatinO
12131 \string<\glsxtrLatinP
12132 \string<q,Q%
12133 \string<r,R%
12134 \string<\glsxtrLatinS
12135 \string& SZ , \glsxtrLatinEszettSz
12136 \string<\glsxtrLatinT
12137 \string& th =\glshex 00DE
12138 \string& TH =\glshex 00FE
12139 \string<u,U%
12140 \string<v,V%
12141 \string<w,W%
12142 \string<\glsxtrLatinX
12143 \string<y,Y%
12144 \string<z,Z%
12145 }

```

alLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, CE between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

12146 \newcommand*{\glsxtrGeneralLatinVIIrules}{%
12147 \glsxtrLatinA
12148 \string<\glsxtrLatinAELigature

```

```

12149 \string<b,B%
12150 \string<c,C%
12151 \string<d,D%
12152 \string<\glxtrLatinEth
12153 \string<\glxtrLatinE
12154 \string<f,F%
12155 \string<\glxtrLatinInsularG
12156 \string<\glxtrLatinH
12157 \string<\glxtrLatinI
12158 \string<j,J%
12159 \string<\glxtrLatinK
12160 \string<\glxtrLatinL
12161 \string<\glxtrLatinM
12162 \string<\glxtrLatinN
12163 \string<\glxtrLatinO
12164 \string<\glxtrLatinOELigature
12165 \string<\glxtrLatinP
12166 \string<q,Q%
12167 \string<r,R%
12168 \string<\glshex 017F=\glxtrLatinS % s and long s
12169 \string<\glxtrLatinT
12170 \string<\glxtrLatinThorn
12171 \string<u,U%
12172 \string<v,V%
12173 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12174 \string<\glxtrLatinX
12175 \string<y,Y%
12176 \string<z,Z%
12177 }

```

1LatinVIIIrules General Latin alphabet (Æ treated as AE and Ć treated as OE, Þ treated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

12178 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
12179 \glxtrLatinA
12180 \string& AE , \glxtrLatinAELigature
12181 \string<b,B%
12182 \string<c,C%
12183 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12184 \string<\glxtrLatinE
12185 \string<f,F%
12186 \string<g,G%
12187 \string<\glxtrLatinH
12188 \string<\glxtrLatinI
12189 \string<j,J%
12190 \string<\glxtrLatinK
12191 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12192 \string<\glxtrLatinM
12193 \string<\glxtrLatinN
12194 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O

```

```

12195 \string& OE , \glxtrLatinOELigature
12196 \string<\glxtrLatinP
12197 \string<q,Q%
12198 \string<r,R%
12199 \string<\glxtrLatinS
12200 \string& SS , \glxtrLatinEszettSs
12201 \string<\glxtrLatinT
12202 \string& th =\glshex 00DE
12203 \string& TH =\glshex 00FE
12204 \string<u,U%
12205 \string<v,V%
12206 \string<w,W%
12207 \string<\glxtrLatinX
12208 \string<y,Y%
12209 \string<z,Z%
12210 }

```

\glxtrLatinA

```

12211 \newcommand*{\glxtrLatinA}{%
12212   a\string=\glshex 00AA\string=\glshex 2090,A
12213 }

```

\glxtrLatinE

```

12214 \newcommand*{\glxtrLatinE}{%
12215   e\string=\glshex 2091,E
12216 }

```

\glxtrLatinH

```

12217 \newcommand*{\glxtrLatinH}{%
12218   h\string=\glshex 2095,H
12219 }

```

\glxtrLatinI

```

12220 \newcommand*{\glxtrLatinI}{%
12221   i\string=\glshex 2071,I
12222 }

```

\glxtrLatinK

```

12223 \newcommand*{\glxtrLatinK}{%
12224   k\string=\glshex 2096,K
12225 }

```

\glxtrLatinL

```

12226 \newcommand*{\glxtrLatinL}{%
12227   l\string=\glshex 2097,L
12228 }

```

\glsxtrLatinM

```
12229 \newcommand*{\glsxtrLatinM}{%
12230   m\string=\glshex 2098,M
12231 }
```

\glsxtrLatinN

```
12232 \newcommand*{\glsxtrLatinN}{%
12233   n\string=\glshex 207F\string=\glshex 2099,N
12234 }
```

\glsxtrLatinO

```
12235 \newcommand*{\glsxtrLatinO}{%
12236   o\string=\glshex 00BA\string=\glshex 2092,O
12237 }
```

\glsxtrLatinP

```
12238 \newcommand*{\glsxtrLatinP}{%
12239   p\string=\glshex 209A,P
12240 }
```

\glsxtrLatinS

```
12241 \newcommand*{\glsxtrLatinS}{%
12242   s\string=\glshex 209B,S
12243 }
```

\glsxtrLatinT

```
12244 \newcommand*{\glsxtrLatinT}{%
12245   t\string=\glshex 209C,T
12246 }
```

\glsxtrLatinX

```
12247 \newcommand*{\glsxtrLatinX}{%
12248   x\string=\glshex 2093,X
12249 }
```

\glsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).

```
12250 \newcommand*{\glsxtrLatinSchwa}{%
12251   \glshex 0259\string=\glshex 2094,\glshex 018F
12252 }
```

\glsxtrLatinEszettSs

```
12253 \newcommand*{\glsxtrLatinEszettSs}{%
12254   \glshex 00DF% eszett
12255   \string=\glshex 017Fs % long S s
12256 }
```


trLatinEszettSz

```
12257 \newcommand*{\glxtrLatinEszettSz}{%
12258   \glshex 00DF% eszett
12259   \string= \glshex 017Fz % long S z
12260 }
```

\glxtrLatinEth

```
12261 \newcommand*{\glxtrLatinEth}{%
12262   \glshex 00F0,\glshex 00D0% eth
12263 }
```

lsxtrLatinThorn

```
12264 \newcommand*{\glxtrLatinThorn}{%
12265   \glshex 00FE,\glshex 00DE% thorn
12266 }
```

LatinAELigature

```
12267 \newcommand*{\glxtrLatinAELigature}{%
12268   \glshex 00E6,\glshex 00C6% AE-ligature
12269 }
```

LatinOELigature

```
12270 \newcommand*{\glxtrLatinOELigature}{%
12271   \glshex 0153,\glshex 0152% OE-ligature
12272 }
```

\glxtrLatinAA

```
12273 \newcommand*{\glxtrLatinAA}{%
12274   \glshex 00E5=a\glshex 030A,% \aa
12275   \glshex 00C5=A\glshex 030A% \AA
12276 }
```

glxtrLatinWynn

```
12277 \newcommand*{\glxtrLatinWynn}{%
12278   \glshex 01BF,\glshex 01F7% wynn
12279 }
```

trLatinInsularG

```
12280 \newcommand*{\glxtrLatinInsularG}{%
12281   \glshex 1D79,\glshex A77D% insular G
12282   \string; g, G
12283 }
```

sxtrLatinOslash

```
12284 \newcommand*{\glxtrLatinOslash}{%
12285   \glshex 00F8,\glshex 00D8% \o, \O
12286 }
```

sxtrLatinLslash

```
12287 \newcommand*{\glsxtrLatinLslash}{%
12288   \glshex 0142,\glshex 0141% \l, \L
12289 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12290 \newcommand*{\glsxtrMathUpGreekIrules}{%
12291   \glsxtrUpAlpha
12292   \string<\glsxtrUpBeta
12293   \string<\glsxtrUpGamma
12294   \string<\glsxtrUpDelta
12295   \string<\glsxtrUpEpsilon
12296   \string<\glsxtrUpDigamma
12297   \string<\glsxtrUpZeta
12298   \string<\glsxtrUpEta
12299   \string<\glsxtrUpTheta
12300   \string<\glsxtrUpIota
12301   \string<\glsxtrUpKappa
12302   \string<\glsxtrUpLambda
12303   \string<\glsxtrUpMu
12304   \string<\glsxtrUpNu
12305   \string<\glsxtrUpXi
12306   \string<\glsxtrUpOmicron
12307   \string<\glsxtrUpPi
12308   \string<\glsxtrUpRho
12309   \string<\glsxtrUpSigma
12310   \string<\glsxtrUpTau
12311   \string<\glsxtrUpUpsilon
12312   \string<\glsxtrUpPhi
12313   \string<\glsxtrUpChi
12314   \string<\glsxtrUpPsi
12315   \string<\glsxtrUpOmega
12316 }
```

hUpGreekIIrules Doesn't include digamma.

```
12317 \newcommand*{\glsxtrMathUpGreekIIrules}{%
12318   \glsxtrUpAlpha
12319   \string<\glsxtrUpBeta
12320   \string<\glsxtrUpGamma
12321   \string<\glsxtrUpDelta
12322   \string<\glsxtrUpEpsilon
12323   \string<\glsxtrUpZeta
12324   \string<\glsxtrUpEta
12325   \string<\glsxtrUpTheta
12326   \string<\glsxtrUpIota
12327   \string<\glsxtrUpKappa
12328   \string<\glsxtrUpLambda
12329   \string<\glsxtrUpMu
12330   \string<\glsxtrUpNu
```

```

12331 \string<\glxtrUpXi
12332 \string<\glxtrUpOmicron
12333 \string<\glxtrUpPi
12334 \string<\glxtrUpRho
12335 \string<\glxtrUpSigma
12336 \string<\glxtrUpTau
12337 \string<\glxtrUpUpsilon
12338 \string<\glxtrUpPhi
12339 \string<\glxtrUpChi
12340 \string<\glxtrUpPsi
12341 \string<\glxtrUpOmega
12342 }

```

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.

```

12343 \newcommand*{\glxtrMathItalicGreekIrules}{%
12344 \glxtrMathItalicAlpha
12345 \string<\glxtrMathItalicBeta
12346 \string<\glxtrMathItalicGamma
12347 \string<\glxtrMathItalicDelta
12348 \string<\glxtrMathItalicEpsilon
12349 \string<\glxtrUpDigamma
12350 \string<\glxtrMathItalicZeta
12351 \string<\glxtrMathItalicEta
12352 \string<\glxtrMathItalicTheta
12353 \string<\glxtrMathItalicIota
12354 \string<\glxtrMathItalicKappa
12355 \string<\glxtrMathItalicLambda
12356 \string<\glxtrMathItalicMu
12357 \string<\glxtrMathItalicNu
12358 \string<\glxtrMathItalicXi
12359 \string<\glxtrMathItalicOmicron
12360 \string<\glxtrMathItalicPi
12361 \string<\glxtrMathItalicRho
12362 \string<\glxtrMathItalicSigma
12363 \string<\glxtrMathItalicTau
12364 \string<\glxtrMathItalicUpsilon
12365 \string<\glxtrMathItalicPhi
12366 \string<\glxtrMathItalicChi
12367 \string<\glxtrMathItalicPsi
12368 \string<\glxtrMathItalicOmega
12369 }

```

alicGreekIIrules Doesn't include digamma.

```

12370 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12371 \glxtrMathItalicAlpha
12372 \string<\glxtrMathItalicBeta
12373 \string<\glxtrMathItalicGamma
12374 \string<\glxtrMathItalicDelta

```

```

12375 \string<\glxtrMathItalicEpsilon
12376 \string<\glxtrMathItalicZeta
12377 \string<\glxtrMathItalicEta
12378 \string<\glxtrMathItalicTheta
12379 \string<\glxtrMathItalicIota
12380 \string<\glxtrMathItalicKappa
12381 \string<\glxtrMathItalicLambda
12382 \string<\glxtrMathItalicMu
12383 \string<\glxtrMathItalicNu
12384 \string<\glxtrMathItalicXi
12385 \string<\glxtrMathItalicOmicron
12386 \string<\glxtrMathItalicPi
12387 \string<\glxtrMathItalicRho
12388 \string<\glxtrMathItalicSigma
12389 \string<\glxtrMathItalicTau
12390 \string<\glxtrMathItalicUpsilon
12391 \string<\glxtrMathItalicPhi
12392 \string<\glxtrMathItalicChi
12393 \string<\glxtrMathItalicPsi
12394 \string<\glxtrMathItalicOmega
12395 }

```

UpperGreekrules Upper case only (includes upright digamma).

```

12396 \newcommand*{\glxtrMathItalicUpperGreekrules}{%
12397 \glshex 1D6E2% upper case alpha (maths italic)
12398 \string<\glshex 1D6E3% upper case beta (maths italic)
12399 \string<\glshex 1D6E4% upper case gamma (maths italic)
12400 \string<\glshex 1D6E5% upper case delta (maths italic)
12401 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12402 \string<\glshex 03DC% upper case digamma
12403 \string<\glshex 1D6E7% upper case zeta (maths italic)
12404 \string<\glshex 1D6E8% upper case eta (maths italic)
12405 \string<\glshex 1D6E9% upper case theta (maths italic)
12406 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12407 \string<\glshex 1D6EA% upper case iota (maths italic)
12408 \string<\glshex 1D6EB% upper case kappa (maths italic)
12409 \string<\glshex 1D6EC% upper case lambda (maths italic)
12410 \string<\glshex 1D6ED% upper case mu (maths italic)
12411 \string<\glshex 1D6EE% upper case nu (maths italic)
12412 \string<\glshex 1D6EF% upper case xi (maths italic)
12413 \string<\glshex 1D6F0% upper case omicron (maths italic)
12414 \string<\glshex 1D6F1% upper case pi (maths italic)
12415 \string<\glshex 1D6F2% upper case rho (maths italic)
12416 \string<\glshex 1D6F4% upper case sigma (maths italic)
12417 \string<\glshex 1D6F5% upper case tau (maths italic)
12418 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12419 \string<\glshex 1D6F7% upper case phi (maths italic)
12420 \string<\glshex 1D6F8% upper case chi (maths italic)
12421 \string<\glshex 1D6F9% upper case psi (maths italic)

```

```

12422 \string<\glshex 1D6FA% upper case omega (maths italic)
12423 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

12424 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
12425 \glshex 1D6E2% upper case alpha (maths italic)
12426 \string<\glshex 1D6E3% upper case beta (maths italic)
12427 \string<\glshex 1D6E4% upper case gamma (maths italic)
12428 \string<\glshex 1D6E5% upper case delta (maths italic)
12429 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12430 \string<\glshex 1D6E7% upper case zeta (maths italic)
12431 \string<\glshex 1D6E8% upper case eta (maths italic)
12432 \string<\glshex 1D6E9% upper case theta (maths italic)
12433 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12434 \string<\glshex 1D6EA% upper case iota (maths italic)
12435 \string<\glshex 1D6EB% upper case kappa (maths italic)
12436 \string<\glshex 1D6EC% upper case lambda (maths italic)
12437 \string<\glshex 1D6ED% upper case mu (maths italic)
12438 \string<\glshex 1D6EE% upper case nu (maths italic)
12439 \string<\glshex 1D6EF% upper case xi (maths italic)
12440 \string<\glshex 1D6F0% upper case omicron (maths italic)
12441 \string<\glshex 1D6F1% upper case pi (maths italic)
12442 \string<\glshex 1D6F2% upper case rho (maths italic)
12443 \string<\glshex 1D6F4% upper case sigma (maths italic)
12444 \string<\glshex 1D6F5% upper case tau (maths italic)
12445 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12446 \string<\glshex 1D6F7% upper case phi (maths italic)
12447 \string<\glshex 1D6F8% upper case chi (maths italic)
12448 \string<\glshex 1D6F9% upper case psi (maths italic)
12449 \string<\glshex 1D6FA% upper case omega (maths italic)
12450 }

```

owerGreekIrules Lower case only (includes upright digamma).

```

12451 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%
12452 \glshex 1D6FC% lower case alpha (maths italic)
12453 \string<\glshex 1D6FD% lower case beta (maths italic)
12454 \string<\glshex 1D6FE% lower case gamma (maths italic)
12455 \string<\glshex 1D6FF% lower case delta (maths italic)
12456 \string<\glshex 1D700% lower case epsilon (maths italic)
12457 \string<\glshex 1D716% lower case epsilon variant (maths italic)
12458 \string<\glshex 03DD% lower case digamma
12459 \string<\glshex 1D701% lower case zeta (maths italic)
12460 \string<\glshex 1D702% lower case eta (maths italic)
12461 \string<\glshex 1D703% lower case theta (maths italic)
12462 \string<\glshex 1D717% lower case theta variant (maths italic)
12463 \string<\glshex 1D704% lower case iota (maths italic)
12464 \string<\glshex 1D705% lower case kappa (maths italic)
12465 \string<\glshex 1D718% lower case kappa variant (maths italic)
12466 \string<\glshex 1D706% lower case lambda (maths italic)

```

```

12467 \string<\glshex 1D707% lower case mu (maths italic)
12468 \string<\glshex 1D708% lower case nu (maths italic)
12469 \string<\glshex 1D709% lower case xi (maths italic)
12470 \string<\glshex 1D70A% lower case omicron (maths italic)
12471 \string<\glshex 1D70B% lower case pi (maths italic)
12472 \string=\glshex 1D71B% lower case pi variant (maths italic)
12473 \string<\glshex 1D70C% lower case rho (maths italic)
12474 \string=\glshex 1D71A% lower case rho variant (maths italic)
12475 \string<\glshex 1D70D% lower case final sigma (maths italic)
12476 \string=\glshex 1D70E% lower case sigma (maths italic)
12477 \string<\glshex 1D70F% lower case tau (maths italic)
12478 \string<\glshex 1D710% lower case upsilon (maths italic)
12479 \string<\glshex 1D711% lower case phi (maths italic)
12480 \string=\glshex 1D719% lower case phi variant (maths italic)
12481 \string<\glshex 1D712% lower case chi (maths italic)
12482 \string<\glshex 1D713% lower case psi (maths italic)
12483 \string<\glshex 1D714% lower case omega (maths italic)
12484 }

```

Lower case only (doesn't includes upright digamma).

```

12485 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
12486 \glshex 1D6FC% lower case alpha (maths italic)
12487 \string<\glshex 1D6FD% lower case beta (maths italic)
12488 \string<\glshex 1D6FE% lower case gamma (maths italic)
12489 \string<\glshex 1D6FF% lower case delta (maths italic)
12490 \string<\glshex 1D700% lower case epsilon (maths italic)
12491 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12492 \string<\glshex 1D701% lower case zeta (maths italic)
12493 \string<\glshex 1D702% lower case eta (maths italic)
12494 \string<\glshex 1D703% lower case theta (maths italic)
12495 \string=\glshex 1D717% lower case theta variant (maths italic)
12496 \string<\glshex 1D704% lower case iota (maths italic)
12497 \string<\glshex 1D705% lower case kappa (maths italic)
12498 \string=\glshex 1D718% lower case kappa variant (maths italic)
12499 \string<\glshex 1D706% lower case lambda (maths italic)
12500 \string<\glshex 1D707% lower case mu (maths italic)
12501 \string<\glshex 1D708% lower case nu (maths italic)
12502 \string<\glshex 1D709% lower case xi (maths italic)
12503 \string<\glshex 1D70A% lower case omicron (maths italic)
12504 \string<\glshex 1D70B% lower case pi (maths italic)
12505 \string=\glshex 1D71B% lower case pi variant (maths italic)
12506 \string<\glshex 1D70C% lower case rho (maths italic)
12507 \string=\glshex 1D71A% lower case rho variant (maths italic)
12508 \string<\glshex 1D70D% lower case final sigma (maths italic)
12509 \string=\glshex 1D70E% lower case sigma (maths italic)
12510 \string<\glshex 1D70F% lower case tau (maths italic)
12511 \string<\glshex 1D710% lower case upsilon (maths italic)
12512 \string<\glshex 1D711% lower case phi (maths italic)
12513 \string=\glshex 1D719% lower case phi variant (maths italic)

```

```

12514 \string<\glshex 1D712% lower case chi (maths italic)
12515 \string<\glshex 1D713% lower case psi (maths italic)
12516 \string<\glshex 1D714% lower case omega (maths italic)
12517 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12518 \newcommand*{\glxtrMathGreekIrules}{%
12519 \glxtrMathItalicAlpha
12520 \string;\glxtrUpAlpha
12521 \string<\glxtrMathItalicBeta
12522 \string;\glxtrUpBeta
12523 \string<\glxtrMathItalicGamma
12524 \string;\glxtrUpGamma
12525 \string<\glxtrMathItalicDelta
12526 \string;\glxtrUpDelta
12527 \string<\glxtrMathItalicEpsilon
12528 \string;\glxtrUpEpsilon
12529 \string<\glxtrUpDigamma
12530 \string<\glxtrMathItalicZeta
12531 \string;\glxtrUpZeta
12532 \string<\glxtrMathItalicEta
12533 \string;\glxtrUpEta
12534 \string<\glxtrMathItalicTheta
12535 \string;\glxtrUpTheta
12536 \string<\glxtrMathItalicIota
12537 \string;\glxtrUpIota
12538 \string<\glxtrMathItalicKappa
12539 \string;\glxtrUpKappa
12540 \string<\glxtrMathItalicLambda
12541 \string;\glxtrUpLambda
12542 \string<\glxtrMathItalicMu
12543 \string;\glxtrUpMu
12544 \string<\glxtrMathItalicNu
12545 \string;\glxtrUpNu
12546 \string<\glxtrMathItalicXi
12547 \string;\glxtrUpXi
12548 \string<\glxtrMathItalicOmicron
12549 \string;\glxtrUpOmicron
12550 \string<\glxtrMathItalicPi
12551 \string;\glxtrUpPi
12552 \string<\glxtrMathItalicRho
12553 \string;\glxtrUpRho
12554 \string<\glxtrMathItalicSigma
12555 \string;\glxtrUpSigma
12556 \string<\glxtrMathItalicTau
12557 \string;\glxtrUpTau
12558 \string<\glxtrMathItalicUpsilon
12559 \string;\glxtrUpUpsilon
12560 \string<\glxtrMathItalicPhi

```

```

12561 \string;\glxtrUpPhi
12562 \string<\glxtrMathItalicChi
12563 \string;\glxtrUpChi
12564 \string<\glxtrMathItalicPsi
12565 \string;\glxtrUpPsi
12566 \string<\glxtrMathItalicOmega
12567 \string;\glxtrUpOmega
12568 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

12569 \newcommand*{\glxtrMathGreekIIrules}{%
12570 \glxtrMathItalicAlpha
12571 \string;\glxtrUpAlpha
12572 \string<\glxtrMathItalicBeta
12573 \string;\glxtrUpBeta
12574 \string<\glxtrMathItalicGamma
12575 \string;\glxtrUpGamma
12576 \string<\glxtrMathItalicDelta
12577 \string;\glxtrUpDelta
12578 \string<\glxtrMathItalicEpsilon
12579 \string;\glxtrUpEpsilon
12580 \string<\glxtrMathItalicZeta
12581 \string;\glxtrUpZeta
12582 \string<\glxtrMathItalicEta
12583 \string;\glxtrUpEta
12584 \string<\glxtrMathItalicTheta
12585 \string;\glxtrUpTheta
12586 \string<\glxtrMathItalicIota
12587 \string;\glxtrUpIota
12588 \string<\glxtrMathItalicKappa
12589 \string;\glxtrUpKappa
12590 \string<\glxtrMathItalicLambda
12591 \string;\glxtrUpLambda
12592 \string<\glxtrMathItalicMu
12593 \string;\glxtrUpMu
12594 \string<\glxtrMathItalicNu
12595 \string;\glxtrUpNu
12596 \string<\glxtrMathItalicXi
12597 \string;\glxtrUpXi
12598 \string<\glxtrMathItalicOmicron
12599 \string;\glxtrUpOmicron
12600 \string<\glxtrMathItalicPi
12601 \string;\glxtrUpPi
12602 \string<\glxtrMathItalicRho
12603 \string;\glxtrUpRho
12604 \string<\glxtrMathItalicSigma
12605 \string;\glxtrUpSigma
12606 \string<\glxtrMathItalicTau
12607 \string;\glxtrUpTau

```



```

12608 \string<\glxtrMathItalicUpsilon
12609 \string;\glxtrUpUpsilon
12610 \string<\glxtrMathItalicPhi
12611 \string;\glxtrUpPhi
12612 \string<\glxtrMathItalicChi
12613 \string;\glxtrUpChi
12614 \string<\glxtrMathItalicPsi
12615 \string;\glxtrUpPsi
12616 \string<\glxtrMathItalicOmega
12617 \string;\glxtrUpOmega
12618 }

```

\glxtrUpAlpha

```

12619 \newcommand*{\glxtrUpAlpha}{%
12620 \glshex 03B1,% lower case alpha
12621 \glshex 0391% upper case alpha
12622 }

```

\glxtrUpBeta

```

12623 \newcommand*{\glxtrUpBeta}{%
12624 \glshex 03B2,% lower case beta
12625 \glshex 0392% upper case beta
12626 }

```

\glxtrUpGamma

```

12627 \newcommand*{\glxtrUpGamma}{%
12628 \glshex 03B3,% lower case gamma
12629 \glshex 0393% upper case gamma
12630 }

```

\glxtrUpDelta

```

12631 \newcommand*{\glxtrUpDelta}{%
12632 \glshex 03B4,% lower case delta
12633 \glshex 0394% upper case delta
12634 }

```

glxtrUpEpsilon

```

12635 \newcommand*{\glxtrUpEpsilon}{%
12636 \glshex 03B5% lower case epsilon
12637 \string=\glshex 03F5,% lower case epsilon variant
12638 \glshex 0395% upper case epsilon
12639 }

```

glxtrUpDigamma

```

12640 \newcommand*{\glxtrUpDigamma}{%
12641 \glshex 03DD,% lower case digamma
12642 \glshex 03DC% upper case digamma
12643 }

```

\backslash glxtrUpZeta

```
12644 \newcommand*{\glxtrUpZeta}{%
12645   \glshex 03B6,% lower case zeta
12646   \glshex 0396% upper case zeta
12647 }
```

\backslash glxtrUpEta

```
12648 \newcommand*{\glxtrUpEta}{%
12649   \glshex 03B7,% lower case eta
12650   \glshex 0397% upper case eta
12651 }
```

\backslash glxtrUpTheta

```
12652 \newcommand*{\glxtrUpTheta}{%
12653   \glshex 03B8% lower case theta
12654   \string=\glshex 03D1,% lower case theta variant
12655   \glshex 0398% upper case theta
12656 }
```

\backslash glxtrUpIota

```
12657 \newcommand*{\glxtrUpIota}{%
12658   \glshex 03B9,% lower case iota
12659   \glshex 0399% upper case iota
12660 }
```

\backslash glxtrUpKappa

```
12661 \newcommand*{\glxtrUpKappa}{%
12662   \glshex 03BA% lower case kappa
12663   \string=\glshex 03F0,% lower case kappa variant
12664   \glshex 039A% upper case kappa
12665 }
```

\backslash glxtrUpLambda

```
12666 \newcommand*{\glxtrUpLambda}{%
12667   \glshex 03BB,% lower lambda
12668   \glshex 039B% upper case lambda
12669 }
```

\backslash glxtrUpMu

```
12670 \newcommand*{\glxtrUpMu}{%
12671   \glshex 03BC,% lower case mu
12672   \glshex 039C% upper case mu
12673 }
```

\backslash glxtrUpNu

```
12674 \newcommand*{\glxtrUpNu}{%
12675   \glshex 03BD,% lower case nu
12676   \glshex 039D% upper case nu
12677 }
```

\glsxtrUpXi

```
12678 \newcommand*{\glsxtrUpXi}{%  
12679 \glshex 03BE,% lower case xi  
12680 \glshex 039E% upper case xi  
12681 }
```

glsxtrUpOmicron

```
12682 \newcommand*{\glsxtrUpOmicron}{%  
12683 \glshex 03BF,% lower case omicron  
12684 \glshex 039F% upper case omicron  
12685 }
```

\glsxtrUpPi

```
12686 \newcommand*{\glsxtrUpPi}{%  
12687 \glshex 03C0% lower case pi  
12688 \string=\glshex 03D6,% lower case pi variant  
12689 \glshex 03A0% upper case pi  
12690 }
```

\glsxtrUpRho

```
12691 \newcommand*{\glsxtrUpRho}{%  
12692 \glshex 03C1% lower case rho  
12693 \string=\glshex 03F1,% lower case rho variant  
12694 \glshex 03A1% upper case rho  
12695 }
```

\glsxtrUpSigma

```
12696 \newcommand*{\glsxtrUpSigma}{%  
12697 \glshex 03C2% lower case sigma  
12698 \string=\glshex 03C3,% lower case sigma  
12699 \glshex 03A3% upper case sigma  
12700 }
```

\glsxtrUpTau

```
12701 \newcommand*{\glsxtrUpTau}{%  
12702 \glshex 03C4,% lower case tau  
12703 \glshex 03A4% upper case tau  
12704 }
```

glsxtrUpUpsilon

```
12705 \newcommand*{\glsxtrUpUpsilon}{%  
12706 \glshex 03C5,% lower case upsilon  
12707 \glshex 03A5% upper case upsilon  
12708 }
```

\glsxtrUpPhi

```
12709 \newcommand*{\glsxtrUpPhi}{%  
12710 \glshex 03C6% lower case phi
```

```

12711 \string=\glshex 03D5,% lower case phi variant
12712 \glshex 03A6% upper case phi
12713 }

```

$\backslash\mathrm{glxtrUpChi}$

```

12714 \newcommand*{\glxtrUpChi}{%
12715 \glshex 03C7,% lower case chi
12716 \glshex 03A7% upper case chi
12717 }

```

$\backslash\mathrm{glxtrUpPsi}$

```

12718 \newcommand*{\glxtrUpPsi}{%
12719 \glshex 03C8,% lower case psi
12720 \glshex 03A8% upper case psi
12721 }

```

$\backslash\mathrm{glxtrUpOmega}$

```

12722 \newcommand*{\glxtrUpOmega}{%
12723 \glshex 03C9,% lower case omega
12724 \glshex 03A9% upper case omega
12725 }

```

$\mathrm{MathItalicAlpha}$

```

12726 \newcommand*{\glxtrMathItalicAlpha}{%
12727 \glshex 1D6FC,% lower case alpha (maths italic)
12728 \glshex 1D6E2% upper case alpha (maths italic)
12729 }

```

$\mathrm{rMathItalicBeta}$

```

12730 \newcommand*{\glxtrMathItalicBeta}{%
12731 \glshex 1D6FD,% lower case beta (maths italic)
12732 \glshex 1D6E3% upper case beta (maths italic)
12733 }

```

$\mathrm{MathItalicGamma}$

```

12734 \newcommand*{\glxtrMathItalicGamma}{%
12735 \glshex 1D6FE,% lower case gamma (maths italic)
12736 \glshex 1D6E4% upper case gamma (maths italic)
12737 }

```

$\mathrm{MathItalicDelta}$

```

12738 \newcommand*{\glxtrMathItalicDelta}{%
12739 \glshex 1D6FF,% lower case delta (maths italic)
12740 \glshex 1D6E5% upper case delta (maths italic)
12741 }

```

$\mathrm{thItalicEpsilon}$

```

12742 \newcommand*{\glxtrMathItalicEpsilon}{%

```

```

12743 \glshex 1D700% lower case epsilon (maths italic)
12744 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12745 \glshex 1D6E6% upper case epsilon (maths italic)
12746 }

```

rMathItalicZeta

```

12747 \newcommand*{\glxtrMathItalicZeta}{%
12748 \glshex 1D701,% lower case zeta (maths italic)
12749 \glshex 1D6E7% upper case zeta (maths italic)
12750 }

```

trMathItalicEta

```

12751 \newcommand*{\glxtrMathItalicEta}{%
12752 \glshex 1D702,% lower case eta (maths italic)
12753 \glshex 1D6E8% upper case eta (maths italic)
12754 }

```

MathItalicTheta

```

12755 \newcommand*{\glxtrMathItalicTheta}{%
12756 \glshex 1D703% lower case theta (maths italic)
12757 \string=\glshex 1D717,% lower case theta variant (maths italic)
12758 \glshex 1D6E9% upper case theta (maths italic)
12759 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12760 }

```

rMathItalicIota

```

12761 \newcommand*{\glxtrMathItalicIota}{%
12762 \glshex 1D704,% lower case iota (maths italic)
12763 \glshex 1D6EA% upper case iota (maths italic)
12764 }

```

MathItalicKappa

```

12765 \newcommand*{\glxtrMathItalicKappa}{%
12766 \glshex 1D705% lower case kappa (maths italic)
12767 \string=\glshex 1D718,% lower case kappa variant (maths italic)
12768 \glshex 1D6EB% upper case kappa (maths italic)
12769 }

```

athItalicLambda

```

12770 \newcommand*{\glxtrMathItalicLambda}{%
12771 \glshex 1D706,% lower case lambda (maths italic)
12772 \glshex 1D6EC% upper case lambda (maths italic)
12773 }

```

xtrMathItalicMu

```

12774 \newcommand*{\glxtrMathItalicMu}{%
12775 \glshex 1D707,% lower case mu (maths italic)
12776 \glshex 1D6ED% upper case mu (maths italic)
12777 }

```

xtrMathItalicNu

```
12778 \newcommand*{\glxtrMathItalicNu}{%
12779 \glshex 1D708,% lower case nu (maths italic)
12780 \glshex 1D6EE% upper case nu (maths italic)
12781 }
```

xtrMathItalicXi

```
12782 \newcommand*{\glxtrMathItalicXi}{%
12783 \glshex 1D709,% lower case xi (maths italic)
12784 \glshex 1D6EF% upper case xi (maths italic)
12785 }
```

thItalicOmicron

```
12786 \newcommand*{\glxtrMathItalicOmicron}{%
12787 \glshex 1D70A,% lower case omicron (maths italic)
12788 \glshex 1D6F0% upper case omicron (maths italic)
12789 }
```

xtrMathItalicPi

