

glossaries-extra.sty v1.35: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-08-13

Abstract

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

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

Contents

1	Main Package Code (glossaries-extra.sty)	5
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	27
1.3	Modifications to Commands Provided by glossaries	39
1.3.1	Existence Checks	43
1.3.2	Document Definitions	51
1.3.3	Existing Glossary Style Modifications	57
1.3.4	Entry Formatting, Hyperlinks and Indexing	61
1.3.5	Entry Counting	98
1.3.6	Acronym Modifications	113
1.3.7	Indexing and Displaying Glossaries	116
1.4	Link Counting	152
1.5	Integration with glossaries-accsupp	154
1.6	Categories	168
1.7	Abbreviations	194
1.7.1	Abbreviation Styles Setup	214
1.7.2	Predefined Styles (Default Font)	217
1.7.3	Predefined Styles (Small Capitals)	234
1.7.4	Predefined Styles (Fake Small Capitals)	248
1.7.5	Predefined Styles (Emphasized)	262
1.7.6	Predefined Styles (User Parentheses Hook)	284
1.7.7	Predefined Styles (Hyphen)	293
1.7.8	Predefined Styles (No Short on First Use)	307
1.8	Using Entries in Headings	310
1.9	Multi-Lingual Support	329
1.10	glossaries-extra-bib2gls.sty	330
2	Style Adjustments (glossaries-extra-stylemods.sty)	366
2.1	Package Initialisation	366
2.2	List-Like Styles	367
2.3	Longtable Styles	370
2.4	Long Ragged Styles	372
2.5	Supertabular Styles	374
2.6	Super Ragged Styles	376
2.7	Inline Style	378
2.8	Tree Styles	378
2.9	Multicolumn Styles	396

3 bookindex style (glossary-bookindex.sty)	402
3.1 Package Initialisation and Options	402
Glossary	408
Change History	409
Index	428

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/08/13 v1.35 (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 }%

```

undefaction

```

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

```

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

\glstr@record Does nothing by default.

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

```

\glstr@recordsee Does nothing by default.

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

```

\glstr@defaultnumberformat

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

```

\GlsXtrSetDefaultNumberFormat

```

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

```

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

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

```

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

```

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

```

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

```

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

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

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

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

Save the label in case it's needed.

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

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

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

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

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

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings.

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

Increment associated counter.

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

Check if noindex option has been used.

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

Check if thevalue has been set.

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

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

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

Save the entry counter.

```
147         \glsxtr@saveentrycounter
```

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

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

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

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

No need to escape special characters.

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

glslink@prekeys

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

lslink@postkeys

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

lossadd@prekeys

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

ossadd@postkeys

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

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

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

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

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

```

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

```

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

```

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

```

`r@recordcounter`

```

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

```

`p@recordcounter`

```

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

```

`op@recordcounter`

```

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

```

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

```

212 \newcommand*{\@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 }

```

unsrtglossaryunit

```

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}

```

`etting@alsoindex`

```

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

```

`rd@setting@only`

```

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

```

`ord@setting@off`

```

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

```

`record` Now define the record package option.

```

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

```

Don't record.

```

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

```

Only record (don't index).

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

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

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

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

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

Warn about using \printglossary:

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

Load glossaries-extra-bib2gls:

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

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

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

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

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

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

Need to provide conditional commands that are backward compatible:

`if@glxtrdocdef`

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

`lsxtrdocdeftrue`

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

`sxtrdocdeffalse`

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

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

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

`ocdefrestricted`

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

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

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

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

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

Switch off since this can increase the build time.

```
339 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

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

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

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

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

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

postdot Shortcut for nopostdot=false

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

nopostdot Needs to redefine \@glxtr@defpostpunc

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

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

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

```

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

```

`glxtrabbrvtype` Glossary type for abbreviations.

```
398 \newcommand*{\glxtrabbrvtype}{\gldefaulttype}
```

`bbreviationsdef` Set by abbreviations option.

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

`bbreviationsdef`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

Disable this command after it's been used.