```
12790 \newcommand*{\glxtrMathItalicPi}{%
12791 \glshex 1D70B% lower case pi (maths italic)
12792 \string=\glshex 1D71B,% lower case pi variant (maths italic)
12793 \glshex 1D6F1% upper case pi (maths italic)
12794 }
```

trMathItalicRho

```
12795 \newcommand*{\glxtrMathItalicRho}{%
12796 \glshex 1D70C% lower case rho (maths italic)
12797 \string=\glshex 1D71A,% lower case rho variant (maths italic)
12798 \glshex 1D6F2% upper case rho (maths italic)
12799 }
```

MathItalicSigma

```
12800 \newcommand*{\glxtrMathItalicSigma}{%
12801 \glshex 1D70D% lower case final sigma (maths italic)
12802 \string=\glshex 1D70E,% lower case sigma (maths italic)
12803 \glshex 1D6F4% upper case sigma (maths italic)
12804 }
```

trMathItalicTau

```
12805 \newcommand*{\glxtrMathItalicTau}{%
12806 \glshex 1D70F,% lower case tau (maths italic)
12807 \glshex 1D6F5% upper case tau (maths italic)
12808 }
```

thItalicUpsilon

```
12809 \newcommand*{\glxtrMathItalicUpsilon}{%
12810 \glshex 1D710,% lower case upsilon (maths italic)
```

```

12811 \glshex 1D6F6% upper case upsilon (maths italic)
12812 }

```

trMathItalicPhi

```

12813 \newcommand*{\glsxtrMathItalicPhi}{%
12814 \glshex 1D711% lower case phi (maths italic)
12815 \string=\glshex 1D719,% lower case phi variant (maths italic)
12816 \glshex 1D6F7% upper case phi (maths italic)
12817 }

```

trMathItalicChi

```

12818 \newcommand*{\glsxtrMathItalicChi}{%
12819 \glshex 1D712,% lower case chi (maths italic)
12820 \glshex 1D6F8% upper case chi (maths italic)
12821 }

```

trMathItalicPsi

```

12822 \newcommand*{\glsxtrMathItalicPsi}{%
12823 \glshex 1D713,% lower case psi (maths italic)
12824 \glshex 1D6F9% upper case psi (maths italic)
12825 }

```

MathItalicOmega

```

12826 \newcommand*{\glsxtrMathItalicOmega}{%
12827 \glshex 1D714,% lower case omega (maths italic)
12828 \glshex 1D6FA% upper case omega (maths italic)
12829 }

```

thItalicPartial

```

12830 \newcommand*{\glsxtrMathItalicPartial}{%
12831 \glshex 1D715% partial differential (maths italic)
12832 }

```

MathItalicNabla

```

12833 \newcommand*{\glsxtrMathItalicNabla}{%
12834 \glshex 1D6FB% nabla (maths italic)
12835 }

```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```

12836 \newcommand*{\glsxtrdigitrules}{%
12837 0\string=\glshex 2080\string=\glshex 2070
12838 \string<1\string=\glshex 2081\string=\glshex 00B9
12839 \string<2\string=\glshex 2082\string=\glshex 00B2
12840 \string<3\string=\glshex 2083\string=\glshex 00B3
12841 \string<4\string=\glshex 2084\string=\glshex 2074
12842 \string<5\string=\glshex 2085\string=\glshex 2075
12843 \string<6\string=\glshex 2086\string=\glshex 2076
12844 \string<7\string=\glshex 2087\string=\glshex 2077

```

```

12845 \string<8\string=\glshex 2088\string=\glshex 2078
12846 \string<9\string=\glshex 2089\string=\glshex 2079
12847 }

```

BasicDigitrules Digits from the Basic Latin set.

```

12848 \newcommand*{\glxtrBasicDigitrules}{%
12849 0\string<1\string<2\string<3\string<4%
12850 \string<5\string<6\string<7\string<8\string<9%
12851 }

```

criptDigitrules Subscript digits.

```

12852 \newcommand*{\glxtrSubScriptDigitrules}{%
12853 \glshex 2080% subscript 0
12854 \string<\glshex 2081% subscript 1
12855 \string<\glshex 2082% subscript 2
12856 \string<\glshex 2083% subscript 3
12857 \string<\glshex 2084% subscript 4
12858 \string<\glshex 2085% subscript 5
12859 \string<\glshex 2086% subscript 6
12860 \string<\glshex 2087% subscript 7
12861 \string<\glshex 2088% subscript 8
12862 \string<\glshex 2089% subscript 9
12863 }

```

criptDigitrules Superscript digits.

```

12864 \newcommand*{\glxtrSuperScriptDigitrules}{%
12865 \glshex 2070% superscript 0
12866 \string<\glshex 00B9% superscript 1
12867 \string<\glshex 00B2% superscript 2
12868 \string<\glshex 00B3% superscript 3
12869 \string<\glshex 2074% superscript 4
12870 \string<\glshex 2075% superscript 5
12871 \string<\glshex 2076% superscript 6
12872 \string<\glshex 2077% superscript 7
12873 \string<\glshex 2078% superscript 8
12874 \string<\glshex 2079% superscript 9
12875 }

```

trfractionrules Vulgar fractions.

```

12876 \newcommand*{\glxtrfractionrules}{%
12877 \glshex 215F% fraction numerator one (1/)
12878 \string<\glshex 2189% zero thirds (0/3 = 0)
12879 \string<\glshex 2152% one tenth (1/10 = 0.1)
12880 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12881 \string<\glshex 215B% one eighth (1/8 = 0.125)
12882 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12883 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12884 \string<\glshex 2155% one fifth (1/5 = 0.2)
12885 \string<\glshex 00BC% one quarter (1/4 = 0.25)

```



```

12886 \string<\glshex 2153% one third ( $1/3 \sim 0.333$ )
12887 \string<\glshex 215C% three eighths ( $3/8 = 0.375$ )
12888 \string<\glshex 2156% two fifths ( $2/5 = 0.4$ )
12889 \string<\glshex 00BD% one half ( $1/2 = 0.5$ )
12890 \string<\glshex 2157% three fifths ( $3/5 = 0.6$ )
12891 \string<\glshex 215D% five eighths ( $5/8 = 0.625$ )
12892 \string<\glshex 2154% two thirds ( $2/3 \sim 0.667$ )
12893 \string<\glshex 00BE% three quarters ( $3/4 = 0.75$ )
12894 \string<\glshex 2158% four fifths ( $4/5 = 0.8$ )
12895 \string<\glshex 215A% five sixths ( $5/6 \sim 0.833$ )
12896 \string<\glshex 215E% seven eighths ( $7/8 = 0.875$ )
12897 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

12898 \renewcommand{\@glxsxtrdialecthook}{%
12899   \ifundef\CurrentTrackedScript
12900   {%
12901     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
12902     {%
12903       \edef\CurrentTrackedScript{%
12904         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
12905       }%
12906     }%
12907   }%
12908   {}%
12909   \ifdef\CurrentTrackedScript
12910   {%
12911     \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
12912     \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
12913     \let\CurrentTrackedTag\CurrentTrackedScript
12914     \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
12915     {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
12916     {}%
12917     \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
12918   }%
12919   {}%
12920 }

```

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.

```

12921 \ifdef\glxsxtr@loaddialect
12922 {%
12923   \@ifpackageloaded{tracklang}
12924   {%
12925     \AnyTrackedLanguages
12926     {%
12927       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
12928     }%

```

12929 {}%
12930 }
12931 {}
12932 }
12933 {}

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:

```
12934 \NeedsTeXFormat{LaTeX2e}
12935 \ProvidesPackage{glossaries-extra-stylemods}[2018/05/09 v1.31 (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`.

`glxtr@loadstyles`

```
12936 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
12937 \DeclareOption{all}{%
12938   \appto\@glxtr@loadstyles{%
12939     \RequirePackage{glossary-inline}%
12940     \RequirePackage{glossary-list}%
12941     \RequirePackage{glossary-tree}%
12942     \RequirePackage{glossary-mcols}%
12943     \RequirePackage{glossary-long}%
12944     \RequirePackage{glossary-longragged}%
12945     \RequirePackage{glossary-longbooktabs}%
12946     \RequirePackage{glossary-super}%
12947     \RequirePackage{glossary-superragged}%
12948     \RequirePackage{glossary-bookindex}%
12949   }
12950 }

12951 \DeclareOption*{%
12952   \IfFileExists{glossary-\CurrentOption.sty}
12953   {\eappto\@glxtr@loadstyles{%
12954     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
12955   }%
12956   {%
12957     \PackageError{glossaries-extra-styles}%

```

```

12958     {Unknown option ‘\CurrentOption’}{}%
12959   }%
12960 }

```

Process the package options:

```
12961 \ProcessOptions
```

Load the required packages:

```
12962 \@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`.

```
12963 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

12964 \providecommand{\renewglossarystyle}[2]{%
12965   \ifcsundef{@glsstyle@#1}%
12966   {%
12967     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
12968   }%
12969   {%
12970     \csdef{@glsstyle@#1}{#2}%
12971   }%
12972 }

```

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.

```

12973 \ifdef{@glsstyle@listdotted}
12974 {%
12975   \renewglossarystyle{listdotted}{%
12976     \setglossarystyle{list}%
12977     \renewcommand*{\glossentry}[2]{%
12978       \item[]\makebox[\glslistdottedwidth][l]{%
12979         \glsentryitem{##1}%
12980         \glstarget{##1}{\glossentryname{##1}}%
12981         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12982         \glossentrydesc{##1}\glspostdescription}%
12983     \renewcommand*{\subglossentry}[3]{%
12984       \item[]\makebox[\glslistdottedwidth][l]{%
12985         \glssubentryitem{##2}%
12986         \glstarget{##2}{\glossentryname{##2}}%
12987         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12988         \glossentrydesc{##2}\glspostdescription}%
12989   }

```

```
12990 }
12991 {%
```

Assume the style isn't required if it hasn't already been defined.

```
12992 }
```

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.

```
12993 \ifdef{\@glsstyle@list}
12994 {%
```

listprelocation Space before number list for top-level entries.

```
12995 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
12996 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
12997 \newcommand{\glslistchildpostlocation}{.}
```

```
\glslistdesc
```

```
12998 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
12999 \renewglossarystyle{list}{%
13000 \renewenvironment{theglossary}%
13001 {\begin{description}}{\end{description}}%
13002 \renewcommand*\{glossaryheader}[1]{%
13003 \renewcommand*\{glsgroupheading}[1]{%
13004 \renewcommand*\{glossentry}[2]{%
13005 \item[\glssentryitem{##1}%
13006 \glstarget{##1}{\glossentryname{##1}}]
13007 \glslistdesc{##1}\glslistprelocation ##2}%
13008 \renewcommand*\{subglossentry}[3]{%
13009 \glssubentryitem{##2}%
13010 \glstarget{##2}{\strut}\space
13011 \glslistdesc{##2}%
13012 \glslistchildprelocation ##3\glslistchildpostlocation}%
13013 \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13014 }
13015 }
13016 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13017 \ifdef{\@glsstyle@altlist}
13018 {%
13019 \renewglossarystyle{altlist}{%
```

```

13020 \setglossarystyle{list}%
13021 \renewcommand*{\glossentry}[2]{%
13022   \item[\glentryitem{##1}%
13023     \glstarget{##1}{\glossentryname{##1}}]%
13024   \mbox{}\par\nobreak\@afterheading
13025   \glslistdesc{##1}\glslistprelocation ##2}%
13026 \renewcommand{\subglossentry}[3]{%
13027   \par
13028   \glssubentryitem{##2}%
13029   \glstarget{##2}{\strut}\glslistdesc{##2}%
13030   \glslistchildprelocation ##3}%
13031 }
13032 }
13033 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13034 \ifdef{\@glsstyle@listgroup}
13035 {%
13036   \renewglossarystyle{listgroup}{%
13037     \setglossarystyle{list}%
13038     \renewcommand*{\glsgroupheading}[1]{%
13039       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13040       \mbox{}\par\nobreak\@afterheading
13041     }%
13042   }
13043 }
13044 {}

```

Similarly for listhypergroup.

```

13045 \ifdef{\@glsstyle@listhypergroup}
13046 {%
13047   \renewglossarystyle{listhypergroup}{%
13048     \setglossarystyle{list}%
13049     \renewcommand*{\glossaryheader}{%
13050       \glslistnavigationitem{\glsnavigation}}%
13051     \renewcommand*{\glsgroupheading}[1]{%
13052       \item[\glslistgroupheaderfmt
13053         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13054       \mbox{}\par\nobreak\@afterheading
13055     }%
13056   }
13057 }
13058 {}

```

Similarly for altlistgroup.

```

13059 \ifdef{\@glsstyle@altlistgroup}
13060 {%
13061   \renewglossarystyle{altlistgroup}{%
13062     \setglossarystyle{altlist}%
13063     \renewcommand*{\glsgroupheading}[1]{%
13064       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```

```

13065      \mbox{}\par\nobreak\@afterheading
13066    }%
13067  }
13068 }
13069 {}

```

Similarly for altlisthypergroup.

```

13070 \ifdef{\@glsstyle@altlisthypergroup}
13071 {%
13072   \renewglossarystyle{altlisthypergroup}{%
13073     \setglossarystyle{altlist}%
13074     \renewcommand*\glossaryheader{%
13075       \glslistnavigationitem{\glsnavigation}}%
13076     \renewcommand*\glsgroupheading[1]{%
13077       \item[\glslistgroupheaderfmt
13078         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13079       \mbox{}\par\nobreak\@afterheading
13080     }%
13081   }
13082 }
13083 {}

```

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`.

```

13084 \ifcsdef{@glsstyle@long}
13085 {%
13086   \renewglossarystyle{long}{%
13087     \renewenvironment{theglossary}%
13088       {\begin{longtable}{lp{\glsdescwidth}}}%
13089       {\end{longtable}}%
13090     \renewcommand*\glossaryheader{}%
13091     \renewcommand*\glsgroupheading[1]{}%
13092     \renewcommand{\glossentry}[2]{%
13093       \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13094       \glossentrydesc{##1}\glspostdescription
13095       \glxtrprelocation ##2\tabularnewline
13096     }%
13097     \renewcommand{\subglossentry}[3]{%
13098       &
13099       \glssubentryitem{##2}%
13100       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13101       \glxtrprelocation ##3\tabularnewline
13102     }%
13103     \ifglsnogroupskip
13104       \renewcommand*\glsgroupskip{}%
13105     \else

```

```

13106      \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13107      \fi
13108    }
13109  }
13110 {}

```

Three column style:

```

13111 \ifcsdef{@glsstyle@long3col}
13112 {%
13113   \renewglossarystyle{long3col}{%
13114     \renewenvironment{theglossary}%
13115       {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13116       {\end{longtable}}}%
13117     \renewcommand*{\glossaryheader}{}%
13118     \renewcommand*{\glsgroupheading}[1]{}%
13119     \renewcommand{\glossentry}[2]{%
13120       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13121       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13122     }%
13123     \renewcommand{\subglossentry}[3]{%
13124       &
13125       \glssubentryitem{##2}%
13126       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13127       ##3\tabularnewline
13128     }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

13129   \ifglsnogroupskip
13130     \renewcommand*{\glsgroupskip}{}%
13131   \else
13132     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13133   \fi
13134 }
13135 }
13136 {}

```

Four column style:

```

13137 \ifcsdef{@glsstyle@long4col}
13138 {%
13139   \renewglossarystyle{long4col}{%
13140     \renewenvironment{theglossary}%
13141       {\begin{longtable}{l111}}}%
13142       {\end{longtable}}}%
13143     \renewcommand*{\glossaryheader}{}%
13144     \renewcommand*{\glsgroupheading}[1]{}%
13145     \renewcommand{\glossentry}[2]{%
13146       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13147       \glossentrydesc{##1}\glspostdescription &
13148       \glossentrysymbol{##1} &
13149       ##2\tabularnewline

```



```

13150 }%
13151 \renewcommand{\subglossentry}[3]{%
13152     &
13153     \glssubentryitem{##2}%
13154     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13155     \glossentrysymbol{##2} & ##3\tabularnewline
13156 }%

13157 \ifglsgroupskip
13158     \renewcommand*\{glsgroupskip}{}%
13159 \else
13160     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13161 \fi
13162 }
13163 }
13164 {}

```

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`.

```

13165 \ifcsdef{@glstyle@longragged}
13166 {%
13167     \renewglossarystyle{longragged}{%
13168         \renewenvironment{theglossary}%
13169             {\begin{longtable}[l>\raggedright]p{\glstdescwidth}}}%
13170             {\end{longtable}}%
13171         \renewcommand*\{glossaryheader}{}%
13172         \renewcommand*\{glsgroupheading}[1]{}%
13173         \renewcommand{\glossentry}[2]{%
13174             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13175             \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13176             \tabularnewline
13177         }%
13178         \renewcommand{\subglossentry}[3]{%
13179             &
13180             \glssubentryitem{##2}%
13181             \glstarget{##2}{\strut}\glossentrydesc{##2}%
13182             \glspostdescription\glxtrprelocation ##3%
13183             \tabularnewline
13184         }%
13185         \ifglsgroupskip
13186             \renewcommand*\{glsgroupskip}{}%
13187         \else
13188             \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13189 \fi
13190 }
13191 }
13192 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13193 \ifcsdef{@glsstyle@longragged3col}
13194 {%
13195 \renewglossarystyle{longragged3col}{%
13196 \renewenvironment{theglossary}%
13197 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13198 >{\raggedright}p{\glspagelistwidth}}}%
13199 {\end{longtable}}}%
13200 \renewcommand*{\glossaryheader}{}%
13201 \renewcommand*{\glsgroupheading}[1]{}%
13202 \renewcommand{\glossentry}[2]{%
13203 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13204 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13205 }%
13206 \renewcommand{\subglossentry}[3]{%
13207 &
13208 \glssubentryitem{##2}%
13209 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13210 ##3\tabularnewline
13211 }%
13212 \ifglsnogroupskip
13213 \renewcommand*{\glsgroupskip}{}%
13214 \else
13215 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13216 \fi
13217 }
13218 }
13219 {}

```

Four column style:

```

13220 \ifcsdef{@glsstyle@altlongragged4col}
13221 {%
13222 \renewglossarystyle{altlongragged4col}{%
13223 \renewenvironment{theglossary}%
13224 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13225 >{\raggedright}p{\glspagelistwidth}}}%
13226 {\end{longtable}}}%
13227 \renewcommand*{\glossaryheader}{}%
13228 \renewcommand*{\glsgroupheading}[1]{}%
13229 \renewcommand{\glossentry}[2]{%
13230 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13231 \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13232 ##2\tabularnewline

```

```

13233 }%
13234 \renewcommand{\subglossentry}[3]{%
13235     &
13236     \glssubentryitem{##2}%
13237     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13238     \glossentrysymbol{##2} & ##3\tabularnewline
13239 }%

13240 \ifglsnogroupskip
13241     \renewcommand*\{glsgroupskip}{}%
13242 \else
13243     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13244 \fi
13245 }
13246 }
13247 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded \space changed to \glxtrprelocation.

```

13248 \ifcsdef{@glsstyle@super}
13249 {%
13250     \renewglossarystyle{super}{%
13251         \renewenvironment{theglossary}%
13252             {\tablehead{}}\tabletail{}}%
13253         \begin{supertabular}{lp{\glsdescwidth}}}%
13254         {\end{supertabular}}%
13255     \renewcommand*\{glossaryheader}{}%
13256     \renewcommand*\{glsgroupheading}[1]{}%
13257     \renewcommand{\glossentry}[2]{%
13258         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13259         \glossentrydesc{##1}\glspostdescription
13260         \glxtrprelocation ##2\tabularnewline
13261     }%
13262     \renewcommand{\subglossentry}[3]{%
13263         &
13264         \glssubentryitem{##2}%
13265         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13266         \glxtrprelocation ##3\tabularnewline
13267     }%
13268     \ifglsnogroupskip
13269         \renewcommand*\{glsgroupskip}{}%
13270     \else
13271         \renewcommand*\{glsgroupskip}{& \tabularnewline}%
13272     \fi
13273 }
13274 }
13275 {}

```

Three column style:

```

13276 \ifcsdef{@glsstyle@super3col}
13277 {%
13278   \renewglossarystyle{super3col}{%
13279     \renewenvironment{theglossary}%
13280       {\tablehead{}\tabletail{}}%
13281       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13282     {\end{supertabular}}}%
13283   \renewcommand*{\glossaryheader}{}%
13284   \renewcommand*{\glsgroupheading}[1]{}%
13285   \renewcommand{\glossentry}[2]{%
13286     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13287     \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
13288   }%
13289   \renewcommand{\subglossentry}[3]{%
13290     &
13291     \glssubentryitem{##2}%
13292     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13293     ##3\tabularnewline
13294   }%

13295   \ifglsgroupskip
13296     \renewcommand*{\glsgroupskip}{}%
13297   \else
13298     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13299   \fi
13300 }
13301 }
13302 {}

```

Four column styles:

```

13303 \ifcsdef{@glsstyle@super4col}
13304 {%
13305   \renewglossarystyle{super4col}{%
13306     \renewenvironment{theglossary}%
13307       {\tablehead{}\tabletail{}}%
13308       \begin{supertabular}{llll}}}%
13309     \end{supertabular}}}%
13310   \renewcommand*{\glossaryheader}{}%
13311   \renewcommand*{\glsgroupheading}[1]{}%
13312   \renewcommand{\glossentry}[2]{%
13313     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13314     \glsentrydesc{##1}\glspostdescription &
13315     \glsentrysymbol{##1} & ##2\tabularnewline
13316   }%
13317   \renewcommand{\subglossentry}[3]{%
13318     &
13319     \glssubentryitem{##2}%
13320     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13321     \glsentrysymbol{##2} & ##3\tabularnewline

```

```

13322 }%
13323 \ifglsgroupskip
13324 \renewcommand*{\glsgroupskip}{}%
13325 \else
13326 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13327 \fi
13328 }
13329 }
13330 {}

```

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`.

```

13331 \ifcsdef{@glstyle@superragged}
13332 {%
13333 \renewglossarystyle{superragged}{%
13334 \renewenvironment{theglossary}%
13335 {\tablehead{}\tabletail}%
13336 \begin{supertabular}{l>\raggedright}p{\glsgdescwidth}}%
13337 {\end{supertabular}}%
13338 \renewcommand*{\glossaryheader}{}%
13339 \renewcommand*{\glsgroupheading}[1]{}%
13340 \renewcommand{\glossentry}[2]{%
13341 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13342 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
13343 \tabularnewline
13344 }%
13345 \renewcommand{\subglossentry}[3]{%
13346 &
13347 \glssubentryitem{##2}%
13348 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13349 \glstrprelocation ##3%
13350 \tabularnewline
13351 }%
13352 \ifglsgroupskip
13353 \renewcommand*{\glsgroupskip}{}%
13354 \else
13355 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13356 \fi
13357 }
13358 }
13359 {}

```

Three column style:

```

13360 \ifcsdef{@glstyle@superragged3col}
13361 {%

```

```

13362 \renewglossarystyle{superragged3col}{%
13363   \renewenvironment{theglossary}%
13364     {\tablehead{}\tabletail{}}%
13365     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
13366       >{\raggedright}p{\glspagelistwidth}}}%
13367     {\end{supertabular}}%
13368   \renewcommand*{\glossaryheader}{}%
13369   \renewcommand*{\glsgroupheading}[1]{}%
13370   \renewcommand{\glossentry}[2]{%
13371     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13372     \glossentrydesc{##1}\glspostdescription &
13373     ##2\tabularnewline
13374   }%
13375   \renewcommand{\subglossentry}[3]{%
13376     &
13377     \glssubentryitem{##2}%
13378     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13379     ##3\tabularnewline
13380   }%

13381   \ifglsnogroupskip
13382     \renewcommand*{\glsgroupskip}{}%
13383   \else
13384     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13385   \fi
13386 }
13387 }
13388 {}

```

Four columns:

```

13389 \ifcsdef{@glsstyle@altsuperragged4col}
13390 {%
13391   \renewglossarystyle{altsuperragged4col}{%
13392     \renewenvironment{theglossary}%
13393       {\tablehead{}\tabletail{}}%
13394       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
13395         >{\raggedright}p{\glspagelistwidth}}}%
13396       {\end{supertabular}}%
13397     \renewcommand*{\glossaryheader}{}%
13398     \renewcommand{\glossentry}[2]{%
13399       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13400       \glossentrydesc{##1}\glspostdescription &
13401       \glossentrysymbol{##1} & ##2\tabularnewline
13402     }%
13403     \renewcommand{\subglossentry}[3]{%
13404       &
13405       \glssubentryitem{##2}%
13406       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13407       \glossentrysymbol{##2} & ##3\tabularnewline
13408     }%

```

```

13409 \ifglsnogroupskip
13410 \renewcommand*{\glsgroupskip}{}%
13411 \else
13412 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13413 \fi
13414 }
13415 }
13416 {}

```

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`.

```

13417 \ifdef{\@glsstyle@inline}
13418 {%
13419 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13420 \renewcommand*{\glsinlinedescformat}[3]{%
13421 \space#1\glxtrpostdescription}
13422 \renewcommand*{\glsinlinesubdescformat}[3]{%
13423 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13424 }
13425 {}

```

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.

```

13426 \ifdef\glstreenamfmt
13427 {

```

```

\glstreedefaultnamfmt

```

```

13428 \newcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}

```

```

\glstreenamfmt

```

```

13429 \renewcommand{\glstreenamfmt}[1]{\glstreedefaultnamfmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13430 \def\glstreegroupheaderfmt#1{\glstreedefaultnamfmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13431 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13432 }
13433 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13434 \ifdef{\@glsstyle@index}
13435 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13436 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13437 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13438 \renewglossarystyle{index}{%
13439   \renewenvironment{theglossary}%
13440     {\setlength{\parindent}{0pt}%
13441      \setlength{\parskip}{0pt plus 0.3pt}%
13442      \let\item\glstreeitem
13443      \let\subitem\glstreesubitem
13444      \let\subsubitem\glstreesubsubitem
13445     }%
13446   {\par}%
13447   \renewcommand*{\glossaryheader}{}%
13448   \renewcommand*{\glsgroupheading}[1]{}%
13449   \renewcommand*{\glossentry}[2]{%
13450     \item\glssentryitem{##1}%
13451     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13452     \glstreesymbol{##1}%
13453     \glstreedesc{##1}%
13454     \glstreeprelocation ##2%
13455   }%
13456   \renewcommand{\subglossentry}[3]{%
13457     \ifcase##1\relax
13458       \item
13459     \or
13460       \subitem
13461       \glssubentryitem{##2}%
13462     \else
13463       \subsubitem
13464     \fi
13465     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13466     \glstreechildsymbol{##2}%
13467     \glstreechilddesc{##2}%
13468     \glstreechildprelocation ##3%
13469   }%
```



```

13470 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
13471 }
13472 }
13473 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13474 \ifdef{\@glsstyle@indexgroup}
13475 {%
13476 \renewglossarystyle{indexgroup}{%
13477 \setglossarystyle{index}%
13478 \renewcommand*{\glsgroupheading}[1]{%
13479 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13480 \nopagebreak\indexspace
13481 \nobreak\@afterheading
13482 }%
13483 }
13484 }
13485 {}

```

Similarly for indexhypergroup.

```

13486 \ifdef{\@glsstyle@indexhypergroup}
13487 {%
13488 \renewglossarystyle{indexhypergroup}{%
13489 \setglossarystyle{index}%
13490 \renewcommand*{\glossaryheader}{%
13491 \item\glstreenavigationfmt{\glsnavigation}%
13492 \nobreak\@afterheading\indexspace}%
13493 \renewcommand*{\glsgroupheading}[1]{%
13494 \item\glstreegroupheaderfmt
13495 {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13496 \nopagebreak\indexspace
13497 \nobreak\@afterheading}%
13498 }%
13499 }
13500 {}

```

Adjust tree style to remove hard coded space before number list.

```

13501 \ifdef{\@glsstyle@tree}
13502 {%
13503 %Provide a command for use with the \glostyle{tree} styles that displays
13504 %the pre-description separator, the
13505 %description and post-description hook.
13506 %\begin{macro}{\glstreedesc}
13507 %\changes{1.31}{2018-05-09}{new}
13508 % \begin{macrocode}
13509 \newcommand{\glstreedesc}[1]{%
13510 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13511 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13512 \newcommand{\glstreesymbol}[1]{%
13513   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13514   }%
```

And for the child entries:

`lstreechilddesc`

```
13515 \newcommand{\glstreechilddesc}[1]{%
13516   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13517   }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13518 \newcommand{\glstreechildsymbol}[1]{%
13519   \glstreesymbol{#1}%
13520   }%

13521 \renewglossarystyle{tree}{%
13522   \renewenvironment{theglossary}%
13523     {\setlength{\parindent}{0pt}%
13524     \setlength{\parskip}{0pt plus 0.3pt}}%
13525   {%
13526     \renewcommand*{\glossaryheader}{}%
13527     \renewcommand*{\glsgroupheading}[1]{}%
13528     \renewcommand{\glossentry}[2]{%
13529       \hangindent0pt\relax
13530       \parindent0pt\relax
13531       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13532       \glstreesymbol{##1}%
13533       \glstreedesc{##1}%
13534       \glstreeprelocation##2\par
13535     }%
13536     \renewcommand{\subglossentry}[3]{%
13537       \hangindent##1\glstreeindent\relax
13538       \parindent##1\glstreeindent\relax
13539       \ifnum##1=1\relax
13540         \glssubentryitem{##2}%
13541       \fi
13542       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13543       \glstreechildsymbol{##2}%
13544       \glstreechilddesc{##2}%
13545       \glstreechildprelocation ##3\par
13546     }%
13547     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13548   }%
13549 }
13550 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13551 \ifdef{\@glstyle@treegroup}
```

```

13552 {%
13553   \renewglossarystyle{treegroup}{%
13554     \setglossarystyle{tree}%
13555     \renewcommand{\glsgroupheading}[1]{\par
13556       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13557       \nopagebreak\indexspace\nobreak\@afterheading}%
13558   }
13559 }
13560 {}

```

Similarly for treehypergroup

```

13561 \ifdef{\@glsstyle@treehypergroup}
13562 {%
13563   \renewglossarystyle{treehypergroup}{%
13564     \setglossarystyle{tree}%
13565     \renewcommand*\{\glossaryheader}{%
13566       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13567       \nobreak\@afterheading\indexspace}%
13568     \renewcommand*\{\glsgroupheading}[1]{%
13569       \par\noindent
13570       \glstreegroupheaderfmt
13571       {\glstravhypertarget{##1}{\glsgrouptitle{##1}}}\par
13572       \nopagebreak\indexspace\nobreak\@afterheading}%
13573   }
13574 }
13575 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13576 \ifdef{\@glsstyle@treenoname}
13577 {%
13578 %Provide a command for use with the \glostyle{treenoname} styles that displays
13579 %the pre-description separator, the
13580 %description and post-description hook.
13581 %\begin{macro}{\glstreenonamedesc}
13582 %\changes{1.31}{2018-05-09}{new}
13583 %   \begin{macrocode}
13584   \newcommand{\glstreenonamedesc}[1]{%
13585     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13586   }%

```

Similarly for the symbol.

treenonamesymbol

```

13587   \newcommand{\glstreenonamesymbol}[1]{%
13588     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13589   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13590   \newcommand{\glstreenonamechilddesc}[1]{%
13591     \glossentrydesc{#1}\glspostdescription
13592   }%

```

```

13593 \renewglossarystyle{treenoname}{%
13594   \renewenvironment{theglossary}%
13595     {\setlength{\parindent}{0pt}%
13596      \setlength{\parskip}{0pt plus 0.3pt}}%
13597     {}%
13598   \renewcommand*{\glossaryheader}{}%
13599   \renewcommand*{\glsgroupheading}[1]{}%
13600   \renewcommand{\glossentry}[2]{%
13601     \hangindent0pt\relax
13602     \parindent0pt\relax
13603     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13604     \glstreenonamesymbol{##1}%
13605     \glstreenonamedesc{##1}%
13606     \glstreeprelocation##2\par
13607   }%
13608   \renewcommand{\subglossentry}[3]{%
13609     \hangindent##1\glstreeindent\relax
13610     \parindent##1\glstreeindent\relax
13611     \ifnum##1=1\relax
13612       \glssubentryitem{##2}%
13613     \fi
13614     \glstarget{##2}{\strut}%
13615     \glstreenonamechilddesc{##2}%
13616     \glstreechildprelocation##3\par
13617   }%
13618   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13619 }
13620 }
13621 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13622 \ifdef{\@glsstyle@treenonamegroup}
13623 {%
13624   \renewglossarystyle{treenonamegroup}{%
13625     \setglossarystyle{treenoname}%
13626     \renewcommand{\glsgroupheading}[1]{\par
13627       \noindent\glstreegroupheaderfmt
13628       {\glsgetgrouptitle{##1}}}%
13629     \nopagebreak\indexspace\nobreak\@afterheading
13630   }%
13631 }
13632 }
13633 {}

```

Similarly for `treenonamehypergroup`

```

13634 \ifdef{\@glsstyle@treenonamehypergroup}
13635 {%
13636   \renewglossarystyle{treenonamehypergroup}{%
13637     \setglossarystyle{treenoname}%
13638     \renewcommand*{\glossaryheader}{%

```

```

13639     \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13640     \nobreak\@afterheading\indexspace}%
13641   \renewcommand*{\glsgroupheading}[1]{%
13642     \par\noindent
13643     \glstreegroupheaderfmt
13644     {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13645     \nopagebreak\indexspace\nobreak\@afterheading}%
13646   }
13647 }
13648 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13649 \ifdef{\@glstyle@almtree}
13650 {}

```

Only redefine this if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

13651 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13652   {%
13653     \let\par\glxtrAltTreePar
13654     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13655     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13656   }%
13657 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13658 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13659 \newcommand{\glxtrAltTreePar}{%
13660   \@@par
13661   \glxtrAltTreeSetHangIndent
13662   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13663 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13664 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13665   \glxtralttreeSymbolDescLocation{#2}{#3}%
13666 }

```

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.

```
13667 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13668 \newcommand*\glxstralttreeInit{%
13669   \settowidth{\glxstrtreetopindent}{\glstreenamfmt{\glsgetwidestname\space}}%
13670   \glxstrAltTreeIndent=\parindent
13671 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13672 \newcommand*\gglsetwidest}[2][0]{%
13673   \csgdef{@glswidestname\romannumeral#1}{#2}%
13674 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13675 \newcommand*\eglssetwidest}[2][0]{%
13676   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13677 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13678 \newcommand*\xglsetwidest}[2][0]{%
13679   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13680 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13681 \newcommand*\glupdatewidest}[2][0]{%
13682   \ifcsundef{@glswidestname\romannumeral#1}%
13683   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13684   {%
13685     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13686     \settowidth{\dimen@ii}{#2}%
13687     \ifdim\dimen@ii>\dimen@
13688       \csdef{@glswidestname\romannumeral#1}{#2}%
13689     \fi
13690   }%
13691 }
```

glupdatewidest As above but global definition.

```
13692 \newcommand*\gglupdatewidest}[2][0]{%
13693   \ifcsundef{@glswidestname\romannumeral#1}%
13694   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13695   {%
13696     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13697     \settowidth{\dimen@ii}{#2}%
13698     \ifdim\dimen@ii>\dimen@
13699       \csgdef{@glswidestname\romannumeral#1}{#2}%
13700     \fi
```

```

13701     }%
13702   }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13703   \newcommand*{\eglsupdatewidest}[2][0]{%
13704     \ifcsundef{@glswidestname\romannumeral#1}%
13705     {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13706     {%
13707       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13708       \settowidth{\dimen@ii}{#2}%
13709       \ifdim\dimen@ii>\dimen0
13710       \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13711       \fi
13712     }%
13713   }

```

`glsupdatewidest` As above but global.

```

13714   \newcommand*{\xglsupdatewidest}[2][0]{%
13715     \ifcsundef{@glswidestname\romannumeral#1}%
13716     {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13717     {%
13718       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13719       \settowidth{\dimen@ii}{#2}%
13720       \ifdim\dimen@ii>\dimen0
13721       \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13722       \fi
13723     }%
13724   }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13725   \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glsgetwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13726   \newcommand*{\glsgetwidestsubname}[1]{%
13727     \ifcsundef{@glswidestname\romannumeral#1}%
13728     {\@glswidestname}%
13729     {\csuse{@glswidestname\romannumeral#1}}%
13730   }

```

`glsfindwidesttoplevelname` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`.

```

13731   \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.

```

13732   \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
13733     \dimen@=0pt\relax

```

```

13734 \gls@tmplen=Opt\relax
13735 \foralllglossaries[#1]{\@gls@type}%
13736 {%
13737   \forglsentries[\@gls@type]{\@glo@label}%
13738   {%
13739     \ifglsused{\@glo@label}%
13740     {%
13741       \ifglshasparent{\@glo@label}%
13742       {}%
13743       {%
13744         \settowidth{\dimen@}%
13745           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13746         \ifdim\dimen@>\gls@tmplen
13747           \gls@tmplen=\dimen@
13748           \eglssetwidest{\glsentryname{\@glo@label}}%
13749         \fi
13750       }%
13751     }%
13752   }%
13753 }%
13754 }%
13755 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13756 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13757   \dimen@=Opt\relax
13758   \gls@tmplen=Opt\relax
13759   \foralllglossaries[#1]{\@gls@type}%
13760   {%
13761     \forglsentries[\@gls@type]{\@glo@label}%
13762     {%
13763       \ifglsused{\@glo@label}%
13764       {%
13765         \settowidth{\dimen@}%
13766           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13767         \ifdim\dimen@>\gls@tmplen
13768           \gls@tmplen=\dimen@
13769           \eglssetwidest{\glsentryname{\@glo@label}}%
13770         \fi
13771       }%
13772     }%
13773   }%
13774 }%
13775 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

13776 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13777   \dimen@=Opt\relax

```



```

13778 \gls@tmplen=0pt\relax
13779 \forallglossaries[#1]{\@gls@type}%
13780 {%
13781   \forglsentries[\@gls@type]{\@glo@label}%
13782   {%
13783     \settowidth{\dimen@}%
13784     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13785     \ifdim\dimen@>\gls@tmplen
13786       \gls@tmplen=\dimen@
13787       \eglssetwidest{\glsentryname{\@glo@label}}%
13788     \fi
13789   }%
13790 }%
13791 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

13792 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13793   \dimen@=0pt\relax
13794   \dimen@i=0pt\relax
13795   \dimen@ii=0pt\relax
13796   \forallglossaries[#1]{\@gls@type}%
13797   {%
13798     \forglsentries[\@gls@type]{\@glo@label}%
13799     {%
13800       \ifglsused{\@glo@label}%
13801       {%
13802         \ifglshasparent{\@glo@label}%
13803         {%
13804           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
13805           \ifglshasparent{\@glo@parent}%
13806           {%
13807             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
13808             \ifglshasparent{\@glo@parent}%
13809             {}%
13810             {%
13811               \settowidth{\gls@tmplen}%
13812               {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13813               \ifdim\gls@tmplen>\dimen@ii
13814                 \dimen@ii=\gls@tmplen
13815                 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13816               \fi
13817             }%
13818           }%
13819         }%
13820         \settowidth{\gls@tmplen}%
13821         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13822         \ifdim\gls@tmplen>\dimen@i
13823         \dimen@i=\gls@tmplen

```

```

13824         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13825     \fi
13826 }%
13827 }%
13828 {%
13829     \settowidth{\gls@tmplen}%
13830     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13831     \ifdim\gls@tmplen>\dimen@
13832         \dimen@=\gls@tmplen
13833         \eglssetwidest{\glsentryname{\@glo@label}}%
13834     \fi
13835 }%
13836 }%
13837 {}%
13838 }%
13839 }%
13840 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13841 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13842     \dimen@=0pt\relax
13843     \dimen@i=0pt\relax
13844     \dimen@ii=0pt\relax
13845     \foralllglossaries[#1]{\@gls@type}%
13846     {%
13847         \forglsentries[\@gls@type]{\@glo@label}%
13848         {%
13849             \ifglshasparent{\@glo@label}%
13850             {%
13851                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13852                 \ifglshasparent{\@glo@parent}%
13853                 {%
13854                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13855                     \ifglshasparent{\@glo@parent}%
13856                     {}%
13857                 }%
13858                 \settowidth{\gls@tmplen}%
13859                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13860                 \ifdim\gls@tmplen>\dimen@ii
13861                     \dimen@ii=\gls@tmplen
13862                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13863                 \fi
13864             }%
13865         }%
13866     {%
13867         \settowidth{\gls@tmplen}%
13868         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13869         \ifdim\gls@tmplen>\dimen@i
13870             \dimen@i=\gls@tmplen

```

```

13871         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13872     \fi
13873 }%
13874 }%
13875 {%
13876     \settowidth{\gls@tmplen}%
13877         {\glsstreenamefmt{\glsentryname{\@glo@label}}}%
13878     \ifdim\gls@tmplen>\dimen@
13879         \dimen@=\gls@tmplen
13880     \eglssetwidest{\glsentryname{\@glo@label}}%
13881     \fi
13882 }%
13883 }%
13884 }%
13885 }

```

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.

```

13886 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13887     \dimen@=0pt\relax
13888     \gls@tmplen=0pt\relax
13889     #2=0pt\relax
13890     \foralllglossaries[#1]{\@gls@type}%
13891     {%
13892         \forglsentries[\@gls@type]{\@glo@label}%
13893         {%
13894             \ifglsused{\@glo@label}%
13895             {%
13896                 \settowidth{\dimen@}%
13897                     {\glsstreenamefmt{\glsentryname{\@glo@label}}}%
13898                 \ifdim\dimen@>\gls@tmplen
13899                     \gls@tmplen=\dimen@
13900                 \eglssetwidest{\glsentryname{\@glo@label}}%
13901                 \fi
13902                 \settowidth{\dimen@}%
13903                     {\glsentrysymbol{\@glo@label}}%
13904                 \ifdim\dimen@>#2\relax
13905                     #2=\dimen@
13906                 \fi
13907             }%
13908         }%
13909     }%
13910 }%
13911 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

13912 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
13913     \dimen@=0pt\relax
13914     \gls@tmplen=0pt\relax

```

```

13915      #2=Opt\relax
13916      \foralllglossaries[#1]{\@gls@type}%
13917      {%
13918        \forglsentries[\@gls@type]{\@glo@label}%
13919        {%
13920          \settowidth{\dimen@}%
13921            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13922          \ifdim\dimen@>\gls@tmplen
13923            \gls@tmplen=\dimen@
13924            \eglssetwidest{\glsentryname{\@glo@label}}%
13925          \fi
13926          \settowidth{\dimen@}%
13927            {\glsentrysymbol{\@glo@label}}%
13928          \ifdim\dimen@>#2\relax
13929            #2=\dimen@
13930          \fi
13931        }%
13932      }%
13933    }

```

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.

```

13934    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
13935      \dimen@=Opt\relax
13936      \gls@tmplen=Opt\relax
13937      #2=Opt\relax
13938      #3=Opt\relax
13939      \foralllglossaries[#1]{\@gls@type}%
13940      {%
13941        \forglsentries[\@gls@type]{\@glo@label}%
13942        {%
13943          \ifglsused{\@glo@label}%
13944          {%
13945            \settowidth{\dimen@}%
13946              {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13947            \ifdim\dimen@>\gls@tmplen
13948              \gls@tmplen=\dimen@
13949              \eglssetwidest{\glsentryname{\@glo@label}}%
13950            \fi
13951            \settowidth{\dimen@}%
13952              {\glsentrysymbol{\@glo@label}}%
13953            \ifdim\dimen@>#2\relax
13954              #2=\dimen@
13955            \fi
13956            \settowidth{\dimen@}%
13957              {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13958            \ifdim\dimen@>#3\relax

```

```

13959         #3=\dimen@
13960     \fi
13961 }%
13962 {}%
13963 }%
13964 }%
13965 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

13966 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
13967     \dimen@=0pt\relax
13968     \gls@tmplen=0pt\relax
13969     #2=0pt\relax
13970     #3=0pt\relax
13971     \forallglossaries[#1]{\@gls@type}%
13972     {%
13973         \forglsentries[\@gls@type]{\@glo@label}%
13974         {%
13975             \settowidth{\dimen@}%
13976             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13977             \ifdim\dimen@>\gls@tmplen
13978                 \gls@tmplen=\dimen@
13979                 \eglssetwidest{\glsentryname{\@glo@label}}%
13980             \fi
13981             \settowidth{\dimen@}%
13982             {\glsentrysymbol{\@glo@label}}%
13983             \ifdim\dimen@>#2\relax
13984                 #2=\dimen@
13985             \fi
13986             \settowidth{\dimen@}%
13987             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13988             \ifdim\dimen@>#3\relax
13989                 #3=\dimen@
13990             \fi
13991         }%
13992     }%
13993 }

```

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.

```

13994 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
13995     \dimen@=0pt\relax
13996     \gls@tmplen=0pt\relax
13997     #2=0pt\relax
13998     \forallglossaries[#1]{\@gls@type}%
13999     {%
14000         \forglsentries[\@gls@type]{\@glo@label}%
14001         {%

```

```

14002     \ifglsused{\@glo@label}%
14003     {%
14004         \settowidth{\dimen@}%
14005         {\glstreenamfmt{\glentryname{\@glo@label}}}%
14006         \ifdim\dimen@>\gls@tmplen
14007             \gls@tmplen=\dimen@
14008             \eglssetwidest{\glentryname{\@glo@label}}%
14009         \fi
14010         \settowidth{\dimen@}%
14011         {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14012         \ifdim\dimen@>#2\relax
14013             #2=\dimen@
14014         \fi
14015     }%
14016     {}%
14017 }%
14018 }%
14019 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14020 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14021     \dimen@=0pt\relax
14022     \gls@tmplen=0pt\relax
14023     #2=0pt\relax
14024     \forallglossaries[#1]{\@gls@type}%
14025     {%
14026         \forglsentries[\@gls@type]{\@glo@label}%
14027         {%
14028             \settowidth{\dimen@}%
14029             {\glstreenamfmt{\glentryname{\@glo@label}}}%
14030             \ifdim\dimen@>\gls@tmplen
14031                 \gls@tmplen=\dimen@
14032                 \eglssetwidest{\glentryname{\@glo@label}}%
14033             \fi
14034             \settowidth{\dimen@}%
14035             {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14036             \ifdim\dimen@>#2\relax
14037                 #2=\dimen@
14038             \fi
14039         }%
14040     }%
14041 }

```

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`.

```

14042 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
14043     \glstreeindent=\glstxtrtreetopindent\relax
14044 }

```

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.

```
14045 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14046 \ifcsundef{@glswidestname\romannumeral#1}%
14047 {%
14048 \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
14049 }%
14050 {%
14051 \settowidth{#3}{\glstreenamfmt{
14052 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14053 }%
14054 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14055 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14056 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14057 \renewglossarystyle{alttree}{%
14058 \renewenvironment{theglossary}%
14059 {%
14060 \glxstralttreeInit
14061 \def\@gls@prevlevel{-1}%
14062 \mbox{}\par}%
14063 {\par}%
14064 \renewcommand*{\glossaryheader}{}%
14065 \renewcommand*{\glsgroupheading}[1]{}%
14066 \renewcommand{\glossentry}[2]{%
14067 \ifnum\@gls@prevlevel=0\relax
14068 \else
14069 \glxstrComputeTreeIndent{##1}%
14070 \fi
14071 \parindent\glstreeindent
14072 \glxstrAltTreeSetHangIndent
14073 \makebox[0pt][r]%
14074 {%
14075 \glstreenamebox{\glstreeindent}%
14076 {%
14077 \glsentryitem{##1}%
14078 \glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
14079 }%
14080 }%
```

```

14081      \glxtralttreeSymbolDescLocation{##1}{##2}%
14082      \def\@gls@prevlevel{0}%
14083  }
14084  \renewcommand{\subglossentry}[3]{%
14085      \ifnum##1=1\relax
14086          \glssubentryitem{##2}%
14087      \fi
14088      \ifnum\@gls@prevlevel=##1\relax
14089      \else
14090          \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14091          \ifnum\@gls@prevlevel<##1\relax
14092              \setlength\glstreeindent\gls@tmplen
14093              \addtolength\glstreeindent\parindent
14094              \parindent\glstreeindent
14095          \else
14096              \ifnum\@gls@prevlevel=0\relax
14097                  \glxtrComputeTreeIndent{##2}%
14098              \else
14099                  \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14100              \fi
14101              \addtolength\parindent{-\glstreeindent}%
14102              \setlength\glstreeindent\parindent
14103          \fi
14104      \fi
14105      \glxtrAltTreeSetSubHangIndent{##1}%
14106      \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14107          \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14108      \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14109      \def\@gls@prevlevel{##1}%
14110  }%
14111  \renewcommand*\@glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
14112 }
14113 }%
14114 {%
14115 }

```

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.

```

14116 \ifdef{\@glsstyle@alttreegroup}
14117 {%
14118     \renewglossarystyle{alttreegroup}{%
14119         \setglossarystyle{alttree}%
14120         \renewcommand{\glsgroupheading}[1]{\par
14121             \def\@gls@prevlevel{-1}%
14122             \hangindent0pt\relax
14123             \parindent0pt\relax
14124             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14125         \nopagebreak\indexspace\nopagebreak
14126     }%
14127 }%

```



```

14128 }%
14129 {%
14130 }

```

Similarly for `alttreehypergroup`.

```

14131 \ifdef{\@glsstyle@alttreehypergroup}
14132 {%
14133   \renewglossarystyle{alttreehypergroup}{%
14134     \setglossarystyle{alttree}%
14135     \renewcommand*{\glossaryheader}{%
14136       \par
14137       \def\@gls@prevlevel{-1}%
14138       \hangindent0pt\relax
14139       \parindent0pt\relax
14140       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14141     }%
14142     \renewcommand*{\glsgroupheading}[1]{%
14143       \par
14144       \def\@gls@prevlevel{-1}%
14145       \hangindent0pt\relax
14146       \parindent0pt\relax
14147       \glstreegroupheaderfmt
14148       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14149       \nopagebreak\indexspace\nopagebreak
14150     }%
14151   }
14152 }%
14153 {%
14154 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14155 \ifdef{\@glsstyle@mcolindexgroup}
14156 {%
14157   \renewglossarystyle{mcolindexgroup}{%
14158     \setglossarystyle{mcolindex}%
14159     \renewcommand*{\glsgroupheading}[1]{%
14160       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14161       \nopagebreak\indexspace\nobreak\@afterheading
14162     }%
14163   }
14164 }%
14165 {%
14166 }

```

Similarly for `mcolindexhypergroup`.

```

14167 \ifdef{\@glsstyle@mcolindexhypergroup}
14168 {%

```

```

14169 \renewglossarystyle{mcolindexhypergroup}{%
14170   \setglossarystyle{mcolindex}%
14171   \renewcommand*{\glossaryheader}{%
14172     \item\glstreenavigationfmt{\glsnavigation}%
14173     \indexspace
14174   }%
14175   \renewcommand*{\glsgroupheading}[1]{%
14176     \item\glstreegroupheaderfmt
14177       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14178     \nopagebreak\indexspace\nobreak\@afterheading
14179   }%
14180 }
14181 }%
14182 {%
14183 }

```

Similarly for mcolindexspannav.

```

14184 \ifdef{\@glsstyle@mcolindexspannav}
14185 {%
14186   \renewglossarystyle{mcolindexspannav}{%
14187     \setglossarystyle{index}%
14188     \renewenvironment{theglossary}%
14189     {%
14190       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14191       \setlength{\parindent}{0pt}%
14192       \setlength{\parskip}{0pt plus 0.3pt}%
14193       \let\item\glstreeitem}%
14194     {\end{multicols}}}%
14195     \renewcommand*{\glsgroupheading}[1]{%
14196       \item\glstreegroupheaderfmt
14197         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14198       \nopagebreak\indexspace\nobreak\@afterheading
14199     }%
14200   }
14201 }%
14202 {%
14203 }

```

Similarly for mcoltreegroup.

```

14204 \ifdef{\@glsstyle@mcoltreegroup}
14205 {%
14206   \renewglossarystyle{mcoltreegroup}{%
14207     \setglossarystyle{mcoltree}%
14208     \renewcommand{\glsgroupheading}[1]{\par
14209       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14210     \nopagebreak\indexspace\nobreak\@afterheading
14211   }%
14212 }
14213 }%
14214 {%

```

14215 }

Similarly for mcoltreehypergroup.

```
14216 \ifdef{\@glsstyle@mcoltreehypergroup}
14217 {%
14218   \renewglossarystyle{mcoltreehypergroup}{%
14219     \setglossarystyle{mcoltree}%
14220     \renewcommand*{\glossaryheader}{%
14221       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14222     }%
14223     \renewcommand*{\glsgroupheading}[1]{%
14224       \par\noindent
14225       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14226       \nopagebreak\indexspace\nobreak\@afterheading
14227     }%
14228   }
14229 }%
14230 {%
14231 }
```

Similarly for mcoltreespannav.

```
14232 \ifdef{\@glsstyle@mcoltreespannav}
14233 {%
14234   \renewglossarystyle{mcoltreespannav}{%
14235     \setglossarystyle{tree}%
14236     \renewenvironment{theglossary}%
14237     {%
14238       \begin{multicols}{\glsmcols}%
14239       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14240       \setlength{\parindent}{0pt}%
14241       \setlength{\parskip}{0pt plus 0.3pt}%
14242     }%
14243     {\end{multicols}}%
14244     \renewcommand*{\glsgroupheading}[1]{%
14245       \par\noindent
14246       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14247       \nopagebreak\indexspace\nobreak\@afterheading
14248     }%
14249   }
14250 }%
14251 {%
14252 }
```

Similarly for mcoltreenonamegroup.

```
14253 \ifdef{\@glsstyle@mcoltreenonamegroup}
14254 {%
14255   \renewglossarystyle{mcoltreenonamegroup}{%
14256     \setglossarystyle{mcoltreenoname}%
14257     \renewcommand{\glsgroupheading}[1]{\par
14258       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14259     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14260 }%
14261 }
14262 }%
14263 {%
14264 }

```

Similarly for mcoltreenonamehypergroup.

```

14265 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14266 {%
14267   \renewglossarystyle{mcoltreenonamehypergroup}{%
14268     \setglossarystyle{mcoltreenoname}%
14269     \renewcommand*{\glossaryheader}{%
14270       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14271     \renewcommand*{\glsgroupheading}[1]{%
14272       \par\noindent
14273       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14274       \nopagebreak\indexspace\nobreak\@afterheading}%
14275   }
14276 }%
14277 {%
14278 }

```

Similarly for mcoltreenonamespannav.

```

14279 \ifdef{\@glsstyle@mcoltreenonamespannav}
14280 {%
14281   \renewglossarystyle{mcoltreenonamespannav}{%
14282     \setglossarystyle{treenoname}%
14283     \renewenvironment{theglossary}%
14284     {%
14285       \begin{multicols}{\glsmcols}%
14286       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14287       \setlength{\parindent}{0pt}%
14288       \setlength{\parskip}{0pt plus 0.3pt}%
14289     }%
14290     {\end{multicols}}}%
14291   \renewcommand*{\glsgroupheading}[1]{%
14292     \par\noindent
14293     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14294     \nopagebreak\indexspace\nobreak\@afterheading}%
14295   }
14296 }%
14297 {%
14298 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{}\par which would unbalance the top of the columns.

```

14299 \ifdef{\@glsstyle@mcolalttree}
14300 {%
14301   \renewglossarystyle{mcolalttree}{%
14302     \setglossarystyle{alttree}%
14303     \renewenvironment{theglossary}%

```

```

14304   {%
14305       \glstralttreeInit
14306       \def\@gls@prevlevel{-1}%
14307       \begin{multicols}{\glsmcols}%
14308   }%
14309   {\par\end{multicols}}%
14310 }
14311 }%
14312 {%
14313 }

```

Redefine mcolalttreegroup to discourage page breaks after the group headings.

```

14314 \ifdef{\@glsstyle@mcolalttreegroup}
14315 {%
14316   \renewglossarystyle{mcolalttreegroup}{%
14317       \setglossarystyle{mcolalttree}%
14318       \renewcommand{\glsgroupheading}[1]{\par
14319           \def\@gls@prevlevel{-1}%
14320           \hangindent0pt\relax
14321           \parindent0pt\relax
14322           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14323       \nopagebreak\indexspace\nopagebreak
14324   }%
14325 }
14326 }%
14327 {%
14328 }

```

Similarly for mcolalttreehypergroup.

```

14329 \ifdef{\@glsstyle@mcolalttreehypergroup}
14330 {%
14331   \renewglossarystyle{mcolalttreehypergroup}{%
14332       \setglossarystyle{mcolalttree}%
14333       \renewcommand*\glossaryheader{%
14334           \par
14335           \def\@gls@prevlevel{-1}%
14336           \hangindent0pt\relax
14337           \parindent0pt\relax
14338           \glstreenavigationfmt{\glsnavigation}%
14339           \par\indexspace
14340       }%
14341       \renewcommand*\glsgroupheading[1]{%
14342           \par
14343           \def\@gls@prevlevel{-1}%
14344           \hangindent0pt\relax
14345           \parindent0pt\relax
14346           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14347           \nopagebreak\indexspace\nopagebreak
14348       }%
14349   }

```

```

14350 }%
14351 {%
14352 }

```

Similarly for mcolalittreespannav.

```

14353 \ifdef{\@glsstyle@mcolalittreespannav}
14354 {%
14355   \renewglossarystyle{mcolalittreespannav}{%
14356     \setglossarystyle{almtree}%
14357     \renewenvironment{theglossary}%
14358     {%
14359       \glsextralittreeInit
14360       \def\@gls@prevlevel{-1}%
14361       \begin{multicols}{\@gls@col}%
14362         [\noindent\glstreenavigationfmt{\@gls@navigation}]%
14363     }%
14364     {\par\end{multicols}}%
14365     \renewcommand*\@gls@groupheading[1]{%
14366       \par
14367       \def\@gls@prevlevel{-1}%
14368       \hangindent0pt\relax
14369       \parindent0pt\relax
14370       \glstreegroupheaderfmt{\@gls@navhypertarget{##1}{\@gls@getgrouptitle{##1}}}%
14371       \nopagebreak\indexspace\nopagebreak
14372     }%
14373   }
14374 }%
14375 {%
14376 }

```

Reset the default style

```

14377 \ifx\@glossary@default@style\relax
14378 \else
14379   \setglossarystyle{\@gls@current@style}
14380 \fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
14381 \NeedsTeXFormat{LaTeX2e}
14382 \ProvidesPackage{glossary-bookindex}[2018/05/09 v1.31 (NLCT)]
```

Load required packages.

```
14383 \RequirePackage{multicol}
14384 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14385 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14386 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14387 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14388 \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.

```
14389 \newcommand*{\glstrbookindexprelocation}[1]{%
14390   \glstrifhasfield{location}{#1}%
14391   {\glstrprelocation}%
14392   {\glstrprelocation}%
14393 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14394 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14395   \glstrbookindexprelocation{#1}%
14396 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14397 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14398 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

bookindexbetween Between two top-level entries identified by the labels in the arguments.

```
14399 \newcommand{\glstrbookindexbetween}[2]{}
```

indexsubbetween Between two level 1 entries identified by the labels in the arguments.

```
14400 \newcommand{\glstrbookindexsubbetween}[2]{}
```

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.

```
14401 \newcommand{\glstrbookindexsubsubbetween}[2]{}
```

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.

```
14402 \newcommand{\glstrbookindexatendgroup}[1]{}
```

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.

```
14403 \newcommand{\glstrbookindexsubatendgroup}[1]{}
```

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.

```
14404 \newcommand{\glstrbookindexsubsubatendgroup}[1]{}
```

kindexgroupskip Group separator.

```
14405 \newcommand{\glstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

indexformatheader Group separator.

```
14406 \newcommand*{\glstrbookindexformatheader}[1]{%
14407 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14408 }
```

bookindexbookmark Book mark group heading if supported.

```
14409 \ifdef\pdfbookmark
14410 {%
14411 \newcommand*{\glstrbookindexbookmark}[2]{%
14412 \ifdefstring{\@@glossarysec}{chapter}%
14413 {\pdfbookmark[1]{#1}{#2}}%
14414 {\pdfbookmark[2]{#1}{#2}}%
14415 }
14416 }
14417 {%
14418 \newcommand*{\glstrbookindexbookmark}[2]{%
14419 }
```

kindexcolspread

```
14420 \newcommand*{\glstrbookindexcolspread}{}%
```

indexmulticolenv

```
14421 \newcommand*{\glstrbookindexmulticolenv}{multicols}
```


Define the style.

```
14422 \newglossarystyle{bookindex}{%
14423   \setglossarystyle{index}%
14424   \renewenvironment{theglossary}%
14425   {%
14426     \ifdefempty{glstrbookindexcolspread}
14427     {%
14428       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14429       {\glstrbookindexcols}%
14430     }%
14431     {%
14432       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14433       {\glstrbookindexcols}[\glstrbookindexcolspread]%
14434     }%
14435     \setlength{\parindent}{0pt}%
14436     \setlength{\parskip}{0pt plus 0.3pt}%
14437     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14438     \let\@glstrbookindex@subsep\glstrbookindexparentsubchildsep
14439     \let\@glstrbookindex@between\@gobble
14440     \let\@glstrbookindex@subbetween\@gobble
14441     \let\@glstrbookindex@subsubbetween\@gobble
14442     \let\@glstrbookindex@atendgroup\relax
14443     \let\@glstrbookindex@subatendgroup\relax
14444     \let\@glstrbookindex@subsubatendgroup\relax
14445     \let\@glstrbookindex@groupskip\relax
14446   }%
14447   {%
```

Do end group hooks.

```
14448     \@glstrbookindex@subsubatendgroup
14449     \@glstrbookindex@subatendgroup
14450     \@glstrbookindex@atendgroup
```

End multicol environment.

```
14451     \expandafter\end\expandafter{\glstrbookindexmulticolseenv}%
14452   }%
```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```
14453   \renewcommand*{\glossaryheader}{\raggedright}%
```

Top level entry format.

```
14454   \renewcommand*{\glossentry}[2]{%
```

Do separator.

```
14455     \@glstrbookindex@between{##1}%
```

Update separators.

```
14456     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14457     \let\@glstrbookindex@subsep\glstrbookindexparentsubchildsep
14458     \let\@glstrbookindex@subbetween\@gobble
14459     \let\@glstrbookindex@subsubbetween\@gobble
14460     \edef\@glstrbookindex@between{%
```

```

14461      \noexpand\glxstrbookindexbetween{##1}%
14462  }%
14463  \edef\@glxstr@bookindex@atendgroup{%
14464      \noexpand\glxstrbookindexatendgroup{##1}%
14465  }%
14466  \let\@glxstr@bookindex@subatendgroup\relax
14467  \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14468  \glstreeitem
14469      \glstryitem{##1}%
14470      \glstarget{##1}{\glxstrbookindexname{##1}}%
14471  \glxstrbookindexprelocation{##1}##2%
14472  }%
14473  \renewcommand{\subglossentry}[3]{%
14474      \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14475      \glstreeitem
14476      \or

```

Level 1.

```

14477      \@glxstr@bookindex@sep
14478      \@glxstr@bookindex@subbetween{##2}%
14479      \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14480      \let\@glxstr@bookindex@subsubbetween\@gobble
14481      \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14482      \edef\@glxstr@bookindex@subbetween{%
14483          \noexpand\glxstrbookindexsubbetween{##2}%
14484      }%
14485      \edef\@glxstr@bookindex@atsubendgroup{%
14486          \noexpand\glxstrbookindexatsubendgroup{##1}%
14487      }%

```

Start sub-item.

```

14488      \glstreesubitem
14489      \glssubentryitem{##2}%
14490      \else

```

All other levels.

```

14491      \@glxstr@bookindex@subsep
14492      \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14493      \let\@glxstr@bookindex@subsep\relax
14494      \edef\@glxstr@bookindex@subsubbetween{%
14495          \noexpand\glxstrbookindexsubsubbetween{##2}%
14496      }%
14497      \edef\@glxstr@bookindex@atsubsubendgroup{%
14498          \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14499      }%

```

Start sub-sub-item.

```
14500 \glstreesubsubitem
14501 \fi
```

Format entry.

```
14502 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14503 \glxtrbookindexsubprelocation{##2}##3%
14504 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14505 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14506 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14507 \@glxtr@bookindex@subsubatendgroup
14508 \@glxtr@bookindex@subatendgroup
14509 \@glxtr@bookindex@atendgroup
14510 \@glxtr@bookindex@groupskip
```

Update separators.

```
14511 \let\@glxtr@bookindex@groupskip\glxtrbookindexgroupskip
14512 \let\@glxtr@bookindex@between\@gobble
14513 \let\@glxtr@bookindex@atendgroup\relax
14514 \let\@glxtr@bookindex@subatendgroup\relax
14515 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14516 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14517 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14518 \glxtrbookindexformatheader{\thisgrptitle}%
14519 \nopagebreak\indexspace\nopagebreak\@afterheading
14520 }%
14521 }
```

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.

```
14522 \newcommand{\glxtrbookindexthepage}{%
14523 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14524 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14525 \newcommand*{\glxtrbookindexmarkentry}[1]{%
14526   \protected@write\@auxout
14527   {\let\glxtrbookindexthepage\relax}%
14528   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14529 }
```

etbookindexmark

```
14530 \newcommand*{\glxtr@setbookindexmark}[2]{%
14531   \ifcsundef{glxtr@idxfirstmark@#1}%
14532   {\csgdef{glxtr@idxfirstmark@#1}{#2}}%
14533   {}%
14534   \csgdef{glxtr@idxlastmark@#1}{#2}%
14535 }
```

indexfirstmarkfmt

```
14536 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
14537   \glsentryname{#1}%
14538 }
```

indexfirstmark

```
14539 \newcommand*{\glxtrbookindexfirstmark}{%
14540   \letcs{\glxtr@label}{glxtr@idxfirstmark@\glxtrbookindexthepage}%
14541   \ifdef\glxtr@label
14542   {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%
14543   {}%
14544 }
```

indexlastmarkfmt

```
14545 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
14546   \glsentryname{#1}%
14547 }
```

okindexlastmark

```
14548 \newcommand*{\glxtrbookindexlastmark}{%
14549   \letcs{\glxtr@label}{glxtr@idxlastmark@\glxtrbookindexthepage}%
14550   \ifdef\glxtr@label
14551   {\glxtrbookindexlastmarkfmt{\glxtr@label}}%
14552   {}%
14553 }
```

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)			
General: Initial experimental release	5	
0.2 (2015-11-30)			
\Glsfmtshort: new	320	
\glsfmtshort: new	319	
\Glsfmtshortpl: new	320	
\glsfmtshortpl: new	320	
short: switched inline full form to short			
(long)	223	
0.3 (2015-12-02)			
\@ACRlong: added redefinition	76	
\@ACRlongpl: added redefinition	77	
\@ACRshort: added redefinition	74	
\@ACRshortpl: added redefinition	75	
\@Acrlong: added redefinition	75	
\@Acrlongpl: added redefinition	76	
\@Acrshort: added redefinition	73	
\@Acrshortpl: added redefinition	74	
\@GLSdesc: added redefinition	69	
\@GLSdescplural: added redefinition	.	70	
\@GLSfirst: added redefinition	67	
\@GLSfirstplural: added redefinition		68	
\@GLSname: added redefinition	69	
\@GLSplural: added redefinition	68	
\@GLSsymbol: added redefinition	70	
\@GLSsymbolplural: added			
redefinition	71	
\@GLStext: added redefinition	66	
\@GLSuseri: added redefinition	71	
\@GLSuserii: added redefinition	72	
\@GLSuseriii: added redefinition	72	
\@GLSuseriv: added redefinition	72	
\@GLSuseriv: added redefinition	72	
\@GLSuseriv: added redefinition	72	
\@GLSuseriv: added redefinition	73	
\@GLSdesc: added redefinition	69	
\@GLSdescplural: added redefinition	.	70	
\@GLSfirst: added redefinition	67	
\@GLSfirstplural: added redefinition		68	
\@GLSname: added redefinition	69	
\@GLSplural: added redefinition	67	
	\@GLSsymbol: added redefinition	70
	\@GLSsymbolplural: added		
	redefinition	71
	\@GLStext: added redefinition	66
	\@GLSuseri: added redefinition	71
	\@GLSuserii: added redefinition	71
	\@GLSuseriii: added redefinition	72
	\@GLSuseriv: added redefinition	72
	\@GLSuseriv: added redefinition	72
	\@GLSuseriv: added redefinition	72
	\@GLSuseriv: added redefinition	73
	\@GLSdesc: added redefinition	69
	\@GLSdescplural: added redefinition	.	70
	\@GLSfirst: added redefinition	67
	\@GLSfirstplural: added redefinition		68
	\@GLSname: added redefinition	69
	\@GLSplural: added redefinition	67
	\@GLSsymbolplural: added		
	redefinition	70
	\@GLSxtr@defaultnoglossarywarning:		
	new	129
	\@GLSxtr@field@linkdefs: new	65
	\@GLSxtr@insertdots: new	191
	\@print@glossary: added redefinition		126
	\glsabbrvdefaultfont: renamed from		
	\abbrvdefaultfont	196
	\glsaccessdesc: new	154
	\glsaccessdescplural: new	155
	\glsaccessfirst: new	152
	\glsaccessfirstplural: new	153
	\glsaccesslong: new	157
	\glsaccesslong: new	157
	\glsaccessname: new	150
	\glsaccessplural: new	151
	\glsaccessshort: new	156
	\glsaccessshort: new	156
	\glsaccessshortpl: new	156

\glsaccesssshortpl: new	156	\cGLSpl: new	102
\glsaccessssymbol: new	153	\cGLSpl@: new	102
\glsaccessssymbolplural: new	154	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	151	new	97
\glstentryfmt: added check for short ..	58	\cGLS: new	102
\glslongpltok: new	191	\cGLSformat: new	102
\glsshortpltok: new	191	\cGLSpl: new	102
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	103
name in \setkeys	192	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	16
for plural	188	\glsenableentrycount: new	98
\GLSxtrlongpl: new	207	\glsfirstabbrvdefaultfont: new ..	196
\Glsxtrlongpl: new	206	\glsfirstlongdefaultfont: new ...	196
\glsxtrlongpl: new	206	\Glsfmtfirst: new	323
\glsxtrNoGlossaryWarning: new	21	\glsfmtfirst: new	322
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	323
new	187	\glsfmtfirstpl: new	323
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	322
new	187	\glsfmtplural: new	322
\glsxtrpostlinkendsentence: new ..	187	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	205	\Glsxtrtitleshort	320
\Glsxtrshortpl: new	204	renamed from \glstentryfmtshort ..	320
\glsxtrshortpl: new	204	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	319
\glslabeltok	218	renamed from \glstentryfmtshort ..	319
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	215	\Glsxtrtitleshortpl	320
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	320
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	320
\Glsxtrshort	320	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	320
\glsxtrshort	319	\Glsfmttext: new	322
\Glsfmtshortpl: changed to use		\glsfmttext: new	321
\glsxtrshortpl	320	\glshasattribute: new	165
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	165
\glsxtrshortpl	320	\glsxtremsuffix: new	259
\glsxtrifemptyglossary: new	27	\GlsXtrEnableEntryCounting: new ..	97
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	100
argument	169	\glsxtrscfont: new	230
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	231
argument	168	\glsxtrsmfont: new	245
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	245
default type to \acronymtype	111	short-em: new	266
\newterm: fixed name argument	168	short-em-desc: new	268
0.5 (2015-12-07)		short-em-footnote: new	277
\cGLS: new	102	short-em-long: new	263
\cGLS@: new	102	short-em-long-desc: new	264

short-em-postfootnote: new	278	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	241	attribute	311
short-sc-postfootnote: new	243	\Glsxtrheadtext: now uses headuc	
short-sm: new	249	attribute	313
short-sm-desc: new	250	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	255	attribute	313
short-sm-long: new	247	short-em-footnote: switch off regular	
short-sm-long-desc: new	248	attribute if set	277
short-sm-postfootnote: new	257	short-long: switch off regular attribute	
long-noshort-em: new	270	if set	216
long-noshort-em-desc: new	273	short-long-desc: switch off regular	
long-noshort-sm: new	252	attribute if set	218
long-noshort-sm-desc: new	254	short-sc-footnote: switch off regular	
long-short-em: new	259	attribute if set	241
long-short-em-desc: new	260	short-sm-footnote: switch off regular	
long-short-sm: new	245	attribute if set	256
long-short-sm-desc: new	246	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	214
\Glsaccessstext: new	151	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	216
\@gls@setup@default@short@access:		long-short-sc-desc: switch off regular	
removed \ifglxtruseuchead ...	310	attribute if set	232
\@glxtr@doaccsupp: new	21	footnote: switch off regular attribute if	
\Glsaccessdesc: new	155	set	219
\Glsaccessdescplural: new	155	postfootnote: switch off regular	
\Glsaccessfirst: new	152	attribute if set	221
\Glsaccessfirstplural: new	153	0.5.2 (2015-12-08)	
\Glsaccessname: new	151	\@GLSdesc@: added accessibility support	69
\Glsaccessplural: new	152	\@GLSdescplural@: added accessibility	
\Glsaccesssymbol: new	153	support	70
\Glsaccesssymbolplural: new	154	\@GLSfirst@: added accessibility	
\Glsxtrheadfirst: now uses headuc		support	67
attribute	315	\@GLSfirstplural@: added accessibility	
\glxtrheadfirst: now uses headuc		support	68
attribute	314	\@GLSname@: added accessibility support	69
\Glsxtrheadfirstplural: now uses		\@GLSplural@: added accessibility	
headuc attribute	316	support	68
\glxtrheadfirstplural: now uses		\@GLSsymbol@: added accessibility	
headuc attribute	315	support	70
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute	314	accessibility support	71
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	66
attribute	314	\@GLSdesc@: added accessibility support	69
\Glsxtrheadshort: now uses headuc		\@GLSdescplural@: added accessibility	
attribute	311	support	70
\glxtrheadshort: now uses headuc		\@GLSfirst@: added accessibility	
attribute	310	support	67
\Glsxtrheadshortpl: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	312	support	68

\@Glsname@: add accessibility support ..	69	\glsdohypertarget: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	150
\@Glsplural@: added accessibility support	67	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here	58
\@Glsymbol@: added accessibility support	70	\GlsXtrEnableInitialTagging: new	183
\@Glsymbolplural@: added accessibility support	71	\glsxtrfieldtitlecase: new	169
\@Glstext@: added accessibility support	66	\GlsXtrFormatLocationList: new ...	56
\@gls@setup@default@short@access: removed \glsxtrabbrvfmt	208	\glsxtrnewabbrevpresetkeyhook: new	194
\@glsdesc@: added accessibility support	69	\glsxtrtagfont: new	184
\@glsdescplural@: added accessibility support	69	\KV@printgloss@nonumberlist: added	57
\@glsfirst@: added accessibility support	66	\mfu@checkword@do: added	184
\@glsfirstplural@: added accessibility support	68	\setabbreviationstyle: added check for post-definition style switch	211
\@Glsname@: added accessibility support	69	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	67	\@glsxtr@autoindex@at: new	180
\@Glsymbol@: added accessibility support	70	\@glsxtr@autoindex@encap: new ...	180
\@Glsymbolplural@: added accessibility support	70	\@glsxtr@autoindex@esc: new	181
\@Glstext@: added accessibility support	66	\@glsxtr@autoindex@level: new ...	181
\@glsxtr@activate@initialtagging: new	184	\@glsxtr@autoindex@setname: new .	179
\@glsxtr@do@titlecaps@warn: new .	184	\@glsxtr@doabbreviationsdef: new .	17
\@glsxtr@tag: new	184	\glsdescwidth: added	55
\glossaryentrynumbers: added	55	\glsdohypertarget: removed \GlsXtrNoGlsWarningNoAutoMakeMain	128
\Glossentrydesc: added	182	\glspagelistwidth: added	55
\Glossentryname: added	174	\glsxtrdoautoindexname: new	178
\Glossentrysymbol: added	183	\glsxtrpostnamehook: new	175
\glossentrysymbol: added	182	\if@glsxtr@format@override: new .	177
\GLSaccessdesc: new	155, 162	\ProvidesGlossariesExtraLang: new	326
\GLSaccessdescplural: new ...	155, 163	\RequireGlossariesExtraLang: new	326
\GLSaccessfirst: new	152, 161	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	153, 162	\@newglossaryentry@defunitcounters: new	103
\GLSaccesslong: new	157, 163	\@GLSxtr@p@acrlong@: new	89
\GLSaccesslongpl: new	157, 164	\@GLSxtr@p@acrlongpl@: new	89
\Glsaccesslongpl: new	157	\@GLSxtr@p@acrshort@: new	88
\glsaccesslongpl: new	157	\@GLSxtr@p@acrshortpl@: new	88
\GLSaccessname: new	151, 161	\@GLSxtr@p@long@: new	88
\GLSaccessplural: new	152, 161	\@GLSxtr@p@longpl@: new	88
\GLSaccessshort: new	156, 163	\@GLSxtr@p@plural@: new	87
\GLSaccessshortpl: new	157, 163	\@GLSxtr@p@short@: new	87
\GLSaccesssymbol: new	154, 162	\@GLSxtr@p@shortpl@: new	88
\GLSaccesssymbolplural: new .	154, 162	\@GLSxtr@p@text@: new	86
\GLSaccessstext: new	151, 161	\@GlsXtrEnableOnTheFly: new	51
		\@Glsxtr: new	52
		\@Glsxtr@p@acrlong@: new	89

\@Glsxtr@p@acrlongpl@: new	89	\glstdonohyperlink: added	85
\@Glsxtr@p@acrshort@: new	88	\glsenableentryunitcount: new	105
\@Glsxtr@p@acrshortpl@: new	88	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	88	entry's existence	165
\@Glsxtr@p@longpl@: new	88	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	87	entry's existence	166
\@Glsxtr@p@short@: new	87	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	87	check	186
\@Glsxtr@p@text@: new	86	\Glsxtr: new	52
\@Glsxtrpl: new	53	\glxtr: new	51
\@alt@glshyp@opt: new	83	\glxtrcat: new	51
\@gl@alt@hyp@opt: new	82	\glxtrdowrglossaryhook: new	82
\@gl@alt@hyp@opt@char: new	83	\GlsXtrEnableEntryUnitCounting:	
\@gl@alt@hyp@opt@keys: new	83	new	109
\@gl@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	50
new	104	\Glsxtrpl: new	52
\@gl@local@increment@currunitcount:		\glxtrpl: new	52
new	105	\glxtrpostlocalreset: new	96
\@gl@setdefault@glslink@opts:		\glxtrpostlocalunset: new	96
new	80	\glxtrpostreset: new	96
\@glxtr: new	51	\glxtrpostunset: new	95
\@glxtr@addunitcounter: new	104	\glxtrprotectlinks: new	86
\@glxtr@currunitcount: new	105	\GlsXtrSetAltModifier: new	83
\@glxtr@ifunitcounter: new	104	\GlsXtrSetDefaultGlsOpts: new	81
\@glxtr@p@acrlong@: new	89	\glxtrstarflywarn: new	51
\@glxtr@p@acrlongpl@: new	89	\GlsXtrWarning: new	53
\@glxtr@p@acrshort@: new	88	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new	88	disables \setacronymstyle	111
\@glxtr@p@long@: new	88	1.0 (2016-01-24)	
\@glxtr@p@longpl@: new	88	\@glxtr@autoindexcrossrefs: new	15
\@glxtr@p@plural@: new	87	\@glxtr@idx@displaynumberlist:	
\@glxtr@p@short@: new	87	new	119
\@glxtr@p@shortpl@: new	87	\@glxtr@idx@entrynumberlist: new	121
\@glxtr@p@text@: new	86	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new	105	new	119
\@glxtr@setentryunitcountunsetattr:		\@glxtr@noidx@entrynumberlist:	
new	109	new	121
\@glxtr@unitcountlist: new	104	\@glxtr@noidx@numberlistloop:	
\@glxtrpl: new	52	new	120
\@newglossaryentryposthook: added		\@glxtr@reg@glosslist: new	112
empty see value if not set and added		\makeglossaries: new	112
'see' to field key map	42	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	50	\glxtrdiscardperiod: added check	
\cGlsformat: added	103	for first use	188
\cGlsformat: added	103	short-desc: fixed typo in	
\cGlsplformat: added	103	\glxtrinlinefullformat and	
\cGlsplformat: added	103	added missing second argument	224
\glstdisablehyper: added	85	1.02 (2016-04-25)	
\glstdohyperlink: added	84	\@glxtr@current@style: new	54

\@glsfirst@: set abbreviation and regular format	66	\glsxtrregularfont: new	58
\@glsfirstplural@: set abbreviation and regular format	68	\glsxtruserfield: new	280
\@glsname@: set abbreviation and regular format	69	\glsxtruserparen: new	280
\@glsplural@: set abbreviation and regular format	67	\glsxtrusersuffix: new	281
\@glsymbol@: set regular format	70	\GlsXtrWarnDeprecatedAbbrStyle:	
\@glsymbolplural@: set regular format	70	new	213
\@glstext@: set abbreviation and regular format	66	short-em-long-em: new	264
\@glsxtr@deprecated@abbrstyle:		short-em-long-em-desc: new	266
new	213	short-em-nolong: new	267
\@glsxtr@do@style: new	22	short-em-nolong-desc: new	269
\@glsxtr@doloctag: new	57	short-em-postfootnote: renamed from “postfootnote-em”	278
\@glsxtr@idx@entrynumberlist:		short-footnote: new	220
switched from \let to \newcommand	121	short-long-user: new	288
\@glsxtr@pagetag: new	57	short-long-user-desc: new	289
\@glsxtr@pagetag: new	57	short-nolong: new	224
\@glsxtr@preloctag: new	57	short-nolong-desc: new	226
\@glsxtr@postloctag: new	57	short-postfootnote: new	222
\@glsxtr@preloctag: new	56, 57	short-sc-footnote: renamed from “footnote-sc”	241
\glossentrydesc: added glossdescfont attribute check	170	short-sc-nolong: new	236
\Glossentryname: added glossnamefont attribute check	174	short-sc-nolong-desc: new	237
\glossentryname: added glossnamefont attribute check	172	short-sc-postfootnote: renamed from “postfootnote-sc”	243
moved post name hook inside condition	174	short-sm-footnote: renamed from “footnote-sm”	255
\glsabbrvmfont: new	259	short-sm-nolong: new	250
\glsabbrvuserfont: new	281	short-sm-nolong-desc: new	251
\glsfirstabbrvmfont: new	259	short-sm-postfootnote: renamed from “postfootnote-sm”	257
\glsfirstabbrvuserfont: new	281	\letabbreviationstyle: new	212
\glsfirstlongemfont: new	259	\newabbreviationstyle: bug fix:	
\glsfirstlonguserfont: new	281	corrected test for existence	211
\glsifnotregularcategory: new ...	166	long-em-noshort-em: new	271
\glslongdefaultfont: new	196	long-em-noshort-em-desc: new	275
\glslongemfont: new	259	long-em-short-em: new	261
\glslongfont: new	196	long-em-short-em-desc: new	262
\glslonguserfont: new	281	long-noshort: new	230
\glsxtrassignfieldfont: new	65	long-noshort-desc: new	229
\GlsXtrEnablePreLocationTag: new .	56	long-noshort-em: renamed from “long-em”	270
\glsxtrfirstscfont: new	231	long-noshort-em-desc: renamed from “long-desc-em”	273
\glsxtrfirstsmfont: new	245	long-noshort-sc: renamed from “long-sc”	238
\glsxtrlongshortdescsort: new ...	215	long-noshort-sc-desc: renamed from “long-desc-sc”	240
\glsxtrpostnamehook: added category check	175	long-noshort-sm: renamed from “long-sm”	252

long-noshort-sm-desc: renamed from		General: disabled docdef key at the start	
\long-desc-sm	254	of the document	27
long-short-user: new	281	docdef option changed to choice	14
long-short-user-desc: new	287	\glstr@usesee: new	43
\renewabbreviationstyle: new	212	\glstrusesee: new	42
style: new	22	\glstruseseeformat: new	43
1.05 (2016-06-10)		\if@glstrdocdefrestricted: new ..	15
\eglssetwidest: new	382	1.07 (2016-08-15)	
\glFindWidestAnyName: new	384	@@glstrp: new	89
\glFindWidestAnyNameLocation:		\GLSfirst@: added check for	
new	390	nohyperfirst attribute	67
\glFindWidestAnyNameSymbol: new	387	\GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute	68
new	389	\GLSxtrp: new	90
\glFindWidestLevelTwo: new	386	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	384	nohyperfirst attribute	67
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new	389	nohyperfirst attribute	68
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new	90
new	387	\@glspreglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts	185
new	388	\@glfirst@: added check for	
\glFindWidestUsedLevelTwo: new ..	385	nohyperfirst attribute	67
\glFindWidestUsedTopLevelName:		\@glfirstplural@: added check for	
new	383	nohyperfirst attribute	68
\glfirstlongfootnotefont: new ..	218	\@glxtrinmark: new	308
\glsgetwidestname: new	383	\@glxtrnotinmark: new	308
\glsgetwidestsubname: new	383	\@glxtrp: new	90
\glslongfootnotefont: new	218	\@glxtrp@opt: new	89
\glxtrAltTreeIndent: new	381	\glossxtrsetpopts: new	89
\glxtralttreeInit: new	382	\glsp: new	92
\glxtrAltTreePar: new	381	\glsp: new	92
\glxtrAltTreeSetHangIndent: new	391	\glxtr@entry@p: new	91
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new	218
new	391	\glxtrchecknohyperfirst: new	66
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new	170
new	381	\glxtrifinmark: new	308
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new	93
new	381	\Glsxtrp: new	92
\glxtrComputeTreeIndent: new ...	390	\glxtrp: new	91
\glxtrComputeTreeSubIndent: new	391	\glxtrsetpopts: new	89
\glxtrtreetopindent: new	382	short-long-desc: added text key	218
short-em-long: fixed incorrect font used		fixed misspelling of \glabbrvfont in	
by long form	263	plural key	218
\xglsetwidest: new	382	long-short-desc: added missing text	
1.06 (2016-06-18)		key	216
\@glsoifexistsorwarn: new	15	fixed misspelling of \glabbrvfont ..	216
\@glxtr@docdefval: new	15	footnote: changed first forms to use	
\@glxtr@usesee: new	43	\glfirstlongfootnotefont ...	219

postfootnote: removed \footnote	\@printglossary: redefined to save
from first keys 220	options 118
switched from \glsfirstlongfont to	\glsxtr@makeglossaries: new 119
\glsfirstlongfootnotefont ... 222	1.10 (2016-12-17)
\RestoreAcronyms: modified	\@GLSp1@: fixed bug caused by typo in
\@gls@link@checkfirsthyper to	command name 60
set \glsxtrifwasfirstuse 111	1.11 (2017-01-19)
1.08 (2016-12-13)	\@glsxtr@do@redef@forglsentries:
\@glsxtr@record: new 8	new 6
\@GLS@: added \@glsxtr@record 59	\@glsxtr@noidx@do: new 140
\@GLSp1@: added \@glsxtr@record ... 60	\@glsxtr@redef@forglsentries: new . 6
\@Gls@: added \@glsxtr@record 59	\@glsxtr@shortcutsval: new 20
\@GLSp1@: added \@glsxtr@record ... 59	\@glsxtr@unsrt@getgrouptitle: new 139
\@Gls@: added \@glsxtr@record 59	\@print@noidx@glossary: added
\@gls@link@: added	redefinition 123
\@glsxtr@record 60	\glsxtr@addloclistfield: added
\@gls@field@link: added	group key 13
\@glsxtr@record 58	added location key 12
\@gls@saveentrycounter: new 27	\glsxtr@fields: new 131
\@glsdisp: added \@glsxtr@record .. 60	\glsxtr@linkprefix: new 132
\@GLSp1@: added \@glsxtr@record ... 59	\glsxtr@org@newignoredglossary:
\@glsxtr@dorecord: new 10	new 38
\@glsxtr@err@undefaction: new 6	\glsxtr@s@newignoredglossary: new 38
\@glsxtr@record: new 7	\glsxtr@shortcutsval: new 132
\@glsxtr@warn@onexistsordo: new ... 6	\glsxtr@texencoding: new 131
\@glsxtr@warn@undefaction: new 6	\glsxtr@writefields: new 132
\@print@unsrt@glossary: new 136	\GlsXtrLoadResources: new 131
General: added record package option ... 13	\glsxtrpageref: new 35
\glsadd: added \@glsxtr@record 64	\glsxtrresourcefile: changed
\glsdoifexists: now defines	extension to .gls tex 130
\glslabel 41	\newignoredglossary: added starred
\glsxtr@do@wrglossary: new 27	version 38
\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 133	definition 185
\glsxtr@resource: new 131	\@glsxtr@counterrecordhook: new . 134
\glsxtr@saveentrycounter: new 27	\@glsxtr@display@loc: new 123
\glsxtr@setup@record: new 12	\@glsxtr@docounterrecord: new ... 134
\glsxtrassignfieldfont: added check	\@glsxtr@longnewglossaryentry:
for existence 65	new 37
\glsxtrresourcefile: new 130	\@glsxtr@noop@recordcounter: new . 11
\printunsrtglossaries: new 136	\@glsxtr@op@recordcounter: new ... 11
\printunsrtglossary: new 136	\@glsxtr@provide@storagekey: new . 28
1.09 (2016-12-16)	\@glsxtr@s@longnewglossaryentry:
\@glsxtr@gettype: new 119	new 37
\@glsxtr@mixed@assign@sortkey:	\@glsxtrentryfmt: new 30
new 119	\@glsxtrindexaliased: new 81
	\@glsxtrsetaliasnoindex: new 80

\@newglossaryentryposthook: added check for alias key	47	\glxtrindexaliased: new	81
\@no@glxtrindexaliased: new	81	\GlsXtrLetField: new	34
\@printunsrtglossary: new	136	\GlsXtrLetFieldToField: new	34
General: added target key to printgloss family	118	\GlsXtrLoadResources: removed restriction on only one per document	131
\apptoglossarypreamble: new	36	\glxtrlocrangefmt: new	124
\csGlsXtrLetField: new	34	\glxtrpostlongdescription: new ..	37
\eGlsXtrSetField: new	34	\glxtrprovidestoragekey: new	28
\gGlsXtrSetField: new	34	\GlsXtrRecordCounter: new	134
\glsdohyperlink: added check for alias field	84	\glxtrresourcecount: new	131
\glsnoidxdisplayloc: added redefinition	123	\glxtrresourcefile: added catcode change for @	131
\glissettoctitle: added patch	39	\glxtrsetaliasnoindex: new	80
\glxtr@counterrecord: new	133	\GlsXtrSetField: new	34
\glxtr@langtag: new	132	\glxtrsetfieldifexists: new	33
\glxtr@newabbreviation: new	192	\glxtrunsrtdo: new	139
\glxtr@org@newignoredglossary: Added check for existence	38	\glxtrusefield: new	33
\glxtr@pluralsuffixes: new	132	\glxtrusefield: new	33
\glxtr@provideignoredglossary: new	39	short-postlong-user: new	285
\glxtr@s@newignoredglossary: Added check for existence	38	short-postlong-user-desc: new ...	287
\glxtr@s@provideignoredglossary: new	40	\longnewglossaryentry: added starred version	36
\glxtrabbrvpluralsuffix: new ...	197	long-postshort-user: new	283
\glxtralias: new	46	long-postshort-user-desc: new ...	284
\glxtrcopytoglossary: new	40	postdot: new	16
\glxtrdeffield: new	33	\pretoglossarypreamble: new	36
\glxtrdisplayendloc: new	124	\print@noop@unsrtglossaryunit: new	139
\glxtrdisplayendlochook: new ...	124	\print@op@unsrtglossaryunit: new	138
\glxtrdisplaysingleloc: new	124	\printunsrtglossary: added starred form	136
\glxtrdisplaystartloc: new	124	\printunsrtglossaryhandler: new .	138
\glxtrdeffield: new	33	\printunsrtglossaryunit: new	12
\glxtrentryfmt: new	30	\printunsrtglossaryunitsetup: new	138
\glxtrfielddolistloop: new	31	\provideignoredglossary: new	39
\glxtrfieldforlistloop: new	31	\s@glxtr@provide@storagekey: new	29
\glxtrfieldifinlist: new	31	\s@printunsrtglossary: new	136
\glxtrfieldlistadd: new	31	\xGlsXtrSetField: new	34
\glxtrfieldlistadd: new	31	1.13 (2017-02-07)	
\glxtrfieldlistgadd: new	31	\@glsdisp: removed	
\glxtrfieldlistxadd: new	31	\@glxtr@org@glsdisp	60
\glxtrfieldxifinlist: new	31	\glxtrsetaliasnoindex: switched to \providecommand	80
\glxtrfmt: new	29	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	29	\@gls@link: added redefinition	62
\GlsXtrFmtField: new	29	\@gls@noidx@getgrouptitle: new ..	121
\glxtrifkeydefined: new	28	\@gls@removespaces: new	125
		\@glxtr@do@automake@err: new ...	133
		\@glxtr@org@gloautosee: new	25

\@glxtr@record: added third arg	7	postfootnote: fixed spelling of	
\@glxtr@recordsee: new	11	\glsabbrvfont	221
General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	64	\@glo@autosee: added redefinition	26
added \glsadd option thevalue	64	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	85	bug	121
\glsenableentrycount: fixed		\@glxtr@addunusedxrefs: added	
assignment of \@cGls@	99	check for seealso field	47
\glsenableentryunitcount: fixed		\@glxtr@checkgroup: use \csuse	
assignment of \@cGls@	107	instead of \csname	140
\glsnavigation: new	122	\@glxtr@dorecordnodefer: new	11
\glxtr@org@getgrouptitle: new ..	122	\@print@unsrt@glossary: corrected	
\glxtr@recordsee: new	7	misspelt command	137
\glxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	133	new	138
\glxtrdisplayendloc: added check		General: added check for	
for empty format	124	\@gls@setupsort@none	14
\glxtrgetgrouptitle: new	122	\gls@checkseeallowed: added	
\glxtrinitwrgloss: new	60	redefinition	26
\glxtrlocationhyperlink: new ...	125	\glxtr@writefields: added	
\glxtrsetgrouptitle: new	122	\providecommand lines	132
\glxtrsupphypernumber: new	125	\glxtrautoindex: new	179
\ifglxtrwrglossbefore: new	61	\glxtrautoindexassignsort: new .	179
1.15 (2017-05-10)		\glxtrautoindexentry: new	179
\@glxtr@dorecord: corrected		\glxtrindexseealso: new	44
premature expansion of \@glslocref	10	\glxtrseealso: new	46
short-em-long-em: fixed spelling of		\glxtrseelist: new	44
\glsabbrvfont	265	\glxtruseseealso: new	43
short-long: fixed spelling of		\glxtruseseealsoformat: new	44
\glsabbrvfont	216	\seealso: new	44
short-long-user: fixed spelling of		autoseeindex: new	16
\glsabbrvfont	288	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@gls@setup@default@short@access:	
\glsabbrvfont	285	removed some inconsistencies in the	
short-postlong-user-desc: fixed		abbreviation styles	214
spelling of \glsabbrvfont	287	\@glxtr@mark@wordseps: new	192
long-em-short-em: fixed spelling of		\@glxtr@markwordseps: new	191
\glsabbrvfont	261	\@glxtr@noidx@displaynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	283	\glxtrundeftag	120
long-postshort-user-desc: fixed		\@glxtr@noidx@entrynumberlist:	
spelling of \glsabbrvfont	284	replace hard-coded ?? with	
long-short: fixed spelling of		\glxtrundeftag	121
\glsabbrvfont	214	\@glxtr@noidx@numberlistloop:	
long-short-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	282	\glxtrundeftag	120
footnote: fixed spelling of		\@glxtrifhyphenstart: new	290
\glsabbrvfont	219	\glsabbrvhyphenfont: new	291
		\glsabbrvonlyfont: new	304

<code>\glsabbrvscfont: new</code>	231	<code>short-hyphen-postlong-hyphen-desc:</code>	
<code>\glsabbrvsmfont: new</code>	245	<code>new</code>	303
<code>\glsabbrvuserfont: initialised to</code>		<code>short-long-user-desc: corrected first</code>	
<code>default font</code>	281	<code>forms</code>	289
<code>\glsfirstabbrvhyphenfont: new</code>	291	<code>short-nolong-desc-noreg: new</code>	226
<code>\glsfirstabbrvonlyfont: new</code>	304	<code>short-nolong-noreg: new</code>	224
<code>\glsfirstabbrvscfont: new</code>	231	<code>long-em-noshort-em-desc-noreg:</code>	
<code>\glsfirstabbrvsmfont: new</code>	245	<code>new</code>	276
<code>\glsfirstlonghyphenfont: new</code>	291	<code>long-em-noshort-em-noreg: new</code>	273
<code>\glsfirstlongonlyfont: new</code>	304	<code>long-hyphen-noshort-desc-noreg:</code>	
<code>\glslonghyphenfont: new</code>	291	<code>new</code>	293
<code>\glslongonlyfont: new</code>	304	<code>long-hyphen-postshort-hyphen: new</code>	296
<code>\glslonguserfont: initialised to default</code>		<code>long-hyphen-postshort-hyphen-desc:</code>	
<code>font</code>	281	<code>new</code>	298
<code>\glxtr@newabbreviation: added</code>		<code>long-hyphen-short-hyphen: new</code>	291
<code>\glxtrorgshort and</code>		<code>long-hyphen-short-hyphen-desc:</code>	
<code>\glxtrorglong</code>	192	<code>new</code>	292
<code>\GlsXtrDefineAcShortcuts: new</code>	18	<code>long-noshort-desc-noreg: new</code>	229
<code>\glxtrgenabbrvfmt: added check for</code>		<code>long-noshort-noreg: new</code>	230
<code>\ifglxtrininsertinside</code>	208	<code>long-only-short-only: new</code>	304
<code>\glxtrhyphensuffix: new</code>	291	<code>long-only-short-only-desc: new</code>	306
<code>\glxtrifhyphenstart: new</code>	290	<code>long-short-user-desc: corrected first</code>	
<code>\glxtrlonghyphen: new</code>	295	<code>forms</code>	287
<code>\glxtrlonghyphennoshort: new</code>	293	1.18 (2017-08-10)	
<code>\glxtrlonghyphenshort: new</code>	290	<code>stylemods: changed default value to</code>	
<code>\glxtrlongshortdescname: new</code>	215	<code>"default"</code>	21
<code>\glxtronlydescname: new</code>	306	1.19 (2017-09-09)	
<code>\glxtronlydescsort: new</code>	306	<code>\@glxtr@defaultnumberformat: new</code>	7
<code>\glxtronlysuffix: new</code>	304	<code>\@glxtr@dorecord: Use</code>	
<code>\glxtrparen: new</code>	195	<code>\@glxtr@recordloc instead of</code>	
<code>\glxtrposthyphenlong: new</code>	301	<code>\@glxtr@recordloc</code>	10
<code>\glxtrposthyphenshort: new</code>	295	<code>\@glxtr@dorecordnodefer: Use</code>	
<code>\glxtrposthyphensubsequent: new</code>	296	<code>\theglentrycounter for the</code>	
<code>\glxtrshortdescname: new</code>	224	<code>location rather than \@glxtr@recordloc</code>	11
<code>\glxtrshorthyphen: new</code>	301	<code>\@glxtr@record@setting: new</code>	13
<code>\glxtrshorthyphenlong: new</code>	299	<code>\@glxtr@record@setting@alsoindex:</code>	
<code>\glxtrshortlongdescname: new</code>	217	<code>new</code>	13
<code>\glxtrshortlongdescsort: new</code>	217	<code>\@glxtrifhasfield: new</code>	32
<code>\GlsXtrsubsequentfmt: new</code>	210	General: added <code>\glslink</code> option	
<code>\glxtrsubsequentfmt: new</code>	210	<code>theHvalue</code>	61
<code>\GlsXtrsubsequentplfmt: new</code>	210	added <code>\glslink</code> option <code>thevalue</code>	61
<code>\glxtrsubsequentplfmt: new</code>	210	<code>\glxtr@writefields: removed</code>	
<code>\glxtrword: new</code>	191	<code>double-quotes around \@jobname</code>	133
<code>\glxtrwordsep: new</code>	191	<code>\glxtrdoautoindexname: changed</code>	
<code>short-hyphen-long-hyphen: new</code>	299	<code>format test</code>	178
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink: new</code>	84
<code>new</code>	300	<code>\glxtrifhasfield: new</code>	32
<code>short-hyphen-postlong-hyphen: new</code>	301	<code>\GlsXtrSetDefaultNumberFormat:</code>	
		<code>new</code>	7

\s@glxstrifhasfield: new	32	\@glxtrsetaliasnoindex: changed to use \glxstrifhasfield instead of \ifglshasfield	80
1.20 (2017-09-11)		\@glxtrwrglossmark: new	24
\@glxtrhypernameprefix: new	118	\@rGLS: new	147
\glsdohypertarget: added redefinition	119	\@rGLS@: new	147
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glxtrhypernameprefix	139	\@rGLSpl: new	147
1.21 (2017-11-03)		\@rGLSpl@: new	147
\@@glxtr@record: added check for default options	9	\@rGls: new	146
\@@glxtrwrglossmark: new	24	\@rGls@: new	146
\@gls@setup@default@short@access: modified index to remove hard coded \space	376	\@rGlspl: new	147
modified list to remove hard coded \space	365	\@rGlspl@: new	147
moved conditional outside of \glsgroupskip	368–375	\@rgls: new	145
redefined altlistgroup to discourage breaks after group headings	366	\@rgls@: new	145
redefined altlisthypergroup to discourage breaks after group headings	367	\@rglspl: new	146
redefined indexgroup to discourage breaks after group headings	377	\@rglspl@: new	146
redefined indexhypergroup to discourage breaks after group headings	377	General: adjusted mcolalttree	396
redefined listgroup to discourage breaks after group headings	366	ac	21
redefined listhypergroup to discourage breaks after group headings	366	new	399
\@glslink: changed \let to \def	85	redefined alttreegroup to discourage breaks after group headings	392
\@glxtr@checkgroup: new	139	redefined alttreehypergroup to discourage breaks after group headings	393
\@glxtr@defpostpunc: new	16	redefined mcolalttreegroup to discourage breaks after group headings	397
\@glxtr@do@record@wrglossary: new	8	redefined mcolalttreehypergroup to discourage breaks after group headings	397
\@glxtr@dosee@alsoindex@glossary: new	25	redefined mcolalttreespannav to discourage breaks after group headings	398
\@glxtr@doseeglossary: new	25	redefined mcolindexgroup to discourage breaks after group headings	393
\@glxtr@noidx@do: removed code dealing with the group	140	redefined mcolindexhypergroup to discourage breaks after group headings	393
\@glxtr@record@setting@off: new .	13	redefined mcolindexspannav to discourage breaks after group headings	394
\@glxtr@record@setting@only: new	13	redefined mcoltreegroup to discourage breaks after group headings	394
\@glxtr@rglstrigger@record: new	144	redefined mcoltreehypergroup to discourage breaks after group headings	395
\@glxtrglossentry: new	134		
\@glxtrnewgls: new	141		

redefined mcoltreenonamegroup to discourage breaks after group headings	395	\glstr@org@dohyperlink: new	83
redefined mcoltreenonamehypergroup to discourage breaks after group headings	396	\glstr@setbookindexmark: new ...	404
redefined mcoltreenonamespannav to discourage breaks after group headings	396	\glstrbookindexatendgroup: new .	400
redefined mcoltreespannav to discourage breaks after group headings	395	\glstrbookindexbetween: new	400
redefined treenonamegroup to discourage breaks after group headings	380	\glstrbookindexbookmark: new ...	400
redefined treenonamehypergroup to discourage breaks after group headings	380	\glstrbookindexcols: new	399
debug: new	24	\glstrbookindexcolspread: new ..	400
\gglsetwidest: new	382	\glstrbookindexfirstmark: new ..	404
\gl:disablehyper: added check for existence	85	\glstrbookindexfirstmarkfmt: new	404
changed to use \def rather than \let .	85	\glstrbookindexformatheader: new	400
\gl:dohypertarget: redefined treegroup to discourage breaks after group headings	378	\glstrbookindexgroupskip: new ..	400
redefined treehypergroup to discourage breaks after group headings	379	\glstrbookindexlastmark: new ...	404
\gl:enablehyper: changed to use \def rather than \let	85	\glstrbookindexlastmarkfmt: new	404
\gl:fmtname: new	321	\glstrbookindexmarkentry: new ..	404
\gl:fmtname: new	321	\glstrbookindexname: new	399
\gl:shex: new	327	\glstrbookindexparentchildsep: new	399
\gl:slstchildpostlocation: new ..	365	\glstrbookindexparentschildsep: new	399
\gl:slstchildprelocation: new ...	365	\glstrbookindexprelocation: new	399
\gl:slstprelocation: new	365	\glstrbookindexsubatendgroup: new	400
\gl:snahyperlink: patched	83	\glstrbookindexsubbetween: new .	400
\gl:seeitemformat: new	43	\glstrbookindexsubname: new	399
\gl:sshowtarget: new	25	\glstrbookindexsubprelocation: new	399
\gl:streechildprelocation: new ...	376	\glstrbookindexsubsubatendgroup: new	400
\gl:streeprelocation: new	376	\glstrbookindexsubsubbetween: new	400
\gl:triggerrecordformat: new	145	\glstrbookindexthepage: new	403
\gl:useabbrvfont: new	208	\glstrdetoklocation: new	143
\gl:uselongfont: new	208	\glstrenablerecordcount: new ...	144
\glstr@do@alsoindex@wrglossary: new	8	\glstrglossentry: new	134
\glstr@org@do@wrglossary: new ..	27	\glstrgroupfield: new	139
		\glstrheadname: new	312
		\glstrheadname: new	312
		\GlsXtrIfFieldEqStr: new	34
		\glstriflabelinlist: new	138
		\glstrifrecordtrigger: new	144
		\glstrindexseealso: added check that the entry exists	44
		\glstrinithyperoutside: new	61
		\GlsXtrLocationRecordCount: new .	143
		\glstrnewgls: new	141, 142
		\glstrnewGLSlike: new	142
		\glstrnewglslike: new	142
		\glstrnewrgls: new	143
		\glstrnewrGLSlike: new	143

\glstrnewrglslike: new	143	\@glstr@orgprintglossary: changed	
\glstrprelocation: new	364, 399	explicit \let for \nopostdesc to	
\GlsXtrRecordCount: new	143	\glstractivatenopost	117
\glstrrecordtriggervalue: new ..	144	\@glstrglossentryother: new	135
\glstrresourcefile: now disables		\glossentrynameother: new	176
record key	131	\glseeitemformat: switched check	
\glstrresourceinit: new	131	from regular to short	43
\GlsXtrSetRecordCountAttribute:		\glstr@setaccessdisplay: new ...	176
new	144	\glstr@writefields: provide	
\glstrtitlename: new	312	\glstr@record in aux file	132
\glstrtitleorpdforheading: new .	308	\glstractivatenopost: new	117
\GlsXtrTotalRecordCount: new	143	\glstrbookindexprelocation:	
\glstrwrglossmark: new	24	removed check for no post dot	399
short-em: new	267	\glstrglossentryother: new	135
short-sc: corrected first letter		\glstrnopostpunc: new	117
uppercasing	235	1.23 (2017-11-12)	
short-sm: corrected first letter		\@@glstrfmt: added check for indexing	30
uppercasing	249	added grouping	29
\ifglstr@hyperoutside: new	61	new	29
all: new	363	\@glstr@nopostpunc@postdesc: new	118
nolong-short: new	226	\@glstr@restore@postpunc: new ..	118
nolong-short-em: new	269	\@glstrentryfmt: fixed missing label	
nolong-short-noreg: new	227	argument	30
nolong-short-sc: new	237	\@glstrfmt: new	29
nolong-short-sm: new	251	\eglupdatewidest: new	383
nopostdot: new	16	\gglupdatewidest: new	382
postpunc: new	16	\glupdatewidest: new	382
\printunrtglossaryentryprocesshook:		\GlsXtrDefineAbbreviationShortcuts:	
new	137, 138	changed \newabbr definition to use	
\printunrtglossarypredoglossary:		\providecommand	18
new	138	\GlsXtrDefineAcShortcuts: changed	
\printunrtglossaryskipentry: new	137	\newabbr definition to use	
\rGLS: new	147	\providecommand	19
\rGls: new	146	\glstrfmtdisplay: new	30
\rgls: new	145	\glstrifcustomdiscardperiod: new	186
\rGLSformat: new	148	\GlsXtrIfFieldUndef: new	33
\rGlsformat: new	148	\glstrrestorepostpunc: new	118
\rglsformat: new	148	\s@glstrfmt: new	29
\rGLSpl: new	147	\s@glstrfmt: new	29
\rGlspl: new	146	\xglupdatewidest: new	383
\rglspl: new	146	1.24 (2017-11-14)	
\rGLSplformat: new	148	\gladd: added \@gls@setsort	65
\rGlsplformat: new	148	\glstrforcsvfield: new	32
\rglsplformat: new	148	\glstrlocalsetgrouptitle: new ..	122
\s@glstrifhasfield: switched from		1.25 (2017-11-14)	
\ifdef to \ifundef	32	\glstrbookindexmulticolenv: new	400
1.22 (2017-11-08)		1.25 (2017-11-24)	
\@glstr@nopostpunc: new	117	\glsextrapostnamehook: new	175
		\glstrfootnotename: new	219

\glxtrlongnoshortdescname: new .	227	\glxtrGeneralLatinIIrules: new .	338
\glxtrlongnoshortname: new	229	\glxtrGeneralLatinIrules: new . .	337
\glxtrlongshortname: new	214	\glxtrGeneralLatinIVrules: new .	339
\glxtrlongshortuserdescname: new	284	\glxtrGeneralLatinVIIrules: new	342
\glxtronlyname: new	304	\glxtrGeneralLatinVIIrules: new	341
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVrules: new .	341
changed to use \glxtrparen	187	\glxtrGeneralLatinVrules: new . .	340
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules: new . .	337
changed to use \glxtrparen	187	\glxtrgeneralpuncIrules: new . . .	336
\glxtrshortlongname: new	216	\glxtrgeneralpuncrules: new	335
\glxtrshortlonguserdescname: new	286	\glxtrhyphenrules: new	335
\glxtrshortnolongname: new	222	\glxtrLatinA: new	343
1.26 (2018-01-05)		\glxtrLatinAA: new	345
\@glxtr@do@inc@linkcount: new . .	149	\glxtrLatinAELigature: new	345
\glslinkpresetkeys: new	62	\glxtrLatinE: new	343
\glxtr@inc@linkcount: new	62	\glxtrLatinEszettSs: new	344
\GlsXtrEnableLinkCounting: new . .	150	\glxtrLatinEszettSz: new	345
\GlsXtrIfLinkCounterDef: new	150	\glxtrLatinEth: new	345
\glxtrinclinkcounter: new	149	\glxtrLatinH: new	343
\GlsXtrLinkCounterName: new	150	\glxtrLatinI: new	343
\GlsXtrLinkCounterValue: new	149	\glxtrLatinInsularG: new	345
\GlsXtrTheLinkCounter: new	149	\glxtrLatinK: new	343
1.27 (2018-02-26)		\glxtrLatinL: new	343
\@glset@setup@default@short@access:		\glxtrLatinLslash: new	346
added glossaries-extra-bib2gls.sty .	327	\glxtrLatinM: new	344
\@glxtrdialecthook: new	27	\glxtrLatinN: new	344
\Alpha: new	329	\glxtrLatinO: new	344
\Beta: new	329	\glxtrLatinOELigature: new	345
\Chi: new	330	\glxtrLatinOslash: new	345
\Digamma: new	330	\glxtrLatinP: new	344
\Epsilon: new	329	\glxtrLatinS: new	344
\Eta: new	330	\glxtrLatinSchwa: new	344
\glxtr@loaddialect: new	326	\glxtrLatinT: new	344
\glxtrBasicDigitrules: new	360	\glxtrLatinThorn: new	345
\glxtrcombiningdiacriticIIrules:		\glxtrLatinWynn: new	345
new	334	\glxtrLatinX: new	344
\glxtrcombiningdiacriticIIrules:		\glxtrMathGreekIIrules: new	352
new	333	\glxtrMathGreekIrules: new	351
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new	356
new	333	\glxtrMathItalicBeta: new	356
\glxtrcombiningdiacriticIVrules:		\glxtrMathItalicChi: new	359
new	335	\glxtrMathItalicDelta: new	356
\glxtrcombiningdiacriticrules:		\glxtrMathItalicEpsilon: new . . .	356
new	333	\glxtrMathItalicEta: new	357
\glxtrcontrolrules: new	332	\glxtrMathItalicGamma: new	356
\glxtrcurrencyrules: new	336	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new	359	new	347
\glxtrfractionrules: new	360	\glxtrMathItalicGreekIrules: new	347
\glxtrGeneralLatinIIrules: new	339	\glxtrMathItalicIota: new	357

<code>\glxtrMathItalicKappa:new</code>	357	<code>\glxtrUpPi:new</code>	355
<code>\glxtrMathItalicLambda:new</code>	357	<code>\glxtrUpPsi:new</code>	356
<code>\glxtrMathItalicLowerGreekIIrules:</code>		<code>\glxtrUpRho:new</code>	355
new	350	<code>\glxtrUpSigma:new</code>	355
<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau:new</code>	355
new	349	<code>\glxtrUpTheta:new</code>	354
<code>\glxtrMathItalicMu:new</code>	357	<code>\glxtrUpUpsilon:new</code>	355
<code>\glxtrMathItalicNabla:new</code>	359	<code>\glxtrUpXi:new</code>	355
<code>\glxtrMathItalicNu:new</code>	358	<code>\glxtrUpZeta:new</code>	354
<code>\glxtrMathItalicOmega:new</code>	359	<code>\Iota:new</code>	330
<code>\glxtrMathItalicOmicron:new</code>	358	<code>\Kappa:new</code>	330
<code>\glxtrMathItalicPartial:new</code>	359	<code>\Mu:new</code>	330
<code>\glxtrMathItalicPhi:new</code>	359	<code>\Nu:new</code>	330
<code>\glxtrMathItalicPi:new</code>	358	<code>\Omicron:new</code>	330
<code>\glxtrMathItalicPsi:new</code>	359	<code>\omicron:new</code>	330
<code>\glxtrMathItalicRho:new</code>	358	<code>\Rho:new</code>	330
<code>\glxtrMathItalicSigma:new</code>	358	<code>\Tau:new</code>	330
<code>\glxtrMathItalicTau:new</code>	358	<code>\Upalpha:new</code>	330
<code>\glxtrMathItalicTheta:new</code>	357	<code>\Upbeta:new</code>	330
<code>\glxtrMathItalicUpperGreekIIrules:</code>		<code>\Upchi:new</code>	331
new	349	<code>\Upsilon:new</code>	330
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta:new</code>	331
new	348	<code>\Upiota:new</code>	331
<code>\glxtrMathItalicUpsilon:new</code>	358	<code>\Upkappa:new</code>	331
<code>\glxtrMathItalicXi:new</code>	358	<code>\Upmu:new</code>	331
<code>\glxtrMathItalicZeta:new</code>	357	<code>\Upnu:new</code>	331
<code>\glxtrMathUpGreekIIrules:new</code>	346	<code>\Upomicron:new</code>	331
<code>\glxtrMathUpGreekIrules:new</code>	346	<code>\upomicron:new</code>	331
<code>\glxtrnonprintablerules:new</code>	333	<code>\Uprho:new</code>	331
<code>\glxtrprovidecommand:new</code>	327	<code>\Uptau:new</code>	331
<code>\glxtrspacerules:new</code>	332	<code>\Upzeta:new</code>	331
<code>\glxtrSubScriptDigitrules:new</code>	360	<code>\Zeta:new</code>	329
<code>\glxtrSuperScriptDigitrules:new</code>	360		
<code>\glxtrUpAlpha:new</code>	353	1.28 (2018-03-06)	
<code>\glxtrUpBeta:new</code>	353	<code>\@glxtr@docdefval:</code> changed from	
<code>\glxtrUpChi:new</code>	356	count register to macro	15
<code>\glxtrUpDelta:new</code>	353	<code>\@glxtrdialecthook:</code> save and restore	
<code>\glxtrUpDigamma:new</code>	353	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon:new</code>	353	361
<code>\glxtrUpEta:new</code>	354	<code>\glxtrredeffield:</code> changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code>	353	<code>\protected@csedef</code>	33
<code>\glxtrUpIota:new</code>	354	<code>\glxtrlocalsetgrouptitle:</code> changed	
<code>\glxtrUpKappa:new</code>	354	<code>\csedef\protected@csedef</code>	122
<code>\glxtrUpLambda:new</code>	354	<code>\glxtrsetgrouptitle:</code> changed	
<code>\glxtrUpMu:new</code>	354	<code>\csxdef\protected@csxdef</code>	122
<code>\glxtrUpNu:new</code>	354	1.29 (2018-04-09)	
<code>\glxtrUpOmega:new</code>	356	<code>\@glxtr@removespaces:</code> added expansion	125
<code>\glxtrUpOmicron:new</code>	355	<code>\@glxtr@dorecord:</code> don't suppress	
<code>\glxtrUpPhi:new</code>	355	expansion of <code>\@glxtrrecordloc</code> if	
		counter isn't page	11

\@glxstr@wrglossary@locationhyperlink: new	23	\Glsxtrlongpl: added \@glxstr@record	206
\glxstr@inc@wrglossaryctr: new ...	22	\glxtrlongpl: added \@glxstr@record	206
\glxstr@wrglossarylocation: new ..	327	\GLSxtrshort: added \@glxstr@record	201
\GlsXtrBibTeXEntryAliases: new ..	328	\Glsxtrshort: added \@glxstr@record	201
\glxtrfieldforlistloop: corrected argument order in \forlistcsloop	31	\Glsxtrshort: added \@glxstr@record	201
\GlsXtrIndexCounterLink: new	328	\glxtrshort: added \@glxstr@record	200
\GlsXtrInternalLocationHyperlink: new	23	\GLSxtrshortpl: added \@glxstr@record	205
\GlsXtrProvideBibTeXFields: new ..	329	\Glsxtrshortpl: added \@glxstr@record	205
indexcounter: new	23	\glxtrshortpl: added \@glxstr@record	204
\setentrycounter: new	124	\GlsXtrStartUnsetBuffering: new ..	95
1.30 (2018-04-25)		\GlsXtrStopUnsetBuffering: new ...	95
\@@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 ..	96
\@GLSxtr@fullpl: added \@glxstr@record	200	1.31 (2018-05-09)	
\@GlsXtrStopUnsetBuffering: new ..	96	\@GlsXtrStartUnsetBuffering: new ..	95
\@Glsxtr@fullpl: added \@glxstr@record	199	\@gl@ifaccessattribute@set: new	158
\@glxstr@dorecord: don't suppress expansion of \@glarecordlocoref ..	11	\@gl@initaccesskeys: new ...	158, 164
\@glxstr@full: added \@glxstr@record	197	\@gl@setup@default@short@access: new	158, 164
\@glxstr@fullpl: added \@glxstr@record	199	\@glxstr@record@noglossarywarning: new	130
\@glxstr@glossadd@postkeys: new ..	10	\@glxstrbuffer@nodup@unset: new ..	95
\@glxstr@glossadd@prekeys: new ...	10	General: added prefix key for glslink	61
\@glxstr@glslink@postkeys: new ...	10	added prefix key for printgloss ..	119
\@glxstr@glslink@prekeys: new	10	changed \let to \def	118
\@glxstr@local@textformat: new ...	61	\glsaddeach: new	65
\@glxstr@unset: new	94	\glsapturedgroup: new	327
\@glxstrbuffer@unset: new	95	\glsdefpostdesc: new	186
\glsadd: added \glsaddpostsetkeys ..	65	\glsdefpostlink: new	186
added \glsaddpresetkeys	65	\glsdefpostname: new	176
\glsaddpostsetkeys: new	64	\glsdohypertarget: bug fix: ensure that new version is picked up	119
\glsaddpresetkeys: new	64	\glslistdesc: new	365
\glsuserdescription: new	281	\glslocalreseteach: new	96
\glxtrabbreviationfont: new	58	\glslocalunseteach: new	97
\GlsXtrDualBackLink: new	328	\glstreechilddesc: new	378
\GlsXtrDualField: new	328	\glstreechildsymbol: new	378
\GlsXtrExpandedFmt: new	62	\glstreedefaultnamefmt: new	375
\GLSxtrlong: added \@glxstr@record	203	\glstreegroupheaderfmt: added redefinition	375
\Glsxtrlong: added \@glxstr@record	203	\glstreenamefmt: added redefinition	375
\glxtrlong: added \@glxstr@record	202		
\GLSxtrlongpl: added \@glxstr@record	207		

\glstreenavigationfmt:added		\GlsXtrIfHasNonZeroChildCount:	
redefinition	376	new	327
\glstreenonamechilddesc:new	379	\GlsXtrIfXpFieldEqXpStr:new	35
\glstreenonamesymbol:new	379	\glxtrpostlinkAddSymbolDescOnFirstUse:	
\glstreesymbol:new	378	new	187
\glxtr@newabbreviation:added		\GlsXtrRecordWarning:new	129
\ExtraCustomAbbreviationFields		\glxtrRevertTocMarks:new	308
.....	192	\GlsXtrStandaloneGlossaryType:	
\GlsXtrForUnsetBufferedList:new .	96	new	135
\GlsXtrIfFieldCmpNum:new	33	\GlsXtrStandaloneSubEntryItem:	
\GlsXtrIfFieldEqNum:new	32	new	135
\GlsXtrIfFieldEqXpStr:new	35	\s@GlsXtrStartUnsetBuffering:new	95
\GlsXtrIfFieldNonZero:new	32		

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	
\\$	327
\.	16, 17, 187, 375
\@	48, 131
\@cGLS@	99, 107
\@cGLSpl@	99, 107
\@cGls@	99, 107
\@cGlspl@	99, 107
\@cglS@	99, 107
\@cglSpl@	99, 107
\@do@wrglossary	8, 10, 114
\@do@wrglossary	12–14, 27, 65, 81
\@glo@assign@sortkey	119
\@glo@list	6
\@glo@type	136
\@glossarysec	400
\@gls@expand@field	29
\@glslocalreset	96
\@glslocalunset	96
\@glsreset	96
\@glsunset	95
\@glsxtr@autoindex@escspch	180–182
\@glsxtr@checkspch	179, 180, 182
\@glsxtr@disabledflycommand	53
\@glsxtr@org@postdescription	117, 118
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 134
\@glsxtrfmt	29
\@glsxtrp	90
\@glsxtrp@postloctag	56
\@glsxtrp@preloctag	56
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 44, 49, 113
\@newglossaryentry@defcounters	98
\@newglossaryentry@defunitcounters	105
\@par	381
\@ACRlong	86
\@ACRlongpl	86
\@ACRshort	86
\@ACRshortpl	86
\@Acrlong	86
\@Acrlongpl	86
\@Acrshort	86
\@Acrshortpl	86
\@GLS@	86, 101, 102, 147
\@GLSdesc@	70
\@GLSpl@	86, 102, 148
\@GLSplural@	87
\@GLSSymbol@	71
\@GLSxtr@	86
\@GLSxtr@full	198
\@GLSxtr@fullpl	199
\@GLSxtr@p@acrlong@	86
\@GLSxtr@p@acrlongpl@	86
\@GLSxtr@p@acrshort@	86
\@GLSxtr@p@acrshortpl@	86
\@GLSxtr@p@long@	86
\@GLSxtr@p@longpl@	86
\@GLSxtr@p@plural@	86
\@GLSxtr@p@short@	86
\@GLSxtr@p@shortpl@	86
\@GLSxtr@p@text@	86
\@GLSxtrlong	86, 203
\@GLSxtrlongpl	86, 207
\@GLSxtrp	93, 94
\@GLSxtrshort	86, 201
\@GLSxtrshortpl	86, 205
\@Gls@	86, 101, 102, 146
\@Gls@acentryname	110
\@Gls@entry@field	77, 92, 93, 177
\@Gls@entryname	110
\@GlsXtrEnableOnTheFly	50, 51
\@GlsXtrStartUnsetBuffering	95
\@GlsXtrStopUnsetBuffering	95

\@Glspl@	86, 101, 102, 147	\@end@glxtr@addunused	48
\@Glsplural@	87	\@end@glxtr@gettype	115, 119
\@Glstext@	86	\@end@glxtr@usesee	43
\@Glsxtr	52, 53	\@end@glxtrifhyphenstart	290
\@Glsxtr@full	197	\@endfortrue	32, 176, 211
\@Glsxtr@fullpl	199	\@firstofone	66, 137, 170, 171, 178, 184
\@Glsxtr@p@acrlong@	86	\@firstofthree	60, 65, 73–76, 82, 83, 197, 199, 200, 202, 204, 206
\@Glsxtr@p@acrlongpl@	86	\@firstoftwo 67, 68, 70, 71, 74–77, 79, 82, 112, 176, 188, 189, 197, 199, 200, 204–207, 308, 309
\@Glsxtr@p@acrshort@	86	\@for	6, 22, 32, 48, 65, 97, 98, 109, 113, 116, 122, 137, 144, 150, 169, 176, 183
\@Glsxtr@p@acrshortpl@	86	\@glo@alias	45, 46
\@Glsxtr@p@long@	86	\@glo@assign@sortkey	115
\@Glsxtr@p@longpl@	86	\@glo@autosee	25
\@Glsxtr@p@plural@	86	\@glo@autoseehook	46
\@Glsxtr@p@shortpl@	86	\@glo@category	103
\@Glsxtr@p@text@	86	\@glo@check@sortallowed	116
\@Glsxtrlong	86, 203	\@glo@counterprefix	10, 11, 125
\@Glsxtrlongpl	86, 206	\@glo@countunit	103
\@Glsxtrp	92, 93	\@glo@default@sorttype	115
\@Glsxtrpl	52, 53	\@glo@desc	37
\@Glsxtrshort	86, 201	\@glo@descplural	37
\@Glsxtrshortpl	86, 204	\@glo@group	13
\@acrlong	86	\@glo@label 12, 13, 28, 42, 45–47, 77, 85, 384–390
\@acrlongpl	86	\@glo@location	12
\@acrshort	86	\@glo@loclist	12
\@acrshortpl	86	\@glo@name	179
\@addtoreset	149	\@glo@no@assign@sortkey	119
\@afterheading 366, 367, 377, 379–381, 393–396, 403	\@glo@parent	385, 386
\@alt@glshyp@opt	82	\@glo@see	42, 43, 46–48
\@auxout	11, 12, 49, 57, 99, 108, 113, 126, 131–134, 404	\@glo@seealso	45, 46
\@bibgls@restreat	131	\@glo@sort	179
\@cGLS	102	\@glo@sorttype	115, 123
\@cGLS@	99, 102, 107	\@glo@text	60
\@cGLSpl	102	\@glo@thislettergrp	140
\@cGLSpl@	99, 102, 107	\@glo@thisvalue	281
\@cGls@	99, 107	\@glo@tmp	28, 44, 77
\@cGlspl@	99, 107	\@glo@type	47, 83, 110, 113, 116, 117, 119, 122, 123, 126, 129, 130, 136, 137
\@cgls@	99, 107	\@glo@types	167, 168, 383–390
\@cglspl@	99, 107	\@glossary@default@style	54, 116, 398
\@disable@onlypremakeg	113	\@glossarystyle	116, 117
\@do@auxoutstuff	126	\@gls@	86, 100, 102, 146
\@do@glsg@getcounterprefix	10, 11	\@gls@@link	60
\@do@glsssee	46, 47	\@gls@ReturnAfterFi	125
\@do@newglossaryentry	110, 194	\@gls@actualchar	179
\@do@seeglossary	13, 14, 25, 49, 113		
\@do@wrglossary	64, 145		
\@empty . 65, 73–77, 118, 124, 179, 180, 197–207			

<code>\@gls@adjustmode</code>	65	<code>\@gls@long</code>	193
<code>\@gls@alt@hyp@opt</code>	83	<code>\@gls@longpl</code>	190, 192–194
<code>\@gls@alt@hyp@opt@char</code>	82, 83	<code>\@gls@map</code>	176
<code>\@gls@alt@hyp@opt@keys</code>	83	<code>\@gls@nameaccess</code>	158, 159
<code>\@gls@automake</code>	116	<code>\@gls@nohyperlist</code>	38, 40
<code>\@gls@between</code>	122	<code>\@gls@noidx@do</code>	123
<code>\@gls@checkedmidx</code>	179, 180, 182	<code>\@gls@noidx@getgrouptitle</code>	137
<code>\@gls@checkmidxchars</code>	44, 179	<code>\@gls@noidx@nosanitizesort</code>	115
<code>\@gls@codepage</code>	126	<code>\@gls@noidx@sanitizesort</code>	115
<code>\@gls@counter</code>	9, 11, 23, 62, 65, 81, 145	<code>\@gls@noidx@loclist@finalsep</code>	120
<code>\@gls@currentlettergroup</code> ...	123, 137, 140	<code>\@gls@noidx@loclist@prev</code>	120
<code>\@gls@declareoption</code>	5	<code>\@gls@noidx@loclist@sep</code>	120
<code>\@gls@default@longpl</code>	192–194	<code>\@gls@noref@warn</code>	114, 123
<code>\@gls@doautomake</code>	116, 133	<code>\@gls@org@glsnoidx@displayloc</code>	120
<code>\@gls@doautomake@err</code>	133	<code>\@gls@org@gls@seeformat</code>	120
<code>\@gls@enablesavenonumberlist</code>	48	<code>\@gls@preglossaryhook</code>	117, 183
<code>\@gls@encapchar</code>	180	<code>\@gls@prevlevel</code>	391–393, 397, 398
<code>\@gls@entry@count</code>	99	<code>\@gls@quotechar</code>	179
<code>\@gls@entry@field</code>		<code>\@gls@reference</code>	49, 113
.....	28, 33, 46, 77, 91–94, 98, 135, 136	<code>\@gls@restoreat</code>	48
<code>\@gls@entry@unitcount</code>	107, 108	<code>\@gls@saveentrycounter</code>	13, 14, 27, 63, 65, 145
<code>\@gls@field@font</code>	66–73	<code>\@gls@see@noindex</code>	26, 131
<code>\@gls@field@link</code>	66–73, 78, 79	<code>\@gls@setdefault@glslink@opts</code>	
<code>\@gls@firstaccess</code>	158, 160	9, 30, 63, 81
<code>\@gls@getcounterprefix</code>	10, 11	<code>\@gls@setsort</code>	63, 65
<code>\@gls@getgrouptitle</code>	122, 137	<code>\@gls@setupsort@none</code>	14
<code>\@gls@grptitle</code>	83, 122	<code>\@gls@short</code>	193
<code>\@gls@hyp@opt</code>		<code>\@gls@shortaccess</code>	158–160
...	78, 79, 83, 102, 142, 145–147, 197–207	<code>\@gls@shortaccesspl</code>	158, 159
<code>\@gls@hyp@opt@cs</code>	82, 83	<code>\@gls@shortpl</code>	190, 193, 194
<code>\@gls@ifaccessattribute@set</code>	159	<code>\@gls@sort</code>	140
<code>\@gls@ifinlist</code>	138	<code>\@gls@textaccess</code>	158, 160
<code>\@gls@increment@currcount</code>	98	<code>\@gls@thislabel</code>	65, 97
<code>\@gls@increment@currunitcount</code>	106	<code>\@gls@thisval</code>	176
<code>\@gls@initaccesskeys</code>	192	<code>\@gls@tmp</code>	35, 122
<code>\@gls@keymap</code> ..	12, 13, 28, 42, 45, 77, 132, 176	<code>\@gls@tmpb</code>	182
<code>\@gls@label</code> ..	8, 9, 11, 49, 82, 113, 114, 134, 211	<code>\@gls@type</code>	113, 114, 116, 211, 384–390
<code>\@gls@levelchar</code>	180	<code>\@gls@write@entrycounts</code>	99
<code>\@gls@link</code>	30, 59, 60, 73–77, 197–207	<code>\@gls@write@entryunitcounts</code>	107
<code>\@gls@link@checkfirsthyper</code>	60, 112	<code>\@gls@write@entryunitcounts@do</code>	108
<code>\@gls@link@label</code>	62, 144	<code>\@gls@xref</code>	12, 44, 45
<code>\@gls@link@nocheckfirsthyper</code>		<code>\@gls@abbrv@current@abbreviation</code>	192, 208
.....	59, 73–77, 197–207	<code>\@gls@sacronymlists</code>	110
<code>\@gls@link@opts</code>	62	<code>\@gls@doifexistsorwarn</code>	15, 172–176
<code>\@gls@list</code>	122	<code>\@gls@entry</code>	99, 108
<code>\@gls@local@increment@currcount</code>	98	<code>\@gls@link</code>	64, 83, 85
<code>\@gls@local@increment@currunitcount</code>	106	<code>\@gls@localreset</code>	97
<code>\@gls@location</code>	140, 141	<code>\@gls@localunset</code>	96, 97
<code>\@gls@loclist</code>	119–121, 140, 141	<code>\@gls@nextpages</code>	117

<code>\@glsnonextpages</code>	117	<code>\@glxtr@accessdisplay</code>	176, 177
<code>\@glsnnumberformat</code>		<code>\@glxtr@activate@initialtagging</code> ..	
.....	9, 11, 62, 65, 81, 145, 175, 178	183, 185
<code>\@glspartner</code>	113	<code>\@glxtr@addunitcounter</code>	103
<code>\@glsp1@</code>	86, 101, 102, 146	<code>\@glxtr@addunused</code>	48
<code>\@glsplural@</code>	87	<code>\@glxtr@addunusedxrefs</code>	47, 48
<code>\@glspunc@token</code>	189	<code>\@glxtr@attrval</code>	63, 170–175, 177–179
<code>\@glsrecordlocref</code>	10, 11	<code>\@glxtr@autoindex@cat</code>	179, 180
<code>\@glsshowtarget</code>	84	<code>\@glxtr@autoindex@doextra@esc</code>	179
<code>\@glssstyle@altlist</code>	365	<code>\@glxtr@autoindex@encap</code>	179–181
<code>\@glssstyle@altlistgroup</code>	366	<code>\@glxtr@autoindex@esc</code>	179, 181, 182
<code>\@glssstyle@altlisthypergroup</code>	367	<code>\@glxtr@autoindex@escat</code>	179, 180
<code>\@glssstyle@alttree</code>	381	<code>\@glxtr@autoindex@escencap</code> ...	180, 181
<code>\@glssstyle@alttreegroup</code>	392	<code>\@glxtr@autoindex@esclevel</code> ...	180, 181
<code>\@glssstyle@alttreehypergroup</code>	393	<code>\@glxtr@autoindex@escquote</code> ...	179, 181
<code>\@glssstyle@index</code>	376	<code>\@glxtr@autoindex@level</code>	180, 181
<code>\@glssstyle@indexgroup</code>	377	<code>\@glxtr@autoindex@setname</code>	178
<code>\@glssstyle@indexhypergroup</code>	377	<code>\@glxtr@autoindex@crossrefs</code> 14, 15, 42, 45	
<code>\@glssstyle@inline</code>	375	<code>\@glxtr@autoseeindexfalse</code>	14
<code>\@glssstyle@list</code>	365	<code>\@glxtr@autoseeindextrue</code>	16
<code>\@glssstyle@listdotted</code>	364	<code>\@glxtr@bookindex@atendgroup</code> .	401–403
<code>\@glssstyle@listgroup</code>	366	<code>\@glxtr@bookindex@atsubendgroup</code> ..	402
<code>\@glssstyle@listhypergroup</code>	366	<code>\@glxtr@bookindex@atsubsubendgroup</code>	402
<code>\@glssstyle@mcolalttree</code>	396	<code>\@glxtr@bookindex@between</code>	401, 403
<code>\@glssstyle@mcolalttreegroup</code>	397	<code>\@glxtr@bookindex@sep</code>	401, 402
<code>\@glssstyle@mcolalttreehypergroup</code> ..	397	<code>\@glxtr@bookindex@subatendgroup</code> ..	
<code>\@glssstyle@mcolalttreesspannav</code>	398	401–403
<code>\@glssstyle@mcolindexgroup</code>	393	<code>\@glxtr@bookindex@subbetween</code> .	401, 402
<code>\@glssstyle@mcolindexhypergroup</code>	393	<code>\@glxtr@bookindex@subsep</code>	401, 402
<code>\@glssstyle@mcolindexspannav</code>	394	<code>\@glxtr@bookindex@subsubatendgroup</code>	
<code>\@glssstyle@mcoltreegroup</code>	394	401–403
<code>\@glssstyle@mcoltreehypergroup</code>	395	<code>\@glxtr@bookindex@subsubbetween</code> ..	
<code>\@glssstyle@mcoltreenamegroup</code>	395	401, 402
<code>\@glssstyle@mcoltreenamehypergroup</code>	396	<code>\@glxtr@bookindex@groupskip</code> ...	401, 403
<code>\@glssstyle@mcoltreenamepannav</code> ..	396	<code>\@glxtr@cat</code>	98, 109, 144, 183
<code>\@glssstyle@mcoltreesspannav</code>	395	<code>\@glxtr@checkgroup</code>	137
<code>\@glssstyle@tree</code>	377	<code>\@glxtr@counterrecordhook</code>	11
<code>\@glssstyle@treegroup</code>	378	<code>\@glxtr@csname</code>	104, 105, 107
<code>\@glssstyle@treehypergroup</code>	379	<code>\@glxtr@current@style</code>	54, 398
<code>\@glssstyle@treename</code>	379	<code>\@glxtr@currentunitcount</code> ..	104, 105, 107
<code>\@glssstyle@treenamegroup</code>	380	<code>\@glxtr@currunitcount</code>	106, 108
<code>\@glssstyle@treenamehypergroup</code> ...	380	<code>\@glxtr@debugnr</code>	24
<code>\@glstarget</code>	85, 118, 119	<code>\@glxtr@debugval</code>	24
<code>\@glstext@</code>	86	<code>\@glxtr@declareoption</code> ...	5, 16, 18, 21, 23
<code>\@glunset</code>	95, 96	<code>\@glxtr@defaultnoglossarywarning</code> ..	21
<code>\@glswidestname</code>	383, 391	<code>\@glxtr@defaultnumberformat</code>	
<code>\@glxtr</code>	51, 53	7, 9, 62, 65, 81, 175, 178
<code>\@glxtr@@do@wrglossary</code>	114	<code>\@glxtr@defpostpunc</code>	16, 17, 25
<code>\@glxtr@abbreviationsdef</code>	18, 26		

<code>\@glsxtr@deprecated@abbrstyle</code>	<code>\@glsxtr@gobbleto@endescspch</code>
240, 241, 243, 245, 254, 255, 257, 259, 271, 275, 278, 280	182
<code>\@glsxtr@disabledflycommand</code>	<code>\@glsxtr@groupheading</code>
53	137, 139, 140
<code>\@glsxtr@display@loc</code>	<code>\@glsxtr@idx@displaynumberlist</code>
123	114
<code>\@glsxtr@do@wrindex</code>	<code>\@glsxtr@idx@entrynumberlist</code>
82	115
<code>\@glsxtr@do@glsglisablehyperinlist</code> ..	<code>\@glsxtr@ifcsstart</code>
80	51
<code>\@glsxtr@do@inc@linkcount</code>	<code>\@glsxtr@ifpunctoken</code>
150	189
<code>\@glsxtr@do@record@wrglossary</code>	<code>\@glsxtr@ifunitcounter</code>
8, 14	103
<code>\@glsxtr@do@redef@forglsentries</code>	<code>\@glsxtr@insert@dots</code>
7	191
<code>\@glsxtr@do@style</code>	<code>\@glsxtr@insert@dots@next</code>
22, 327	191
<code>\@glsxtr@do@titlecaps@warn</code>	<code>\@glsxtr@insertdots</code>
170–173, 177, 184	159, 193
<code>\@glsxtr@doabbreviationsdef</code>	<code>\@glsxtr@label</code>
18	32, 48, 150, 169
<code>\@glsxtr@doaccsupp</code>	<code>\@glsxtr@loadstyles</code>
21, 24	363, 364
<code>\@glsxtr@docdefsetting</code>	<code>\@glsxtr@local@textformat</code>
15	62–64
<code>\@glsxtr@docdefval</code>	<code>\@glsxtr@longnewglossaryentry</code>
15, 48, 50	37
<code>\@glsxtr@docounterrecord</code>	<code>\@glsxtr@mark@wordseps</code>
11	191
<code>\@glsxtr@doglossary</code>	<code>\@glsxtr@mark@wordseps@next</code>
137	192
<code>\@glsxtr@doiflabelinlist</code>	<code>\@glsxtr@markwordseps</code>
138	193, 194
<code>\@glsxtr@dolocktag</code>	<code>\@glsxtr@mixed@assign@sortkey</code>
56, 57	115
<code>\@glsxtr@dorecord</code>	<code>\@glsxtr@noidx@displaynumberlist</code> ..
8, 10	114
<code>\@glsxtr@dorecordnodefer</code>	<code>\@glsxtr@noidx@do</code>
8, 10	139
<code>\@glsxtr@dosee@alsoindex@glossary</code> ..	<code>\@glsxtr@noidx@entrynumberlist</code>
14	115
<code>\@glsxtr@doseeglossary</code>	<code>\@glsxtr@noidx@numberlistloop</code>
13, 25	115
<code>\@glsxtr@dostylewarn</code>	<code>\@glsxtr@nomissingglstextnr</code>
211	21
<code>\@glsxtr@enabletagging</code>	<code>\@glsxtr@nomissingglstextval</code>
183	21
<code>\@glsxtr@end@</code>	<code>\@glsxtr@noop@recordcounter</code>
51	11, 13
<code>\@glsxtr@endescspch</code>	<code>\@glsxtr@nopostpunc</code>
179–182	117
<code>\@glsxtr@entrycount@org@localreset</code> .	<code>\@glsxtr@nopostpunc@postdesc</code> ..
99	117, 118
<code>\@glsxtr@entrycount@org@localunset</code> .	<code>\@glsxtr@notfoundinlist</code>
98	189
<code>\@glsxtr@entrycount@org@reset</code> ...	<code>\@glsxtr@op@recordcounter</code>
98, 99	14
<code>\@glsxtr@entrycount@org@unset</code>	<code>\@glsxtr@optlist</code>
98	53
<code>\@glsxtr@entryunitcount@org@localreset</code>	<code>\@glsxtr@org@@starttoc</code>
107	307, 308
<code>\@glsxtr@entryunitcount@org@localunset</code>	<code>\@glsxtr@org@GLS@</code>
106	59, 60
<code>\@glsxtr@entryunitcount@org@reset</code> .	<code>\@glsxtr@org@GLSpl@</code>
106	60
<code>\@glsxtr@entryunitcount@org@unset</code> .	<code>\@glsxtr@org@Gls@</code>
106	59
<code>\@glsxtr@err@undefaction</code>	<code>\@glsxtr@org@Glspl@</code>
7, 13	59
<code>\@glsxtr@field@linkdefs</code>	<code>\@glsxtr@org@Glsxtrtitlefirst</code> .
59	309, 310
<code>\@glsxtr@format@overridefalse</code>	<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>
178	309, 310
<code>\@glsxtr@format@overridefalse</code>	<code>\@glsxtr@org@Glsxtrtitlefull</code> ..
178	309, 310
<code>\@glsxtr@foundinlist</code>	<code>\@glsxtr@org@Glsxtrtitlefullpl</code>
189	309, 310
<code>\@glsxtr@full</code>	<code>\@glsxtr@org@Glsxtrtitlelong</code> ..
197	309, 310
<code>\@glsxtr@fullpl</code>	<code>\@glsxtr@org@Glsxtrtitlelongpl</code>
198	309, 310
<code>\@glsxtr@gettype</code>	<code>\@glsxtr@org@Glsxtrtitlename</code> ..
115	309, 310
<code>\@glsxtr@glossdescfont</code>	<code>\@glsxtr@org@Glsxtrtitleplural</code>
170, 171	309, 310
<code>\@glsxtr@glossnamefont</code>	<code>\@glsxtr@org@Glsxtrtitleshort</code> .
172–175, 177	309, 310
	<code>\@glsxtr@org@Glsxtrtitleshortpl</code>
	309, 310
	<code>\@glsxtr@org@Glsxtrtitletext</code> ..
	309, 310
	<code>\@glsxtr@org@MakeUppercase</code>
	309, 310
	<code>\@glsxtr@org@checkfirsthyper</code> ...
	79, 112

<code>\@glxtr@org@delimN</code>	56, 57	<code>\@glxtr@printglossopts</code>	53, 115, 118
<code>\@glxtr@org@delimR</code>	56, 57	<code>\@glxtr@printglossval</code>	118
<code>\@glxtr@org@doseeglossary</code>	25, 113	<code>\@glxtr@printunsrtglossaryskipentry</code>	137, 138
<code>\@glxtr@org@gloautosee</code>	26	<code>\@glxtr@provide@addstoragekey</code>	29
<code>\@glxtr@org@glolinkprefix</code>	62, 64	<code>\@glxtr@provide@storagekey</code>	28
<code>\@glxtr@org@glsl@</code>	59	<code>\@glxtr@record</code>	
<code>\@glxtr@org@glsdohypertarget</code>	119	13, 14, 58–60, 65, 197, 199–207
<code>\@glxtr@org@glsignore</code>	56, 57	<code>\@glxtr@record@noglossarywarning</code> ..	14
<code>\@glxtr@org@glspl@</code>	59	<code>\@glxtr@record@setting</code>	
<code>\@glxtr@org@glxtrtitlefirst</code> ..	309, 310	8, 10, 13, 44, 49, 50, 112
<code>\@glxtr@org@glxtrtitlefirstplural</code>	309, 310	<code>\@glxtr@record@setting@alsoindex</code> ..	
<code>\@glxtr@org@glxtrtitlefull</code> ..	309, 310	8, 10, 44, 112
<code>\@glxtr@org@glxtrtitlefullpl</code> ..	309, 310	<code>\@glxtr@record@setting@off</code>	49
<code>\@glxtr@org@glxtrtitlelong</code> ..	309, 310	<code>\@glxtr@record@setting@only</code>	112
<code>\@glxtr@org@glxtrtitlelongpl</code> ..	309, 310	<code>\@glxtr@recordsee</code>	14, 25, 44
<code>\@glxtr@org@glxtrtitlename</code> ..	309, 310	<code>\@glxtr@redef@forglsentries</code>	7, 26
<code>\@glxtr@org@glxtrtitleorpdforheading</code>	309, 310	<code>\@glxtr@redefstyles</code>	21, 22, 327
<code>\@glxtr@org@glxtrtitleplural</code> ..	309, 310	<code>\@glxtr@reg@glosslist</code>	113–116, 119
<code>\@glxtr@org@glxtrtitleshort</code> ..	309, 310	<code>\@glxtr@restore@postpunc</code>	117, 118
<code>\@glxtr@org@glxtrtitleshortpl</code> ..	309, 310	<code>\@glxtr@rglstrigger@record</code> ...	146, 147
<code>\@glxtr@org@glxtrtitletext</code> ..	309, 310	<code>\@glxtr@save@longnewglossaryentry</code> ...	37
<code>\@glxtr@org@makeglossaries</code>	112	<code>\@glxtr@save@preloctag</code>	56, 57
<code>\@glxtr@org@markboth</code>	307, 308	<code>\@glxtr@setentrycountunsetattr</code>	97
<code>\@glxtr@org@markright</code>	307, 308	<code>\@glxtr@setentryunitcountunsetattr</code> ..	109
<code>\@glxtr@org@newacronymstyle</code>	111	<code>\@glxtr@setupshortcuts</code>	20, 21, 26
<code>\@glxtr@org@postdescription</code> ..	118, 185	<code>\@glxtr@shortcutsnr</code>	20
<code>\@glxtr@org@see@noindex</code>	131	<code>\@glxtr@shortcutsval</code>	20, 133
<code>\@glxtr@org@setacronymstyle</code>	111	<code>\@glxtr@swaptwo</code>	190
<code>\@glxtr@org@theHvalue</code>	8, 9	<code>\@glxtr@tag</code>	183
<code>\@glxtr@org@unset@buffer</code>	95, 96	<code>\@glxtr@taggingcs</code>	183
<code>\@glxtr@org@prefix</code>	10, 11	<code>\@glxtr@textformat</code>	63, 64
<code>\@glxtr@org@printglossary</code>	53, 118	<code>\@glxtr@theHvalue</code>	8–10, 61–65, 145
<code>\@glxtr@org@warndep</code>	190	<code>\@glxtr@thevalue</code> ...	8–10, 61–65, 144, 145
<code>\@glxtr@p@acrlong@</code>	86	<code>\@glxtr@thisloctag</code>	56, 57
<code>\@glxtr@p@acrlongpl@</code>	86	<code>\@glxtr@titlelabel</code>	121, 122, 139
<code>\@glxtr@p@acrshort@</code>	86	<code>\@glxtr@tmp</code>	22, 62, 124
<code>\@glxtr@p@acrshortpl@</code>	86	<code>\@glxtr@type</code>	169
<code>\@glxtr@p@long@</code>	86	<code>\@glxtr@unitcountlist</code>	104
<code>\@glxtr@p@longpl@</code>	86	<code>\@glxtr@unset</code>	95, 96
<code>\@glxtr@p@plural@</code>	86	<code>\@glxtr@unset@buffer</code>	95, 96
<code>\@glxtr@p@short@</code>	86	<code>\@glxtr@unsrt@getgrouptitle</code>	137
<code>\@glxtr@p@shortpl@</code>	86	<code>\@glxtr@usesee</code>	43
<code>\@glxtr@p@text@</code>	86	<code>\@glxtr@warn@onexistsordo</code>	7, 14
<code>\@glxtr@pagetag</code>	56	<code>\@glxtr@warn@undefaction</code>	7, 14
<code>\@glxtr@pagetag</code>	56	<code>\@glxtr@wrglossary@locationhyperlink</code>	24
<code>\@glxtr@prevunitcount</code>	106	<code>\@glxtr@wrglossnr</code>	61
<code>\@glxtr@printglossnr</code>	118	<code>\@glxtr@wrglossval</code>	61

\@glxtrbuffer@nodup@unset	95	\@newglossaryentryprehook	12, 13, 28, 37, 45, 77
\@glxtrbuffer@unset	95	\@nll	125, 140
\@glxtrdialecthook	327	\@nnll	179, 180, 182, 189, 191, 192
\@glxtrdocdeffalse	49	\@no@glxtrindexaliased	81
\@glxtrentryfmt	30	\@no@makeglossaries	130
\@glxtrfmt	29	\@nocounterr	150
\@glxtrglossentry	134	\@nopostdesc	117
\@glxtrglossentryother	135	\@onelevel@sanitize	12, 44, 53, 122, 139
\@glxtrhypernameprefix	118, 119, 139	\@onlypreamble	53, 56, 108, 131, 134, 150, 178, 180, 181, 183
\@glxtrifhasfield	32	\@org@glossaryentrynumbers	116, 117
\@glxtrifhyphenstart	290	\@org@newglossaryentryprehook	37
\@glxtrindexaliased	80	\@print@unsrt@glossary	136
\@glxtrindexcrossrefsfalse	15	\@printgloss@setsort	115, 116
\@glxtrindexcrossrefstrue	15	\@printglossary	53, 136
\@glxtrinmark	307, 308	\@printunsrt@glossary@handler	137
\@glxtrlong	86, 202	\@printunsrtglossary	136
\@glxtrlongpl	86, 206	\@rGLS	147
\@glxtrnewgls	142, 143	\@rGLS@	147
\@glxtrnewgls@inner	141, 142	\@rGLSpl	147
\@glxtrnewgls@innercsname	142	\@rGLSpl@	147
\@glxtrnotinmark	307, 308	\@rGls	146
\@glxtrp	91, 92	\@rGls@	146
\@glxtrp@opt	89	\@rGlspl	146
\@glxtrpl	52, 53	\@rGlspl@	147
\@glxtrpostloctag	55, 56, 58	\@rgls	145
\@glxtrpreloctag	55, 56, 58	\@rgls@	145
\@glxtrsetaliasnoindex	80, 81	\@rglspl	146
\@glxtrshort	86, 200	\@rglspl@	146
\@glxtrshortpl	86, 204	\@sGlsXtrEnableOnTheFly	50
\@glxtrundeftag	6, 27	\@secondofthree	66–68, 73, 75–78, 198, 199, 201, 203, 205, 207
\@glxtrwrglossmark	24	\@secondoftwo	60, 65, 69–77, 79, 85, 112, 118, 176, 190, 197, 198, 200–207, 221, 244, 258, 279, 308, 310
\@gobble	7, 13, 16, 66, 138, 139, 191, 401–403	\@sglsxtr@provide@storagekey	28
\@gobbletwo	190	\@starttoc	308
\@ifnextchar	82	\@thirdofthree	66–68, 74–77, 79, 198, 200, 202, 203, 205, 207, 309
\@ifpackageloaded	5, 17, 133, 150, 170, 171, 174, 176, 178, 327, 330, 361	\@thirdoftwo	69–73
\@ifstar	28, 29, 32, 37–39, 50, 82, 95, 136, 183	\@this@key	176
\@ifundefined	326	\@warn@nomakeglossaries	126
\@ignored@glossaries	38–40	\@xdy@main@language	126
\@input	131	\@xdycrossrefhook	44
\@input@	126	\@xdy@language	126
\@istfilename	113	\@xdy@locationclassorder	44
\@makeglossary	113	\\	125
\@mfu@domakefirstuc	184		
\@mfu@nocaplist	184		
\@ne	99, 108, 141		
\@newglossaryentry@defcounters	98, 105		
\@newglossaryentryposthook	12, 13, 28, 45, 77		

<code>_</code>	50, 128, 129	<code>\Acp</code>	19
		<code>\acp</code>	18
		<code>\ACRfullfmt</code>	110
		<code>\Acrfullfmt</code>	110
		<code>\acrfullfmt</code>	110
		<code>\ACRfullplfmt</code>	110
		<code>\Acrfullplfmt</code>	110
		<code>\acrfullplfmt</code>	110
		<code>\acronymentry</code>	110
		<code>\acronymfont</code>	73–77, 88, 111
		<code>\acronymname</code>	17
		<code>\acronymsort</code>	110
		<code>\acronymtype</code>	17, 110, 111
		<code>\acrpluralsuffix</code>	110, 132
		<code>\ACS</code>	19
		<code>\Acs</code>	19
		<code>\acs</code>	18
		<code>\ACSP</code>	19
		<code>\Acsp</code>	19
		<code>\acsp</code>	19
		<code>\actualchar</code>	181
		<code>\addtolength</code>	392
		<code>\advance</code>	99, 108, 131, 141
		<code>\AF</code>	18
		<code>\Af</code>	18
		<code>\af</code>	18
		<code>\AFP</code>	18
		<code>\Afp</code>	18
		<code>\afp</code>	18
		<code>\AL</code>	18
		<code>\Al</code>	18
		<code>\al</code>	18
		<code>\ALP</code>	18
		<code>\Alp</code>	18
		<code>\alp</code>	18
		<code>amsmath package</code>	23
		<code>\AnyTrackedLanguages</code>	327, 361
		<code>\appto</code>	12, 13, 22, 28, 42, 44–47, 77, 82, 98, 105, 137, 139, 178, 189, 191, 192, 363
		<code>\arabic</code>	23, 403
		<code>\AS</code>	18
		<code>\As</code>	18
		<code>\as</code>	18
		<code>\ASP</code>	18
		<code>\Asp</code>	18
		<code>\asp</code>	18
		<code>\AtBeginDocument</code>	24, 27, 54, 55, 133
		<code>\AtEndDocument</code>	47, 99, 107, 126
A			
<code>\AA</code>	345		
<code>\aa</code>	345		
<code>\AB</code>	18		
<code>\Ab</code>	18		
<code>\ab</code>	18		
abbreviation styles:			
long-hyphen-postshort-hyphen	295, 296, 298		
long-hyphen-short-hyphen	292, 296		
long-postshort-user	284		
long-short-user	283		
nolong-short	227		
short	224		
short-hyphen-long-hyphen	300, 301		
short-hyphen-postlong-hyphen	301, 303		
short-long-user	285		
short-nolong	224, 226		
short-nolong-desc	226		
short-postlong-user	287		
<code>\abbreviationsname</code>	17		
<code>\abbrvpluralsuffix</code>			
.....	132, 159, 193, 215, 217, 219, 221, 223, 225, 228, 231, 233, 235, 236, 238, 240, 242, 244, 246, 247, 249, 250, 252, 254, 256, 258, 260, 261, 263, 265, 266, 268, 270, 272, 273, 275, 277, 279, 282, 283, 286, 289, 292, 293, 297, 300, 302, 305		
<code>\ABP</code>	18		
<code>\Abp</code>	18		
<code>\abp</code>	18		
<code>\AC</code>	19		
<code>\Ac</code>	19		
<code>\ac</code>	18		
<code>\ACF</code>	19		
<code>\Acf</code>	19		
<code>\acf</code>	19		
<code>\ACFP</code>	19		
<code>\Acfp</code>	19		
<code>\acfp</code>	19		
<code>\ACL</code>	19		
<code>\Acl</code>	19		
<code>\acl</code>	19		
<code>\ACLP</code>	19		
<code>\Aclp</code>	19		
<code>\aclp</code>	19		
<code>\ACP</code>	19		

B	
babel package	178, 180, 188
\begin	123, 128, 137, 365, 367–374, 377, 379, 394–398, 401
\begingroup	8, 9, 29, 30, 81, 134–136, 149
\bgroup	37, 116
bib2gls	23, 30, 139, 144, 145, 326–329
C	
\c@wrglossary	23
\catcode	48, 131
category attributes:	
accessinsertdots	159
aposplural	193
discardperiod	188
entrycount	94, 97–99, 109
firstshortaccess	160
firstuc	174
glossdesc	170
glossdescfont	170
glossname	171
glossnamefont	172, 174
headuc	310
indexname	179
indexonlyfirst	81
insertdots	193
linkcount	149
linkcountmaster	149
markshortwords	193
markwords	192, 194, 290, 291, 299
nameshortaccess	159
nohyper	80
nohyperfirst	66–68
noshortplural	193
regular	58, 103, 214, 216, 218, 219, 221, 223– 227, 229, 230, 232–235, 237, 239, 241, 243, 247, 248, 250, 251, 253, 255–257, 261–268, 271–274, 276, 277, 279, 282, 288, 289, 291–293, 295, 299, 300, 305, 306
textformat	63
textshortaccess	160
\cdot	24
\centering	400
\cGLS	18, 19, 97, 109
\cGls	18, 19, 97, 109
\cglS	18, 97, 109
\cGLSformat	101
\cGlsformat	101
\cglSformat	100, 102
\cGLSpl	18, 19, 97, 109
\cGlspl	18, 19, 97, 109
\cglSpl	18, 97, 109
\cGLSplformat	101
\cGlsplformat	101
\cglSplformat	100, 103
\changes	313, 377, 379
\char	121
\columnwidth	54, 55
\count@	99, 100, 108
\csappto	36
\csdef	28, 33, 34, 36, 77–79, 99, 104, 105, 107, 164, 176, 186, 211–213, 221, 243, 257, 279, 283–285, 287, 296, 298, 302, 303, 364, 382
\cseappto	40
\csedef	105
\csgdef	34, 38–40, 49, 56, 99, 104, 107, 382, 404
\cslet	34, 37, 117
\csletcs	34, 212, 213
\csname	6, 29, 39, 44, 49, 54, 57, 60, 62, 65, 73–79, 81, 89, 105, 113, 114, 122, 126, 129, 130, 137, 142, 143, 145, 149, 170, 190, 197–207, 213, 391
\cspreto	36
\csuse	9, 30, 39, 47, 56, 78, 79, 91–93, 103–107, 116, 121, 123, 124, 134, 135, 138–140, 164, 175, 185–187, 211, 213, 382, 383, 385, 386
\csxdef	42, 45, 46, 104, 107
\currentglossary	117, 134, 135, 403
\CurrentOption	24, 363, 364
\CurrentTrackedLanguage	361
\CurrentTrackedLanguageTag	132
\CurrentTrackedScript	361
\CurrentTrackedTag	326, 361
\CustomAbbreviationFields	194, 214– 216, 218, 219, 221, 223, 224, 227, 230– 234, 236, 238, 241, 243, 245–250, 252, 255, 257, 259–264, 266, 268, 270, 271, 275, 277, 279, 281, 283–285, 287–289, 291–293, 295, 296, 298–301, 303, 304, 306
D	
\DeclareAcronymList	110
\DeclareOption	5, 363
\DeclareOptionX	5, 24
\def	9, 10, 12–15, 25, 27, 29, 33, 37, 39, 43–45, 48, 51–53,

55, 57, 59–77, 83, 85–89, 95, 100–102, 110, 114–116, 118–125, 137, 139, 140, 142, 144–147, 158, 159, 179–184, 189– 193, 197–207, 211, 290, 293, 295, 296, 299, 301, 361, 375, 376, 391–393, 397, 398	\empty 123–125
\defglentryfmt 38–40	\encapchar 182
\define@boolkey 15, 16, 61, 80	\end 119, 123, 128, 137, 365, 367–374, 394–398, 401
\define@choicekey 7, 13, 15, 16, 20, 21, 24, 61, 118	\end@glxtr@display@loc 123
\define@key 12, 13, 16, 21, 22, 28, 45, 61, 64, 77, 118, 119, 158, 190	\endcsname 6, 29, 39, 44, 49, 54, 57, 60, 62, 65, 73–79, 81, 89, 105, 113, 114, 122, 126, 129, 130, 137, 142, 143, 145, 149, 170, 190, 197–207, 213, 391
\DefineAcronymSynonyms 20	\endgroup 8, 10, 30, 81, 134, 136, 149
\delimN 56, 57	\ensurermath 24
\delimR 56, 57	entry categories:
\detokenize 51	abbreviation 207
\dimen@ 112, 382–390	general 164, 166
\dimen@i 385, 386	index 168
\dimen@ii 382, 383, 385, 386	\epreto 179
\dimexpr 54, 55, 381	\equal 129, 130, 186
\disable@keys 17, 27, 49, 131	etoolbox package 5
\do 6, 22, 32, 48, 65, 97, 98, 109, 113, 116, 122, 137, 144, 150, 169, 176, 183	\expandafter 24, 29, 30, 32, 43, 47, 48, 51–53, 62, 78, 79, 82, 83, 89, 95, 102, 103, 113–115, 119, 122, 124, 125, 137, 140, 142, 148, 159, 170, 173, 174, 176, 177, 179, 181, 182, 189, 193, 194, 290, 401
\do@gl@link@checkfirsthyper 30, 59, 60, 63, 73–77, 197–207	\expandonce 110, 140, 159, 160, 179, 180, 215, 293
\do@gl@disablehyperinlist 63, 80	\ExtraCustomAbbreviationFields 159, 160, 192, 194
doc package 181	
\dolistcsloop 31	
\DTLifinlist 113–115, 119	
\DTLifint 121	
E	
\eappto 11, 22, 38–40, 137, 140, 159, 160, 179, 363	
\edef 6, 8–11, 38–41, 44, 46–49, 62, 63, 65, 79, 81, 83, 85, 103– 105, 107, 113, 114, 119, 121, 124–126, 131, 134, 135, 144, 145, 149, 170–177, 179, 180, 182, 190, 361, 385, 386, 401, 402	
\eglssetwidest 384–390	
\egroup 37, 117	
\else 8–12, 15–17, 20, 21, 25, 30, 33, 35, 36, 49, 51, 55, 57, 60, 63, 64, 81, 82, 100, 112, 113, 115–118, 121, 123–125, 128, 130, 131, 144, 145, 178–180, 182, 189, 191, 192, 194, 200–207, 210, 215, 217, 219–229, 232–246, 248–260, 262– 265, 267–280, 282–284, 286, 289–291, 293, 296, 298, 299, 301, 303, 305, 306, 365, 367–378, 380, 391, 392, 398, 400, 402	
\emph 259	
F	
\fi 7–12, 14–16, 18, 20, 21, 24–26, 30, 33, 35, 36, 42, 44, 45, 47, 49–51, 54–56, 58, 60, 61, 64, 81, 82, 100, 107, 108, 112, 115–119, 121, 124– 126, 128–131, 133, 144, 145, 178–180, 182, 189, 191, 192, 194, 200–207, 210, 215, 217, 219–229, 231–246, 248–260, 262–265, 267–280, 282–284, 286, 289– 291, 293, 296–299, 301, 303, 305, 306, 365, 368–378, 380, 382–392, 398, 400, 403	
first use 405	
flag 405	
text 405	
\firstacronymfont 111, 112	
fontspec package 133	
\footnote 218	
\forall glossaries 47, 136, 167–169, 384–390	
\forall glsentries 99, 108	
\ForEachTrackedDialect 327, 361	
\forglsentries 6, 47, 167–169, 384–390	

\forlistcsloop 31, 108, 123
 \forlistloop 96, 120, 184
 \futurelet 189

G

\gdef 56, 57, 180, 181
 \Genacrfullformat 110, 111
 \genacrfullformat 110
 \GenericAcronymFields 110
 \Genplacrfullformat 110, 111
 \genplacrfullformat 110, 111
 \glo@grabfirst 140
 \glo@name 172–174, 177
 \gloaliaslabel 84
 \global 10, 37, 117, 140
 \glolinkprefix ... 61, 62, 64, 84, 85, 119, 133
 glossaries package
 14, 25, 26, 35, 42, 44–46, 116, 364
 glossaries-accsupp package 21, 24, 150, 194
 glossaries-extra package 2, 361
 glossaries-extra-bib2gls package 14, 27, 327, 361
 glossaries-extra-stylemods package . 21, 185, 327
 glossaries-stylemods package 399
 glossaries.sty package 37
 \GlossariesExtraWarning 6,
 16, 36, 51, 53, 63, 111, 114, 124, 128,
 130, 131, 136, 170–175, 177, 183, 184, 213
 \GlossariesExtraWarningNoLine 16, 100, 108
 \GlossariesWarning 56, 114, 116, 120, 121, 211
 \GlossariesWarningNoLine 114, 126
 glossary styles:
 altlist 365
 altlistgroup 366
 altlisthypergroup 367
 almtree 381, 382, 391
 almtreegroup 392
 almtreehypergroup 393
 index 376
 indexgroup 377
 indexhypergroup 377
 inline 375
 list 365
 listdotted 364
 listdottedstyle 365
 listgroup 366
 listhypergroup 366
 mcolalmtree 396
 mcolalmtreegroup 397
 mcolalmtreehypergroup 397

mcolalmtreespannav 398
 mcolindexgroup 393
 mcolindexhypergroup 393
 mcolindexspannav 394
 mcoltreegroup 394
 mcoltreehypergroup 395
 mcoltreenonamegroup 395
 mcoltreenonamehypergroup 396
 mcoltreenonamespannav 396
 mcoltreespannav 395
 sublistdotted 365
 tree 377
 treegroup 378
 treehypergroup 379
 treenoname 379
 treenonamegroup 380
 treenonamehypergroup 380
 glossary-bookindex package 364
 glossary-hypernav package 83
 glossary-long package 369
 glossary-longbooktabs package 369
 glossary-tree package 375, 376
 \glossaryentrynumbers 57, 116, 117, 140, 141
 \glossaryheader 123,
 137, 365–374, 376–380, 391, 393–397, 401
 \glossaryname 116
 \glossarypostamble 123, 137, 139
 \glossarypreamble 123, 136
 \glossarysection ... 123, 129, 130, 136, 139
 \glossarytitle 39, 116, 117, 123, 129, 130, 136
 \glossarytoctitle 39, 116, 123, 129, 130, 136
 \glossentry
 117, 141, 364–374, 376, 378, 380, 391, 401
 \glossentrydesc
 364, 365, 367–374, 377–379, 381
 \glossentryname
 134, 364–374, 376, 378, 380, 391, 392, 399
 \glossentrynameother 136
 \glossentrysymbol 368–372, 374, 378, 379, 381
 \glossxtrsetpopts 185
 \glostyle 377, 379
 \GLS 97, 109, 144
 \Gls 52, 97, 109, 144
 \gls 35, 36, 52, 53, 97, 109, 114, 127, 144
 \gls@assign@desc 37
 \gls@assign@field 12, 13, 28, 77
 \gls@checkseeallowed 50, 113
 \gls@codepage 126
 \gls@defdocnewglossaryentry .. 49, 98, 106

<code>\gls@defglossaryentry</code>	37, 51–53	<code>\glsaccesslong</code>	75, 76, 195, 202, 203, 215, 217, 219, 220, 222, 223, 225– 229, 231, 233–244, 246, 248, 249, 251– 258, 260, 262–265, 267–276, 278, 280, 282–284, 286, 289, 292, 294, 297, 300, 305
<code>\gls@dotocitle</code>	116, 117	<code>\Glsaccesslongpl</code>	77, 195, 207, 215, 223, 228, 229, 232, 238–241, 246, 252– 255, 260, 262, 269–272, 274–276, 282, 284, 292, 294, 295, 297, 298, 300, 305, 306
<code>\gls@glossary</code>	44	<code>\glsaccesslongpl</code>	76, 77, 195, 206, 207, 215, 217, 220, 222, 223, 225–229, 232–246, 248, 249, 251–260, 262, 264, 265, 267–276, 278, 280, 282– 284, 286, 289, 292, 294, 297, 298, 300, 305
<code>\gls@grplabel</code>	83	<code>\GLSaccessname</code>	69
<code>\gls@ifnotmeasuring</code>	96, 97	<code>\Glsaccessname</code>	69
<code>\gls@level</code>	140	<code>\glsaccessname</code>	43, 69
<code>\gls@noidxglossary</code>	114	<code>\GLSaccessplural</code>	68
<code>\gls@org@glossaryentryfield</code>	117	<code>\Glsaccessplural</code>	67
<code>\gls@org@glossarysubentryfield</code>	117	<code>\glsaccessplural</code>	67
<code>\gls@orgTrackLangRequireDialectPrefix</code>	361	<code>\Glsaccessshort</code>	74, 201, 210, 217, 220, 222, 225, 227, 233, 235, 236, 242– 244, 248, 249, 251, 256–258, 264, 265, 267, 268, 278, 280, 286, 289, 297, 302, 303
<code>\gls@save@numberlist</code>	55, 57, 58	<code>\glsaccessshort</code>	73, 74, 195, 200, 202, 210, 215, 217, 219–226, 228, 232, 233, 235–242, 244, 246, 248–256, 258, 260, 262, 263, 265, 267–272, 274, 276–280, 282, 284, 286, 289, 292, 294, 297, 300, 302, 303, 305
<code>\gls@set@xr@key</code>	45	<code>\Glsaccessshortpl</code>	75, 205, 210, 217, 220, 222, 225, 227, 234, 235, 237, 242–245, 248, 249, 251, 256–259, 264, 265, 267, 268, 278, 280, 286, 289, 297, 302, 303, 306
<code>\gls@tmplen</code>	384–390, 392	<code>\glsaccessshortpl</code>	74, 75, 195, 204, 205, 210, 215, 217, 220, 222– 229, 232, 233, 235–242, 244, 246, 248– 253, 255–258, 260, 262, 263, 265, 267– 272, 274, 276, 278–280, 282, 284, 286, 289, 292, 294, 297, 300, 302, 303, 305, 306
<code>\gls@type</code>	114	<code>\GLSaccesssymbol</code>	70
<code>\glsabbrvdefaultfont</code> ...	196, 215, 217, 219, 221, 223, 225, 228, 281, 291, 294, 304	<code>\Glsaccesssymbol</code>	70, 183
<code>\glsabbrvmfont</code>	259–266, 268, 270, 272, 273, 275, 277, 279	<code>\glsaccesssymbol</code>	70, 182, 187
<code>\glsabbrvfont</code>	87, 88, 111, 196, 200–202, 204, 205, 208, 210, 214–219, 221–225, 228, 229, 231, 233, 235, 236, 238, 240, 242, 244, 246, 247, 249, 250, 252, 254, 256, 258, 260, 261, 263, 265, 266, 268, 270, 272, 273, 275, 277, 279, 282, 283, 286, 288, 289, 292, 294, 296, 297, 300, 302, 305	<code>\GLSaccesssymbolplural</code>	71
<code>\glsabbrvhyphenfont</code>	291, 292, 296–303	<code>\Glsaccesssymbolplural</code>	71
<code>\glsabbrvonlyfont</code>	304–306	<code>\glsaccesssymbolplural</code>	71
<code>\glsabbrvscfont</code>	231–236, 238, 240–244	<code>\GLSaccesssymbolplural</code>	71
<code>\glsabbrvsmfont</code> .	245–250, 252, 254, 256–258	<code>\GLSaccesssymbolplural</code>	71
<code>\glsabbrvuserfont</code>	281–289	<code>\GLSaccesssymbolplural</code>	71
<code>\GLSaccessdesc</code>	69	<code>\GLSaccesssymbolplural</code>	71
<code>\Glsaccessdesc</code>	69, 170, 182	<code>\GLSaccesssymbolplural</code>	71
<code>\glsaccessdesc</code>	69, 171, 187	<code>\GLSaccesssymbolplural</code>	71
<code>\GLSaccessdescplural</code>	70	<code>\GLSaccesssymbolplural</code>	71
<code>\Glsaccessdescplural</code>	70	<code>\GLSaccesssymbolplural</code>	71
<code>\glsaccessdescplural</code>	70	<code>\GLSaccesssymbolplural</code>	71
<code>\GLSaccessfirst</code>	67	<code>\GLSaccesssymbolplural</code>	71
<code>\Glsaccessfirst</code>	67	<code>\GLSaccesssymbolplural</code>	71
<code>\glsaccessfirst</code>	67	<code>\GLSaccesssymbolplural</code>	71
<code>\GLSaccessfirstplural</code>	68	<code>\GLSaccesssymbolplural</code>	71
<code>\Glsaccessfirstplural</code>	68	<code>\GLSaccesssymbolplural</code>	71
<code>\glsaccessfirstplural</code>	68	<code>\GLSaccesssymbolplural</code>	71
<code>\Glsaccesslong</code>	76, 195, 203, 215, 223, 228, 232, 238–240, 246, 252–255, 260, 262, 269–272, 274– 276, 282–284, 292, 294, 297, 298, 300, 305	<code>\GLSaccesssymbolplural</code>	71

<code>\glsacrshortcutstrue</code>	20, 21	<code>\Glsentrydesc</code>	155, 162, 171
<code>\glsacspacemax</code>	112	<code>\glsentrydesc</code>	154, 155, 162, 171
<code>\glsadd</code>	30, 48, 65, 127	<code>\Glsentrydescplural</code>	155, 163
<code>\glsadd options</code>		<code>\glsentrydescplural</code>	155, 162, 163
<code>theHvalue</code>	9	<code>\Glsentryfirst</code>	103, 148, 152, 161
<code>thevalue</code>	9, 10	<code>\glsentryfirst</code> ..	103, 148, 152, 153, 161, 323
<code>\glsaddpostsetkeys</code>	10, 65	<code>\Glsentryfirstplural</code> ...	103, 148, 153, 162
<code>\glsaddpresetkeys</code>	10, 65	<code>\glsentryfirstplural</code>	103, 148, 153, 162, 323
<code>\glsaddstoragekey</code>	46, 164, 329	<code>\glsentryfmt</code>	38–40
<code>\glsbackslash</code>	51	<code>\Glsentryfull</code>	111
<code>\glscapscase</code>	60, 65–79, 197–209	<code>\glsentryfull</code>	110
<code>\glscategory</code>		<code>\Glsentryfullpl</code>	111
.....	58, 66, 79, 87, 88, 165–167, 170–176, 182, 183, 185–187, 197–202, 204, 205	<code>\glsentryfullpl</code>	111
<code>\glscategorylabel</code>		<code>\glsentryitem</code>	
..	79, 158–160, 190, 192–194, 221, 243, 257, 279, 283–285, 287, 296, 298, 302, 303		134, 136, 364–374, 376, 378, 380, 391, 402
<code>\glsclosebrace</code>	44, 128–130	<code>\Glsentrylong</code>	88, 89, 103, 148, 157, 163
<code>\glscounter</code>	9	<code>\glsentrylong</code>	88, 89, 103, 148, 157, 163, 221, 243, 257, 279, 285, 287, 301, 324
<code>\glscurrentrylabel</code>	55–57, 117, 125, 134, 135, 137, 138, 184, 185	<code>\Glsentrylongpl</code>	88, 89, 103, 157, 163
<code>\glscurrentfieldvalue</code>		<code>\glsentrylongpl</code>	
.....	30, 32, 33, 35, 280, 281, 328	88, 89, 103, 157, 163, 164, 324, 325
<code>\glscustomtext</code> ..	59, 60, 73–77, 197–208, 210	<code>\Glsentrylongplural</code>	148
<code>\glsdefaulttype</code>		<code>\glsentrylongplural</code>	148
.....	6, 17, 36, 115, 116, 127, 136, 138	<code>\Glsentryname</code>	151, 161, 172–175
<code>\glsdescriptionaccessdisplay</code>	154, 155, 170	<code>\glsentryname</code>	134, 150, 151, 161, 179, 321, 384–390, 404
<code>\glsdescriptionpluralaccessdisplay</code>	155	<code>\glsentrynumberlist</code>	114, 121, 388–390
<code>\glsdescwidth</code>	367–374	<code>\Glsentryplural</code>	152, 161
<code>\glsdetoklabel</code>	8, 9, 31–37, 41, 43, 47–49, 51, 62, 65, 81, 84, 98, 99, 104–108, 113, 117, 119–121, 134, 135, 140, 143, 144, 170, 172–174, 177, 385, 386	<code>\glsentryplural</code>	152, 161, 322
<code>\glsdisplaynumberlist</code>	114, 119	<code>\glsentryprevcount</code>	98, 100, 106
<code>\glsdohyperlink</code>	83, 85, 86	<code>\glsentryprevmaxcount</code>	106
<code>\glsdohypertarget</code>	85, 118	<code>\glsentryprevtotalcount</code>	106
<code>\glsdoifexists</code>	15, 25, 33, 40, 43, 44, 58, 60, 65, 73–77, 85, 97, 113, 120, 121, 134, 135, 197–207	<code>\Glsentryshort</code>	87, 88, 156, 163
<code>\glsdoifexistsordo</code>	29, 30, 60	<code>\glsentryshort</code> ..	87, 88, 112, 156, 163, 283, 285, 296, 320
<code>\glsdoifexistsorwarn</code>	15, 170, 171, 182, 183	<code>\Glsentryshortpl</code>	87, 88, 156, 163
<code>\glsdoifnoexists</code>	37	<code>\glsentryshortpl</code>	
<code>\glsdonohyperlink</code>	64, 85	87, 88, 156, 157, 163, 320, 321
<code>\glsdosanitizesort</code>	115	<code>\Glsentrysymbol</code>	154, 162
<code>\glsenableentrycount</code>	97, 100, 108	<code>\glsentrysymbol</code>	153, 154, 162, 387–389
<code>\glsenableentryunitcount</code>	99, 108, 109	<code>\Glsentrysymbolplural</code>	154, 162
<code>\glsentrycounter</code>	23, 125	<code>\glsentrysymbolplural</code>	154, 162
<code>\GlsEntryCounterLabelPrefix</code>	35	<code>\Glsentrytext</code>	151, 161
<code>\glsentrycurrcount</code>	98, 99, 106	<code>\glsentrytext</code>	85, 151, 161, 321, 322
		<code>\glsentrytype</code>	135
		<code>\Glsentryuseri</code>	71
		<code>\glsentryuseri</code>	71
		<code>\Glsentryuserii</code>	71
		<code>\glsentryuserii</code>	72

<code>\Glsentryuseriii</code>	72	<code>\glsfirstplural</code>	315, 316
<code>\glsentryuseriii</code>	72	<code>\glsfirstpluralaccessdisplay</code>	153
<code>\Glsentryuseriv</code>	72	<code>\glsforeachincategory</code>	211
<code>\glsentryuseriv</code>	72	<code>\glsgenentryfmt</code>	58
<code>\Glsentryuserv</code>	72	<code>\glsgetattribute</code>	63, 84, 100, 104–106, 125, 144, 149, 170–175, 177, 178
<code>\glsentryuserv</code>	73	<code>\glsgetcategoryattribute</code>	165
<code>\Glsentryuservi</code>	73	<code>\glsgetgrouptitle</code>	366, 367, 377, 379–381, 392–398
<code>\glsentryuservi</code>	73	<code>\glsgetwidestname</code>	382
<code>\glsextrapostnamehook</code>	175	<code>\glsgroupheading</code>	140, 365–374, 376–381, 391–398, 403
<code>\glsfieldxdef</code>	84	<code>\glsgroupskip</code>	140, 365, 367–375, 377, 378, 380, 392, 403
<code>\glsfindwidesttoplevelname</code>	383	<code>\glshasattribute</code>	63, 84, 99, 100, 104, 105, 107, 108, 125, 144, 149, 170–176, 178, 214, 216– 219, 221, 224, 226, 227, 229–234, 242, 243, 245, 247, 248, 256, 257, 259, 261– 266, 273, 277, 279, 282, 283, 285, 287– 289, 291–293, 295, 297–300, 302, 304–306
<code>\GLSfirst</code>	315	<code>\glshascategoryattribute</code>	165
<code>\Glsfirst</code>	315	<code>\glshex</code> ..	332–337, 340–346, 348–351, 353–361
<code>\glsfirst</code>	315	<code>\glshyperlink</code>	85, 328
<code>\glsfirstabbrvdefaultfont</code>	196, 215, 217, 219, 221, 223, 225, 228, 294	<code>\glshypernavsep</code>	122
<code>\glsfirstabbrvmfont</code>	259–280	<code>\glshypernumber</code>	125, 178
<code>\glsfirstabbrvfont</code>	111, 195, 214–229, 231, 233, 235, 236, 238, 240, 242, 244, 246, 247, 249, 250, 252, 254, 256, 258, 260, 261, 263, 265, 266, 268, 270, 272, 273, 275, 277, 279, 282, 283, 286, 289, 292, 294, 297, 300, 302, 305	<code>\glsifattribute</code> ..	60, 61, 66, 80, 82, 91, 149, 168, 170–173, 177, 184, 188, 310–319
<code>\glsfirstabbrvhyphenfont</code>	291, 292, 296, 297, 299–303	<code>\glsifcategory</code>	167
<code>\glsfirstabbrvonlyfont</code>	305, 306	<code>\glsifcategoryattribute</code>	79, 158–160, 166, 192–194
<code>\glsfirstabbrvscfont</code>	231–245	<code>\glsifnotregular</code>	66
<code>\glsfirstabbrvsmfont</code>	245–259	<code>\glsifnotregularcategory</code>	167
<code>\glsfirstabbrvuserfont</code>	282–289	<code>\glsifplural</code>	60, 65, 67, 68, 70, 71, 73–77, 188, 197–209
<code>\glsfirstaccessdisplay</code>	152, 153	<code>\glsifregular</code>	58, 66, 103, 148
<code>\glsfirstlongdefaultfont</code>	215, 217, 223, 225, 228, 231–241, 245– 255, 259, 260, 263, 264, 266–271, 273–275	<code>\glsifregularcategory</code>	167
<code>\glsfirstlongemfont</code>	261, 262, 265, 266, 271–273, 275, 276	<code>\glsifusetranslator</code>	39
<code>\glsfirstlongfont</code>	195, 214–219, 221, 223, 225–231, 233, 235, 236, 238, 240, 242, 244, 246, 247, 249, 250, 252, 254, 256, 258, 260, 262, 263, 265, 266, 268, 270, 272, 273, 275, 277, 279, 282, 283, 286, 289, 292, 294, 297, 300, 302, 305	<code>\glsignore</code>	56, 57
<code>\glsfirstlongfootnotefont</code>	219–222, 241–245, 255–259, 277–280	<code>\glsinlinedescformat</code>	375
<code>\glsfirstlonghyphenfont</code>	291–302	<code>\glsinlinesubdescformat</code>	375
<code>\glsfirstlongonlyfont</code>	304–306	<code>\glsinsert</code>	60, 65, 73–77, 197–210, 290, 296–298, 302–304
<code>\glsfirstlonguserfont</code>	282–289	<code>\glskeylisttok</code>	110, 192, 194
<code>\GLSfirstplural</code>	315, 316	<code>\glslabel</code>	8, 9, 29, 41, 43, 58, 60–64, 79–81, 84, 85, 112, 144, 145, 149, 186, 187, 208–210, 221, 243, 257, 279, 283, 285, 287, 296–298, 302–304
<code>\Glsfirstplural</code>	316		

<code>\glslabeltok</code>	110,	266, 270, 271, 275, 277, 282–285, 287–
	192, 194, 214, 216–219, 221, 223–236,	289, 291–293, 295, 296, 298–300, 304, 306
	238, 241–243, 245–250, 252, 255–266,	
	268, 270, 271, 273, 275, 277, 279, 282–	<code>\glslongpluralaccessdisplay</code>
	285, 287–293, 295, 297–300, 302, 304–306	157
<code>\glsletentryfield</code>	179	<code>\glslongtok</code>
<code>\glslink</code>	110	. 110, 192–194, 214–219, 221, 223, 225,
<code>\glslink options</code>		227, 228, 230–236, 238, 241, 243, 245–
counter	10	250, 252, 255, 257, 259–266, 268, 270,
format	177	271, 275, 277, 279, 282–285, 287–289,
hyper	307	291–293, 295, 296, 298–300, 302, 304, 306
hyperoutside	61	<code>\glslonguserfont</code> ... 281–284, 286, 287, 289
noindex	8, 9, 80, 307	<code>\glsmcols</code>
textformat	63	394–398
theHvalue	63	<code>\GLSname</code>
thevalue	63, 141	312, 313
wrgloss	8, 60, 61	<code>\Glsname</code>
<code>\glslinkcheckfirsthyperhook</code>	79	313
<code>\glslinkpostsetkeys</code>	10, 63, 145	<code>\glsname</code>
<code>\glslinkpresetkeys</code>	10, 63, 145	312
<code>\glslinkvar</code>	82, 83	<code>\glsnameaccessdisplay</code> .. 150, 151, 172–174
<code>\glslistchildpostlocation</code>	365	<code>\glsnamefont</code>
<code>\glslistchildprelocation</code>	365, 366	172–175, 177
<code>\glslistdesc</code>	365, 366	<code>\glsnavhyperlink</code>
<code>\glslistdottedwidth</code>	364	122
<code>\glslistgroupheaderfmt</code>	366, 367	<code>\glsnavhyperlinkname</code>
<code>\glslistnavigationitem</code>	366, 367	83
<code>\glslistprelocation</code>	365, 366	<code>\glsnavhypertarget</code>
<code>\glslocalunset</code>	60, 145 366, 367, 377, 379, 381, 393–398
<code>\glslongaccessdisplay</code>	157	<code>\glsnavigation</code>
<code>\glslongdefaultfont</code> . 196, 215, 217, 218,	 366, 367, 377, 379, 381, 393–398
223, 225, 228, 231, 233, 235, 236, 238–		<code>\glsnextpages</code>
240, 246, 247, 249, 250, 252–254, 260,		117
263, 266, 268, 270, 273, 274, 281, 291, 304		<code>\glsnoidxdisplayloc</code>
<code>\glslongemfont</code> 259, 261, 262, 265, 271, 272, 275		120
<code>\glslongfont</code>	88, 196, 202, 203, 206–	<code>\glsnoidxdisplaylocclsthandler</code> 120
208, 215, 217, 219, 221, 223, 225, 227,		<code>\glsnoidxloclist</code>
228, 230, 231, 233, 235, 236, 238, 240,		121, 140, 141
242, 244, 246, 247, 249, 250, 252, 254,		<code>\glsnoidxnumberlistloophandler</code> 120
256, 258, 260, 262, 263, 265, 266, 268,		<code>\glsnonnextpages</code>
270, 272, 273, 275, 277, 279, 282, 283,		117
286, 289, 292, 294, 297, 300, 302, 305, 306		<code>\glsnonnumberlistfalse</code>
<code>\glslongfootnotefont</code>		55
218, 219, 221, 242, 244, 256, 258, 277, 279		<code>\glsnonnumberlisttrue</code>
<code>\glslonghyphenfont</code> . 291–297, 299, 300, 302		55
<code>\glslongonlyfont</code>	304, 305	<code>\glsnopostdotfalse</code>
<code>\glslongpltok</code>		118
.... 194, 214, 216, 218, 219, 228, 230–		<code>\glsnopostdottrue</code>
234, 238, 241, 245–248, 252, 256, 259–		117
		<code>\glsnumberlistloop</code>
		115
		<code>\glsnumlistlastsep</code>
		120
		<code>\glsnumlistsep</code>
		120
		<code>\glsopenbrace</code>
		44, 128–130
		<code>\glsorder</code>
		113
		<code>\glspagelistwidth</code>
		368, 370, 372, 374
		<code>\glspar</code>
		139
		<code>\GLSpl</code>
		97, 109, 144
		<code>\Glspl</code>
		53, 97, 109, 144
		<code>\glspl</code>
		52, 97, 109, 144
		<code>\GLSplural</code>
		314
		<code>\Glsplural</code>
		314
		<code>\glsplural</code>
		314
		<code>\glspluralaccessdisplay</code>
		152
		<code>\glspluralsuffix</code>
		132, 192, 193, 197
		<code>\glspostdescription</code>
		16, 17, 117,
		118, 185, 364, 365, 367–374, 377–379, 381

<code>\glspostinline</code>	375	<code>\glstreegroupheaderfmt</code>	
<code>\glspostlinkhook</code> .	59, 60, 73–77, 89, 197–207	377, 379–381, 392–398, 400
<code>\glsprestandardsort</code>	115	<code>\glstreeindent</code>	378, 380, 390–392
<code>\glsresetentrylist</code>	123, 137	<code>\glstreeitem</code>	376, 394, 402
<code>\glsee</code>	45–47	<code>\glstreenamebox</code>	391, 392
<code>\glseeformat</code>	43, 44, 49, 114, 120	<code>\glstreenamefmt</code>	
<code>\glseeelist</code>	44	375, 376, 378, 380, 382, 384–392
<code>\glsssetabbrvfmt</code>	58, 66, 87, 88, 170–176, 182, 183, 197–202, 204, 205, 208	<code>\glstreenavigationfmt</code>	377, 379, 381, 393–398
<code>\glsssetattribute</code>	214, 216–219, 221, 223–236, 238, 241–243, 246–250, 252, 256, 258, 260–266, 268, 270, 271, 273, 275, 277, 279, 282, 283, 285, 287, 288, 290–293, 295, 297–300, 302, 304–306	<code>\glstreenonamechilddesc</code>	380
<code>\glsssetcategoryattribute</code>	98, 109, 112, 144, 150, 165, 166, 168, 169, 183	<code>\glstreenonamedesc</code>	379, 380
<code>\glsssetnoexpandfield</code>	12, 13	<code>\glstreenonamesymbol</code>	380
<code>\glsssettoctitle</code>	116	<code>\glstreepredesc</code>	377, 379
<code>\glssshortaccessdisplay</code>	156	<code>\glstreeprelocation</code>	376, 378, 380, 381
<code>\glssshortpltok</code>	194, 214, 216, 218, 219, 221, 223, 225, 231–234, 236, 241, 243, 245–250, 255–257, 259, 261–266, 268, 277, 279, 282–285, 287– 289, 291, 292, 296, 298–301, 303, 304, 306	<code>\glstreesubitem</code>	376, 402
<code>\glssshortpluralaccessdisplay</code> ..	156, 157	<code>\glstreesubsubitem</code>	376, 403
<code>\glssshorttok</code>	110, 192–194, 214–219, 221–225, 229–234, 236, 238, 241, 243, 245–250, 252, 255, 257, 259–266, 268, 270, 271, 277, 279, 281–285, 287–289, 291, 292, 295, 296, 298–301, 303, 304, 306	<code>\glstreesymbol</code>	376, 378
<code>\glssubentryitem</code>		<code>\GLstrLetField</code>	34
.....	135, 364–374, 376, 378, 380, 392, 402	<code>\glstype</code>	60, 62, 73–77, 145, 197–207
<code>\glssymbolaccessdisplay</code>	153, 154	<code>\glsunset</code>	48, 60, 100, 101, 145
<code>\glssymbolpluralaccessdisplay</code>	154	<code>\gluserdescription</code>	282, 283, 285, 288
<code>\glstarget</code>	134, 136, 364–374, 376, 378, 380, 391, 392, 402, 403	<code>\glswrite</code>	44, 113
<code>\GLStext</code>	313	<code>\glswriteentry</code>	8, 9
<code>\GlStext</code>	313, 314	<code>\Glsxtr</code>	53
<code>\glstext</code>	313	<code>\glxtr</code>	53
<code>\glstextaccessdisplay</code>	151	<code>\glxtr@@do@wrglossary</code>	8, 10, 12, 13
<code>\glstextformat</code>	60, 63	<code>\glxtr@addloclistfield</code>	14
<code>\glstextup</code>	231	<code>\glxtr@addunused</code>	48
<code>\glstreechilddesc</code>	376, 378	<code>\glxtr@applyabbrvfmt</code>	207, 208
<code>\glstreechildpredesc</code>	378	<code>\glxtr@applyabbrvstyle</code>	190, 192, 211
<code>\glstreechildprelocation</code> ...	376, 378, 380	<code>\glxtr@counterrecord</code>	134
<code>\glstreechildsymbol</code>	376, 378	<code>\glxtr@do@alsoindex@wrglossary</code>	14
<code>\glstreedefaultnamefmt</code>	375, 376	<code>\glxtr@doooption</code>	5, 16, 17, 23, 24, 26
<code>\glstreedesc</code>	376–378	<code>\glxtr@fields</code>	132
		<code>\glxtr@headentry@p</code>	91, 92
		<code>\glxtr@hyperoutsidefalse</code>	62
		<code>\glxtr@hyperoutsidettrue</code>	61, 62
		<code>\glxtr@ifnextpunc</code>	189
		<code>\glxtr@ifpunctoken</code>	189
		<code>\glxtr@inc@linkcount</code>	63, 150
		<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 23, 27
		<code>\glxtr@indexonly@saveentrycounter</code>	13, 14, 27
		<code>\glxtr@keylist</code>	51–53
		<code>\glxtr@label</code>	404
		<code>\glxtr@langtag</code>	132
		<code>\glxtr@linkprefix</code>	132, 133
		<code>\glxtr@loaddialect</code>	327, 361
		<code>\glxtr@makeglossaries</code>	113
		<code>\glxtr@newabbreviation</code>	111, 192

<code>\glxtr@next</code>	189	<code>\glxtrbibaddress</code>	329
<code>\glxtr@org@do@wrglossary</code>	27	<code>\glxtrbibauthor</code>	329
<code>\glxtr@org@dohyperlink</code>	83	<code>\glxtrbibbooktitle</code>	329
<code>\glxtr@org@getgrouptitle</code>	122	<code>\glxtrbibchapter</code>	329
<code>\glxtr@org@newignoredglossary</code>	38	<code>\glxtrbibedition</code>	329
<code>\glxtr@orgmakenoidxglossaries</code>	49	<code>\glxtrbibhowpublished</code>	329
<code>\glxtr@pluralsuffixes</code>	132	<code>\glxtrbibinstitution</code>	329
<code>\glxtr@process</code>	137, 138	<code>\glxtrbibjournal</code>	329
<code>\glxtr@provideignoredglossary</code>	39	<code>\glxtrbibmonth</code>	329
<code>\glxtr@punctlist</code>	189	<code>\glxtrbibnote</code>	329
<code>\glxtr@record</code>	11, 132	<code>\glxtrbibnumber</code>	329
<code>\glxtr@record@nr</code>	13	<code>\glxtrbiborganization</code>	329
<code>\glxtr@recordsee</code>	12	<code>\glxtrbibpages</code>	329
<code>\glxtr@resource</code>	131, 132	<code>\glxtrbibpublisher</code>	329
<code>\glxtr@s@newignoredglossary</code>	38	<code>\glxtrbibschool</code>	329
<code>\glxtr@s@provideignoredglossary</code> ..	39	<code>\glxtrbibseries</code>	329
<code>\glxtr@saveentrycounter</code>	8, 9, 12, 81	<code>\glxtrbibtitle</code>	329
<code>\glxtr@setaccessdisplay</code>	176	<code>\glxtrbibtype</code>	329
<code>\glxtr@setbookindexmark</code>	404	<code>\glxtrbibvolume</code>	329
<code>\glxtr@setup@record</code>	13, 14, 26, 27	<code>\glxtrbookindexatendgroup</code>	402
<code>\glxtr@shortcutsval</code>	132, 133	<code>\glxtrbookindexatsubendgroup</code>	402
<code>\glxtr@texencoding</code>	132, 133	<code>\glxtrbookindexatsubsubendgroup</code> ..	402
<code>\glxtr@undefaction@nr</code>	7	<code>\glxtrbookindexbetween</code>	402
<code>\glxtr@undefaction@val</code>	7	<code>\glxtrbookindexbookmark</code>	403
<code>\glxtr@usesee</code>	43	<code>\glxtrbookindexcols</code>	401
<code>\glxtr@warnonexistsordo</code> ..	7, 13, 14, 41, 42	<code>\glxtrbookindexcolspread</code>	401
<code>\glxtr@writefields</code>	131	<code>\glxtrbookindexfirstmarkfmt</code>	404
<code>\glxtrabbreviationfont</code>	58	<code>\glxtrbookindexformatheader</code>	403
<code>\glxtrabbrvfootnote</code>		<code>\glxtrbookindexgroupskip</code>	403
.....	219–221, 241–243, 255–257, 277–279	<code>\glxtrbookindexlastmarkfmt</code>	404
<code>\glxtrabbrvpluralsuffix</code>	132,	<code>\glxtrbookindexmulticolenv</code>	401
	197, 215, 217, 219, 221, 223, 225, 228,	<code>\glxtrbookindexname</code>	399, 402
	231, 245, 259, 281, 291, 293, 297, 302, 304	<code>\glxtrbookindexparentchildsep</code> ..	399, 401
<code>\glxtrabbrvtype</code>	17, 194	<code>\glxtrbookindexparentschildsep</code> ..	
<code>\glxtractivateno</code>	117	401, 402
<code>\glxtraddallcrossrefs</code>	47	<code>\glxtrbookindexprelocation</code> ...	399, 402
<code>\glxtralias</code>	81	<code>\glxtrbookindexsubbetween</code>	402
<code>\glxtrAltTreeIndent</code>	381, 382	<code>\glxtrbookindexsubname</code>	403
<code>\glxtralttreeInit</code>	391, 397, 398	<code>\glxtrbookindexsubprelocation</code>	403
<code>\glxtrAltTreePar</code>	381	<code>\glxtrbookindexsubsubbetween</code>	402
<code>\glxtrAltTreeSetHangIndent</code> ...	381, 391	<code>\glxtrbookindexthepage</code>	404
<code>\glxtrAltTreeSetSubHangIndent</code> ...	392	<code>\glxtrcat</code>	51–53
<code>\glxtralttreeSubSymbolDescLocation</code> ..	392	<code>\glxtrchecknohyperfirst</code>	67, 68
<code>\glxtralttreeSymbolDescLocation</code> ..		<code>\glxtrcombingdiacriticIIrules</code> ..	333
.....	381, 392	<code>\glxtrcombingdiacriticIrules</code> ...	333
<code>\glxtrassignfieldfont</code>	66–73	<code>\glxtrcombingdiacriticIVrules</code> ..	333
<code>\glxtrautoindex</code>	179	<code>\glxtrComputeTreeIndent</code>	391, 392
<code>\glxtrautoindexassignsort</code>	179	<code>\glxtrComputeTreeSubIndent</code>	392
<code>\glxtrautoindexentry</code>	179		

\glxtrcounterprefix	124	223, 225, 229, 231, 233, 235, 237, 239,
\glxtrcurrencyrules	336	241, 242, 244, 246, 248, 250, 251, 253,
\Glsxtrdefaultsubsequentfmt ...	210, 212	255, 256, 258, 260, 262, 263, 265, 267,
\glxtrdefaultsubsequentfmt ...	210, 212	268, 271, 273, 274, 276, 277, 279, 282,
\Glsxtrdefaultsubsequentplfmt .	210, 212	283, 286, 289, 292, 294, 297, 300, 302, 305
\glxtrdefaultsubsequentplfmt .	210, 212	\Glsxtrfullpl
\GlsXtrDefineAbbreviationShortcuts .	20	18, 19, 318, 319
\GlsXtrDefineAcShortcuts	20, 21	\Glsxtrfullpl
\GlsXtrDefineOtherShortcuts	20	18, 19, 318, 319
\glxtrdetoklocation	143	\Glsxtrfullplformat
\glxtrdiscardperiod	186 196, 209, 212, 215, 217, 220,
\glxtrdisplayendloc	123	222, 224, 226, 229, 232, 234, 236, 237,
\glxtrdisplayendlohook	124	239, 241, 242, 244, 246, 248, 250, 251,
\glxtrdisplaysingleloc	124	254–256, 258, 260, 262, 264, 265, 267,
\glxtrdisplaystartloc	123	269, 271, 273, 275, 276, 278, 280, 282,
\glxtrdoautoindexname	82, 175	284, 286, 289, 292, 295, 297, 300, 303, 305
\glxtrdopostpunc	221, 243, 257, 279	\glxtrfullplformat
\glxtrdowrglossaryhook	82 209, 212, 215, 217, 219, 221,
\GlsXtrDualField	328	224, 225, 229, 232, 233, 235, 237, 239,
\glxtrdmsuffix	260, 261, 263,	241, 242, 244, 246, 248, 250, 251, 253,
265, 266, 268, 270, 272, 273, 275, 277, 279		255, 256, 258, 260, 262, 263, 265, 267,
\GlsXtrEnableEntryCounting	109	268, 271, 273, 274, 276, 278, 279, 282,
\GlsXtrEnableEntryUnitCounting	97	283, 286, 289, 292, 294, 297, 300, 302, 305
\GlsXtrEnableOnTheFly	51, 53	\glxtrfullsep
\glxtrendfor	32	. 195, 214–218, 220, 222, 223, 225–229,
\glxtrfieldlistgadd	133	231–255, 257–272, 274, 276, 278, 280,
\glxtrfieldtitlecase	170–173, 177	281, 291, 292, 294, 296, 299–301, 305, 306
\glxtrfieldtitlecasecs	170	\glxtrrgenabbrvfmt
\glxtrfieldxifinlist	138	58
\glxtrfirstscfont	231	\glxtrrgeneralpuncIIrules
\glxtrfirstsmfont	245	336
\GlsXtrFmtDefaultOptions	30	\glxtrrgeneralpuncIrules
\glxtrfmtdisplay	30	336
\GlsXtrFmtField	30	\glxtrrgetgroupitle
\glxtrfootnotename	219, 221, 241, 243, 255, 257, 277, 279	122, 403
\GlsXtrFormatLocationList	55, 58, 388–390	\glxtrrheadfirst
\Glsxtrfull	18, 19, 318, 319	309
\Glsxtrfull	18, 19, 319	\glxtrrheadfirst
\glxtrfull	18, 19, 318	309
\Glsxtrfullformat	195, 209, 212, 215, 217, 220, 222,	\Glsxtrrheadfirstplural
224, 226, 229, 232, 233, 235, 237, 239,		309
241, 242, 244, 246, 248, 250, 251, 253,		\Glsxtrrheadfull
255, 256, 258, 260, 262, 264, 265, 267,		309
269, 271, 273, 274, 276, 278, 280, 282,		\Glsxtrrheadfullpl
283, 286, 289, 292, 294, 297, 300, 302, 305		310
\glxtrfullformat	195, 209, 210, 212, 215, 217, 219, 221,	\glxtrrheadfullpl
		309
		\Glsxtrrheadlong
		309
		\glxtrrheadlong
		309
		\Glsxtrrheadlongpl
		309
		\glxtrrheadlongpl
		309
		\Glsxtrrheadname
		309
		\glxtrrheadname
		134, 309
		\Glsxtrrheadplural
		309
		\glxtrrheadplural
		309
		\Glsxtrrheadshort
		309
		\glxtrrheadshort
		309

\Glsxtrheadshortpl	309	235–237, 239, 240, 242, 244, 249, 251–
\glxtrheadshortpl	309	254, 257, 258, 267–270, 272, 274, 276,
\Glsxtrheadtext	309	278, 280, 284, 286, 294, 298, 303, 305, 325
\glxtrheadtext	309	\glxtrininsertinsidefalse
\glxtrhyperlink	23, 24, 84	214
\glxtrhyphensuffix	292, 300	\GlsXtrInternalLocationHyperlink 23, 125
\glxtrifcounttrigger	100, 101	\glxtrLatinA
\glxtrifcustomdiscardperiod	186	337–342
\glxtrifemptyglossary	123, 129, 136	\glxtrLatinAELigature
\GlsXtrIfFieldCmpNum	32	339, 341, 342
\GlsXtrIfFieldEqNum	135	\glxtrLatinE
\GlsXtrIfFieldNonZero	327	337–342
\glxtrifhasfield	35, 80, 328, 399	\glxtrLatinEszettSs
\glxtrifhyphenstart		338, 340, 343
.....	290, 293, 295, 296, 299, 301	\glxtrLatinEszettSz
\glxtrifindexing	81	339, 341
\glxtrifinmark	64, 91–94, 308–310	\glxtrLatinEth
\glxtrifnextpunc	189, 190	338–342
\glxtrifperiod	186, 188	\glxtrLatinH
\glxtrifrecordtrigger	145–147	338–342
\glxtrifwasfirstuse		\glxtrLatinI
.....	65, 67, 68, 73–77, 79, 112, 187,	338–342
	188, 197, 200–207, 221, 243, 244, 257,	\glxtrLatinK
	258, 279, 283, 285, 287, 296, 298, 302, 303	338–342
\glxtrinclinkcounter	149	\glxtrLatinL
\glxtrindexaliased	80, 81	338–342
\glxtrindexseealso	46, 47	\glxtrLatinM
\glxtrinithyperoutside	62	338–342
\glxtrinitwrgloss	62, 145	\glxtrLatinN
\glxtrinitwrglossbeforefalse	61	338–342
\glxtrinitwrglossbeforetrue	61	\glxtrLatinOELigature
\Glsxtrinlinefullformat 196, 198, 212,		340, 342, 343
220, 222, 223, 225, 227, 228, 235, 236,		\glxtrLatinP
238–240, 243, 244, 249, 251–253, 255,		338–343
257, 258, 267–269, 271, 272, 274, 276,		\glxtrLatinS
278, 280, 284, 286, 294, 298, 303, 305, 325		338–343
\glxtrinlinefullformat 196–198,		\glxtrLatinT
212, 220, 222, 223, 225, 226, 228, 235–		338–343
237, 239, 240, 242, 244, 249, 250, 252–		\glxtrLatinThorn
254, 256, 258, 267–270, 272, 274, 276,		342
278, 280, 284, 286, 294, 297, 303, 305, 325		\glxtrLatinX
\Glsxtrinlinefullplformat 196, 199, 212,		338–343
220, 222, 223, 225, 227, 228, 235, 237–		\glxtrlocationhyperlink
239, 241, 243, 244, 249, 251–253, 255,		125
257, 259, 267–269, 271, 272, 274, 276,		\glxtrlocrangefmt
278, 280, 284, 286, 294, 298, 303, 306, 326		124
\glxtrinlinefullplformat .. 196, 199,		\GLSxtrlong
200, 212, 220, 222, 223, 225, 226, 228,		18, 19, 316, 317
		\Glsxtrlong
		18, 19, 317
		\glxtrlong
		18, 19, 316
		\glxtrlonghyphen
		297
		\glxtrlonghyphennoshort
		294, 295
		\glxtrlonghyphenshort
		292
		\glxtrlongnoshortdescname . 227, 275, 293
		\glxtrlongnoshortname
	
		230, 238, 252, 270, 271, 295
		\GLSxtrlongpl
		18, 19, 317
		\Glsxtrlongpl
		18, 19, 318
		\glxtrlongpl
		18, 19, 317
		\glxtrlongshortdescname
	
		216, 232, 246, 260, 262, 292, 298
		\glxtrlongshortdescsort
	
		216, 232, 246, 260, 262, 287, 292, 298
		\glxtrlongshortname
	
		214, 231, 245, 259, 261, 281, 283, 291, 296
		\glxtrlongshortuserdescname .. 284, 287
		\glxtrmarkhook
		307, 308
		\glxtrMathItalicAlpha
		347, 351, 352
		\glxtrMathItalicBeta
		347, 351, 352

<code>\glxtrMathItalicChi</code> ...	347, 348, 352, 353	251–255, 257–272, 274, 276, 278, 280,
<code>\glxtrMathItalicDelta</code>	347, 351, 352	281, 291, 292, 294, 296, 299–301, 305, 306
<code>\glxtrMathItalicEpsilon</code>	347, 348, 351, 352	<code>\Glsxtrpl</code> 53
<code>\glxtrMathItalicEta</code> ...	347, 348, 351, 352	<code>\glxtrpl</code> 53
<code>\glxtrMathItalicGamma</code>	347, 351, 352	<code>\glxtrpostdescription</code>
<code>\glxtrMathItalicIota</code> ..	347, 348, 351, 352 117, 118, 168, 185, 375
<code>\glxtrMathItalicKappa</code> .	347, 348, 351, 352	<code>\glxtrposthyphenlong</code> 302, 303
<code>\glxtrMathItalicLambda</code>	347, 348, 351, 352	<code>\glxtrposthyphenshort</code> 296, 298
<code>\glxtrMathItalicMu</code>	347, 348, 351, 352	<code>\glxtrposthyphensequent</code>
<code>\glxtrMathItalicNu</code>	347, 348, 351, 352 297, 298, 302, 304
<code>\glxtrMathItalicOmega</code> .	347, 348, 352, 353	<code>\glxtrpostlink</code> 186
<code>\glxtrMathItalicOmicron</code>	347, 348, 351, 352	<code>\glxtrpostlinkendsentence</code> 186
<code>\glxtrMathItalicPhi</code> ...	347, 348, 351, 353	<code>\glxtrpostlinkhook</code> 186
<code>\glxtrMathItalicPi</code>	347, 348, 351, 352	<code>\glxtrpostlocalreset</code> 96, 99, 107
<code>\glxtrMathItalicPsi</code> ...	347, 348, 352, 353	<code>\glxtrpostlocalunset</code> 96, 98, 106
<code>\glxtrMathItalicRho</code> ...	347, 348, 351, 352	<code>\glxtrpostlongdescription</code> 37
<code>\glxtrMathItalicSigma</code> .	347, 348, 351, 352	<code>\glxtrpostnamehook</code> 173–175, 177
<code>\glxtrMathItalicTau</code> ...	347, 348, 351, 352	<code>\GlsXtrPostNewAbbreviation</code>
<code>\glxtrMathItalicTheta</code> .	347, 348, 351, 352	. 194, 212, 214, 216–219, 221, 223–236,
<code>\glxtrMathItalicUpsilon</code>	347, 348, 351, 353	238, 241, 243, 245, 247–250, 252, 256,
<code>\glxtrMathItalicXi</code>	347, 348, 351, 352	257, 259, 261–266, 268, 270, 271, 273,
<code>\glxtrMathItalicZeta</code> ..	347, 348, 351, 352	275, 277, 279, 282–285, 287–289, 291–
<code>\glxtrnewabbrevpresetkeyhook</code>	193	293, 295, 296, 298–300, 302, 303, 305, 306
<code>\glxtrnewnumber</code>	19	<code>\glxtrpostreset</code> 96, 98, 99, 106, 107
<code>\glxtrnewsymbol</code>	19	<code>\glxtrpostunset</code> 95, 98, 106
<code>\glxtrNoGlossaryWarning</code>	14, 21, 126	<code>\glxtrprelocation</code>
<code>\GlsXtrNoGlsWarningAutoMake</code>	130 365, 367, 369, 371, 373, 376, 399
<code>\GlsXtrNoGlsWarningBuildInfo</code>	130	<code>\glxtrprotectlinks</code> 84, 85
<code>\GlsXtrNoGlsWarningCheckFile</code>	129	<code>\GlsXtrRecordCounter</code> 11
<code>\GlsXtrNoGlsWarningEmptyMain</code> ..	129, 130	<code>\glxtrrecordtriggervalue</code> 144
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	129	<code>\GlsXtrRecordWarning</code> 130
<code>\GlsXtrNoGlsWarningEmptyStart</code>	129	<code>\glxtrregularfont</code> 58, 66
<code>\GlsXtrNoGlsWarningHead</code>	129	<code>\glxtrresourcecount</code> 131
<code>\GlsXtrNoGlsWarningMisMatch</code>	130	<code>\glxtrresourcefile</code> 131
<code>\GlsXtrNoGlsWarningNoOut</code>	130	<code>\glxtrresourceinit</code> 131
<code>\GlsXtrNoGlsWarningTail</code>	130	<code>\glxtrrestoremarkhook</code> 307, 308
<code>\glxtrnopostpunc</code>	117	<code>\glxtrrestorepostpunc</code> 117, 118
<code>\glxtronlydescname</code>	306	<code>\glxtrscfont</code> 231
<code>\glxtronlydescsort</code>	306	<code>\glxtrscsuffix</code>
<code>\glxtronlyname</code>	304 231, 233, 235, 236, 238, 240, 242, 244
<code>\glxtronlysuffix</code>	305	<code>\GlsXtrSetActualChar</code> 181
<code>\glxtrorg@ifKV@glslink@hyper</code>	59	<code>\glxtrsetaliasnoindex</code> 13, 14, 81
<code>\glxtrorglong</code>	192, 215, 293	<code>\GlsXtrSetEncapChar</code> 182
<code>\glxtrorgshort</code>	192, 215	<code>\GlsXtrSetEscChar</code> 181
<code>\GLSxtrp</code>	91	<code>\glxtrsetfieldifexists</code> 34
<code>\Glsxtrp</code>	90	<code>\GlsXtrSetLevelChar</code> 181
<code>\glxtrp</code>	90, 92	<code>\glxtrsetpopts</code> 89
<code>\glxtrparen</code>	187, 195, 214–	<code>\glxtrsetupfulldefs</code>
	218, 220, 222, 223, 225–229, 231–249, 197–200, 221, 244, 258, 279

<code>\GLSxtrshort</code>	18, 19, 93, 94, 310, 311	<code>\glxtrrtitlename</code>	309, 310, 321
<code>\Glsxtrshort</code>	18, 19, 311, 312	<code>\glxtrrttitleorpdforheading</code>	
<code>\glxtrshort</code>	18, 310, 311	25, 134, 135, 309, 310
<code>\glxtrshortdescname</code> ...	225, 236, 250, 268	<code>\Glsxtrrttitleplural</code>	309, 310, 322
<code>\glxtrshorthyphen</code>	302, 303	<code>\glxtrrttitleplural</code>	309, 310, 322
<code>\glxtrshorthyphenlong</code>	300	<code>\Glsxtrrttitleshort</code>	309, 310, 320
<code>\glxtrshortlongdescname</code>		<code>\glxtrrttitleshort</code>	309, 310, 320
.....	218, 234, 248, 264, 266, 300, 303	<code>\Glsxtrrttitleshortpl</code>	309, 310, 321
<code>\glxtrshortlongdescsort</code>		<code>\glxtrrttitleshortpl</code>	309, 310, 320
....	218, 234, 248, 264, 266, 289, 300, 303	<code>\Glsxtrrttitletext</code>	309, 310, 322
<code>\glxtrshortlongname</code>		<code>\glxtrrttitletext</code>	309, 310, 321
.....	216, 233, 247, 263, 264, 285, 288, 299, 301	<code>\GlsXtrTotalRecordCount</code>	144
<code>\glxtrshortlonguserdescname</code> ..	287, 289	<code>\glxtrrtreetopindent</code>	382, 390
<code>\glxtrshortnolongname</code> .	223, 234, 249, 266	<code>\glxtrundefaction</code> ..	7, 13, 14, 28, 38, 40–42
<code>\GLSxtrshortpl</code>	18, 19, 311, 312	<code>\glxtrundeftag</code>	27, 120, 121
<code>\Glsxtrshortpl</code>	18, 19, 312	<code>\glxtrtrunsrtdo</code>	138
<code>\glxtrshortpl</code>	18, 19, 311	<code>\glxstrUpAlpha</code>	346, 351, 352
<code>\glxtrsmfont</code>	245	<code>\glxstrUpBeta</code>	346, 351, 352
<code>\glxtrsmsuffix</code>		<code>\glxstrUpChi</code>	346, 347, 352, 353
....	246, 247, 249, 250, 252, 254, 256, 258	<code>\glxstrUpDelta</code>	346, 351, 352
<code>\GlsXtrStandaloneGlossaryType</code> .	134, 135	<code>\glxstrUpDigamma</code>	346, 347, 351
<code>\GlsXtrStandaloneSubEntryItem</code> .	134, 136	<code>\glxstrUpEpsilon</code>	346, 351, 352
<code>\Glsxtrsubsequentfmt</code>		<code>\glxstrUpEta</code>	346, 351, 352
.....	209, 212, 228, 238, 240,	<code>\glxstrUpGamma</code>	346, 351, 352
.....	253, 254, 270, 272, 274, 275, 294, 297, 302	<code>\glxstrUpIota</code>	346, 351, 352
<code>\glxtrsubsequentfmt</code>		<code>\glxstrUpKappa</code>	346, 351, 352
.....	208, 209, 212, 228, 238, 240,	<code>\glxstrUpLambda</code>	346, 351, 352
.....	253, 254, 270, 272, 274, 275, 294, 297, 302	<code>\glxstrUpMu</code>	346, 351, 352
<code>\Glsxtrsubsequentplfmt</code>		<code>\glxstrUpNu</code>	346, 351, 352
.....	208, 212, 228, 239, 240,	<code>\glxstrUpOmega</code>	346, 347, 352, 353
.....	253, 254, 270, 272, 274, 275, 294, 297, 302	<code>\glxstrUpOmicron</code>	346, 347, 351, 352
<code>\glxtrsubsequentplfmt</code>		<code>\glxstrUpPhi</code>	346, 347, 352, 353
.....	208, 212, 228, 238, 240,	<code>\glxstrUpPi</code>	346, 347, 351, 352
.....	253, 254, 270, 272, 274, 275, 294, 297, 302	<code>\glxstrUpPsi</code>	346, 347, 352, 353
<code>\glxtrsupplocationurl</code>	125	<code>\glxstrUpRho</code>	346, 347, 351, 352
<code>\glxtrtagfont</code>	184	<code>\glxstrUpSigma</code>	346, 347, 351, 352
<code>\Glsxtrrttitlefirst</code>	309, 310, 323	<code>\glxstrUpTau</code>	346, 347, 351, 352
<code>\glxtrrttitlefirst</code>	309, 310, 323	<code>\glxstrUpTheta</code>	346, 351, 352
<code>\Glsxtrrttitlefirstplural</code>	309, 310, 323	<code>\glxstrUpUpsilon</code>	346, 347, 351, 353
<code>\glxtrrttitlefirstplural</code>	309, 310, 323	<code>\glxstrUpXi</code>	346, 347, 351, 352
<code>\Glsxtrrttitlefull</code>	309, 310, 325	<code>\glxstrUpZeta</code>	346, 351, 352
<code>\glxtrrttitlefull</code>	309, 310, 325	<code>\GlsXtrUseAbbrStyleFmts</code>	
<code>\Glsxtrrttitlefullpl</code>	309, 310, 326	216, 218, 224, 226, 227,
<code>\glxtrrttitlefullpl</code>	309, 310, 325	229, 230, 232, 234, 237, 247, 249, 251,
<code>\Glsxtrrttitlelong</code>	309, 310, 324	261, 263, 264, 266, 269, 273, 277, 285,
<code>\glxtrrttitlelong</code>	309, 310, 324	287, 288, 290, 293, 295, 299, 301, 304, 306
<code>\Glsxtrrttitlelongpl</code>	309, 310, 325	<code>\GlsXtrUseAbbrStyleSetup</code> ..	224, 226, 227,
<code>\glxtrrttitlelongpl</code>	309, 310, 324	229, 230, 237, 240, 251, 254, 269, 273, 276
<code>\Glsxtrrttitlename</code>	309, 310, 321	<code>\glxstruserfield</code>	281

<code>\glxtruserparen</code>	282–289	<code>\IfFileExists</code>	22, 126, 129, 131, 133, 361, 363
<code>\glxtrusersuffix</code>	282, 283, 286, 289	<code>\ifglossaryexists</code>	42
<code>\glxtruseseealsoformat</code>	43, 44	<code>\ifglssacronym</code>	17, 129
<code>\glxtruseseeeformat</code>	43	<code>\ifglssacrshortcuts</code>	20
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	190, 213	<code>\ifglssautomake</code>	116, 129, 133
<code>\GlsXtrWarning</code>	51–53	<code>\ifglssentrycounter</code>	35
<code>\glxtrword</code>	192	<code>\ifglssentryexists</code>	9, 41, 51–53, 55, 65, 140, 165, 166, 185, 186
<code>\glxtrwordsep</code>	192, 290, 293, 295, 296, 299, 301	<code>\ifglssfieldeq</code>	164
<code>\glxtrwrglossmark</code>	24	<code>\ifglsshasfield</code>	30, 281
H			
<code>\hangindent</code>	378, 380, 381, 391–393, 397, 398	<code>\ifglsshaslong</code>	103, 148
<code>\hbox</code>	364	<code>\ifglsshasparent</code>	134, 136, 137, 140, 384–386
<code>\hfill</code>	364	<code>\ifglsshasshort</code>	43, 58, 66
<code>\href</code>	84	<code>\ifglsshasymbol</code>	187, 378, 379, 381
<code>\hsize</code>	54, 55	<code>\ifglssindexonlyfirst</code>	81
<code>\hss</code>	364	<code>\ifglssnogroupskip</code>	365, 367–375, 377, 378, 380, 392, 400
<code>\hyperlink</code>	84, 85	<code>\ifglssnonumberlist</code>	57
<code>\hyperpage</code>	178	<code>\ifglssnopostdot</code>	16, 117
<code>\hyperref</code>	84, 125, 328	<code>\ifglssanitizesort</code>	115
<code>hyperref</code> package	85, 178, 307, 319	<code>\ifglssubentrycounter</code>	35, 36
I			
<code>\if</code>	51	<code>\ifglssused</code>	47, 48, 79, 81, 82, 99, 108, 112, 208, 384, 385, 387, 388, 390
<code>\if@glxtr@autoseeindex</code>	25, 26, 42, 45	<code>\ifglssxindy</code>	126, 128
<code>\if@glxtr@format@override</code>	178	<code>\ifglxtr@hyperoutside</code>	64
<code>\if@glxtr@docdefrestricted</code>	49	<code>\ifglxtr@trinitwrglossbefore</code>	61, 64, 145
<code>\if@glxtr@indexcrossrefs</code>	15, 47	<code>\ifglxtr@insertinside</code>	200–207, 210, 215, 217, 219–229, 231–246, 248– 260, 262–265, 267–280, 282–284, 286, 289, 291, 293, 296–299, 301, 303, 305, 306
<code>\ifblank</code>	28, 51–53, 112	<code>\ifHy@hyperindex</code>	178
<code>\ifcase</code>	7, 13, 20, 21, 24, 50, 61, 118, 376, 402	<code>\ifinlist</code>	95
<code>\ifcsdef</code>	27, 36, 38–40, 61, 63, 77, 78, 90–94, 104, 116, 122, 123, 135, 139, 141, 143, 149, 170–177, 187, 190, 207, 211, 367–374	<code>\ifinlistcs</code>	31, 49
<code>\ifcsstring</code>	28, 166, 211	<code>\ifinner</code>	25
<code>\ifcsundef</code>	33, 36, 38–40, 49, 54, 56, 85, 98, 104–107, 121, 122, 126, 139, 149, 150, 166, 210, 212, 213, 364, 382, 383, 391, 404	<code>\ifKV@glslink@hyper</code>	59, 62, 64
<code>\ifcsvoid</code>	47, 165	<code>\ifKV@glslink@local</code>	60, 145
<code>\ifdef</code>	14, 19, 26, 30, 35, 41, 42, 44, 45, 48, 54, 55, 80, 83, 84, 91–93, 116, 119–121, 132, 141, 168, 169, 181, 182, 185, 280, 308, 319–326, 328, 361, 364–367, 375–381, 392–398, 400, 403, 404	<code>\ifKV@glslink@noindex</code>	8, 9, 12, 30, 81
<code>\ifdefempty</code>	7–9, 32, 33, 38–40, 43, 63, 65, 98, 109, 110, 113, 116, 124, 137, 140, 144, 145, 159, 160, 183, 192, 208, 401	<code>\ifmmode</code>	25
<code>\ifdefequal</code>	35, 49, 130, 140, 176	<code>\ifnum</code>	15, 33, 48, 100, 107, 108, 121, 131, 144, 378, 380, 391, 392
<code>\ifdefstring</code>	6, 23, 35, 178, 184, 400	<code>\ifstreempty</code>	135, 142, 150
<code>\ifdefvoid</code>	42, 45–48, 84, 103, 121, 125, 140, 141	<code>\ifstrequal</code>	16, 17, 21, 22
<code>\ifdim</code>	54, 55, 112, 382–390	<code>\ifthenelse</code>	129, 130, 186
		<code>\IfTrackedLanguageFileExists</code>	326
		<code>\ifundef</code>	23, 32, 33, 113, 183–185, 361
		<code>\ifx</code>	8–11, 44, 54, 55, 63, 112, 116, 119, 123–125, 133, 178–180, 182, 189, 191, 192, 194, 290, 398
		<code>\immediate</code>	99, 108, 126, 133

<code>\index</code>	179	<code>\makeatother</code>	180
<code>\indexspace</code> .	365, 377–381, 392–398, 400, 403	<code>\makebox</code>	364, 391, 392
<code>\input</code>	326	<code>\makefirstuc</code>	184
<code>\inputencodingname</code>	132	<code>makeglossaries</code>	119
<code>\InputIfFileExists</code>	48	<code>\makeglossaries</code>	112, 126, 128–130, 133
<code>\istfilename</code>	113	<code>\makeglossary</code>	113
<code>\item</code>	128, 364–367, 376, 377, 393, 394	<code>makeindex</code>	405
J			
<code>\jobname</code>	48, 126, 128–131, 133	<code>makeindex</code>	14, 112
K			
<code>\key@ifundefined</code> .	12, 13, 28, 29, 77, 137, 139	<code>\makenoidxglossaries</code>	128
<code>\KV@glslink@hyperfalse</code> ..	66, 79, 80, 85, 86	<code>\MakeTextUppercase</code>	309
<code>\KV@glslink@hypertrue</code>	85	<code>\MakeUppercase</code>	309, 310
<code>\KV@glslink@noindexfalse</code>	80	<code>\marginpar</code>	25
<code>\KV@glslink@noindextrue</code>	80, 86	<code>\markboth</code>	308
L			
<code>\L</code>	342, 346	<code>\markright</code>	308
<code>\l</code>	346	<code>\mathit</code>	330
<code>\label</code>	23	<code>\mathrm</code>	329–331
<code>\LaTeX</code>	127–129	<code>\maxdimen</code>	54, 55
<code>\leaders</code>	364	<code>\mbox</code>	366, 367, 391
<code>\leavevmode</code>	37, 62	<code>\medskip</code>	129, 130, 139
<code>\let</code>	5, 7–14, 16, 18, 19, 25–27, 30, 32, 35, 37, 49, 50, 53, 54, 56, 57, 59, 60, 62–83, 85, 86, 89, 95–99, 106– 122, 131, 133, 137, 138, 140, 144, 145, 150, 159, 170–180, 183–185, 189–193, 197–207, 210, 212, 221, 244, 258, 279, 307–310, 361, 376, 381, 383, 394, 401–404	<code>\MessageBreak</code>	50, 53, 100, 108, 112, 113, 115, 116, 130, 211
<code>\letabbreviationstyle</code> ..	220, 222, 224, 226, 229, 230, 236, 237, 250, 251, 267, 269	<code>mfirstuc package</code>	183, 184
<code>\letcs</code>	28, 32, 33, 43, 47, 48, 61, 63, 77, 119–122, 139, 140, 170–177, 404	<code>\mfirstucMakeUppercase</code>	66–77, 79, 87–89, 91, 93, 94, 102, 103, 110, 148, 151–157, 161–164, 172, 174, 177, 198, 200, 202, 203, 205, 207–210
<code>\levelchar</code>	181	<code>\mfu@checkword@arg</code>	184
<code>\listadd</code>	104	<code>\mfu@checkword@do</code>	184
<code>\listbreak</code>	184	N	
<code>\listcsadd</code>	31	<code>\NeedsTeXFormat</code>	5, 327, 363, 399
<code>\listcseadd</code>	31, 105	<code>\new@glossaryentry</code>	50, 115
<code>\listcsgadd</code>	31, 49	<code>\new@ifnextchar</code>	29, 78, 79, 102, 142, 145–147, 188, 197–207
<code>\listcsxadd</code>	31, 104	<code>\newabbr</code>	18, 19
<code>\listxadd</code>	95	<code>\newabbreviation</code>	18, 19
<code>\loadglsentries</code>	50, 127	<code>\newabbreviationhook</code>	194
<code>\long</code>	37	<code>\newabbreviationstyle</code>	214–216, 218–220, 222, 224, 226, 227, 229–234, 236–238, 240, 241, 243, 245–252, 254, 255, 257, 259–264, 266, 268–271, 273, 275–277, 279, 281, 283–285, 287–289, 291–293, 295, 296, 298–301, 303, 304, 306
M			
<code>\MakeAcronymsAbbreviations</code>	111	<code>\newacronym</code>	110, 111
<code>\makeatletter</code>	48, 126, 131, 180	<code>\newacronymhook</code>	110
		<code>\newacronymstyle</code>	111
		<code>\newcommand</code>	5–13, 15– 33, 35–44, 46–48, 50–54, 56–58, 60–62, 64–66, 77–86, 89, 91–98, 100, 102–106,

108, 109, 111, 112, 116–122, 124, 125, 127–159, 161–170, 175, 176, 178–192, 194–208, 210–219, 222, 224, 227, 229– 231, 245, 259, 280, 281, 284, 286, 290, 291, 293, 295, 296, 299, 301, 304, 306, 308, 310–329, 332–360, 363, 365, 375– 379, 381–383, 390, 391, 399, 400, 403, 404	\newwrite 113
\newcount 131, 141	\nobreak 366, 367, 377, 379–381, 393–396
\newcounter 23, 149	\NoCaseChange 91–94, 135, 310–319
\newentry 19	\noexpand 10, 11, 22, 44, 46–48, 110, 126, 131, 137, 138, 140, 149, 179, 180, 194, 363, 402
\newglossary 17, 113	\nofiles 129
\newglossaryentry 19, 50, 98, 106, 110, 168, 169, 194	\noindent ... 129, 130, 379–381, 394–396, 398
\newglossaryentry options	\nopagebreak 377, 379–381, 392–399, 403
access 159	\nostdesc 37, 52, 53, 117, 168
alias 16, 42, 45, 47	\ns@GLSxtrfull 198
desc 154, 155, 162	\ns@GLSxtrfull 197
descplural 155, 162, 163	\ns@GLSxtrfull 197
first 83, 152, 161, 214, 314–316, 322, 405	\ns@GLSxtrfullpl 199
firstaccess 160	\ns@GLSxtrfullpl 199
firstplural .. 153, 162, 214, 315, 316, 323, 405	\ns@GLSxtrfullpl 198
group 139, 140	\ns@GLSxtrlong 203
loclist 31	\ns@GLSxtrlong 202, 203
long 157, 163, 324	\ns@GLSxtrlong 202
longplural 157, 164, 324	\ns@GLSxtrlongpl 207
name ... 43, 150, 151, 161, 178, 312, 313, 321	\ns@GLSxtrlongpl 206
plural 151, 152, 161, 214, 314, 322	\ns@GLSxtrlongpl 206
see 15, 16, 26, 42, 45, 47, 50, 113	\ns@GLSxtrshort 201
seealso 16, 42, 43, 45, 47, 416	\ns@GLSxtrshort 201
short 156, 163, 191	\ns@GLSxtrshort 200
shortaccess 158–160	\ns@GLSxtrshortpl 205
shortaccessplural 159	\ns@GLSxtrshortpl 204
shortplural 157, 163, 191	\ns@GLSxtrshortpl 204
symbol 153, 154, 162	\null 21
symbolplural 154, 162	\number 48, 105–107, 131, 142
text 83, 151, 161, 214, 216, 313, 314, 321	\numexpr 105, 107
textaccess 160	
\newglossarystyle 401	
\newif 61, 178, 214	
\newlength 381, 382	
\newnum 19	
\newrobustcmd 29, 30, 32, 34, 35, 44, 45, 62, 65, 78, 79, 89–91, 102, 117, 122, 134, 135, 138, 139, 142, 143, 145–147, 176, 183, 184, 197–208, 290, 308, 311–319, 383–390	
\newsym 19	
\newterm 168	
\newtoks 191	

O

\O 342, 345
\o 345
\or 7, 13, 14, 20, 21, 24, 50, 61, 376, 402
\org@glossaryentrynumbers 55, 117
\org@glossarytitle 116
\org@ifKV@glslink@hyper 62, 64

P

\p@glS@hyp@opt 82
package options:
abbreviations 17, 18
accsupp 21, 150
acronym 17
automake 116, 127, 133
true 133
autoseeindex 26
false 25

<code>\TrackLangIfHasDefaultScript</code>	361	<code>\write</code>	44, 99, 108, 113, 126, 133
<code>\TrackLangRequireDialectPrefix</code>	361		
U		X	
<code>\u</code>	327	<code>\x</code>	125, 149
<code>\undef</code>	13, 14, 48, 183	<code>\xcapitalisewords</code>	170
<code>\underline</code>	184	<code>\xdef</code>	117, 138
<code>\unskip</code>	37, 48, 364	<code>\xifinlist</code>	104
<code>upgreek</code> package	330	<code>\xifinlistcs</code>	31
<code>\usepackage</code>	128–130	<code>xindy</code>	405
W		<code>xindy</code>	14, 112
<code>\warn@nomakeglossaries</code>	114	<code>xkeyval</code> package	5
<code>\warn@noprintglossary</code>	114, 117	<code>\XKV@checkchoice</code>	57
<code>wrglossary</code> (counter)	23	<code>\XKV@plfalse</code>	57
		<code>\XKV@resa</code>	57
		<code>\XKV@sttrue</code>	57