```

444   \let\GlsXtrDefineAbbreviationShortcuts\relax
445 }

```

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

```

446 \newcommand*{\GlsXtrDefineAcShortcuts}{%

```

```

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

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

```

Disable this command after it's been used.

```

472 \let\GlsXtrDefineAcShortcuts\relax
473 }

```

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

```

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

```

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

@setupshortcuts Command used to set the shortcuts option.

```

486 \newcommand*{\@glxtr@setupshortcuts}{\fi}%

tr@shortcutsval Store the value of the shortcuts option. (Needed by bib2gls.)
487 \newcommand*{\@glxtr@shortcutsval}{\ifglacrshortcutsacro\else none\fi}%

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

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

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

523       \GlsXtrDefineAcShortcuts
524       \GlsXtrDefineAbbreviationShortcuts

```

```

525     \GlsXtrDefineOtherShortcuts
526 }%

```

```

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

```

Leave none and false as last option.

```

532 \else % none, false
533   \renewcommand*{\@glxtr@setupshortcuts}{}%
534 \fi
535 }

```

lsxtr@doaccsupp

```

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

```

accsupp If accsupp, load glossaries-accsupp package.

```

537 \@glxtr@declareoption{accsupp}{%
538 \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

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

```

539 \newcommand{\glxtrNoGlossaryWarning}[1]{%
540 \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
541 \@glxtr@defaultnoglossarywarning{#1}%
542 }

```

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

```

543 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
544 [\@glxtr@nomissingglstextval\@glxtr@nomissingglstextnr]%
545 {true,false}[true]{%
546   \ifcase\@glxtr@nomissingglstextnr\relax % true
547     \renewcommand{\glxtrNoGlossaryWarning}[1]{\null}%
548   \else % false
549     \renewcommand{\glxtrNoGlossaryWarning}[1]{%
550       \@glxtr@defaultnoglossarywarning{#1}%
551     }%
552   \fi
553 }

```

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

lsxtr@redefstyles

```

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

```

stylemods

```

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

```

glxtr@do@style

```

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

```

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

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

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

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

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

`ocationHyperlink`

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

The first two arguments are always control sequences.

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

`cationhyperlink`

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

`indexcounter`

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

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

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

Only increment if the current counter is `wrglossary`.

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

```

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

```

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

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

```

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

```

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

```

`sxtrwrglossmark` Does nothing by default.

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

```

`debug` Provide extra debug options.

```

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

```

Pass all other options to glossaries.

```

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

```

Process options.

```
653 \ProcessOptionsX

```

Load glossaries if not already loaded.

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

Load the glossaries-accsupp package if required.

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

Redefine \glspostdescription if required.

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

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

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

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

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

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

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

oindex@glossary

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

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

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

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

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

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

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

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

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

Define abbreviations glossaries if required.

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

Setup shortcuts if required.

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

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

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

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

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

Now define the user command:

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

```

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

```

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

```

720 \let\glxtr@org@@do@wrglossary\@@do@wrglossary

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`sxtrdialecthook`

```

728 \newcommand*{\@glxtrdialecthook}{}

```

Set up record option if required.

```

729 \glxtr@setup@record

```

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

```

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

```

1.2 Extra Utilities

`unusedOrUndefined`

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\glxtrprovide@storagekey` Unstarred version.

```

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

```

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

```

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

```

Provide the no-link command if not already defined.

```

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

```

`\provide@storagekey` Starred version.

```

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

```

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

`\GlsXtrFmtField`

```

790 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

791 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```

`\@glxtrfmt` Unstarred form.

```

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

```

`\s@glstrfmt` Starred form.

```

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

```

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

```

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

```

`\@@glstrfmt` Actual inner working.

```

799 \newcommand*\@@glstrfmt}[4]{%

```

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

```

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

```

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

```

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

```

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

```

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

```

`\glstrentryfmt` No link or indexing.

```

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

```

```

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

```

@glstrententryfmt

```

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

```

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

```

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

```

trfieldlistgadd Similarly but uses \listcsgadd.

```

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

```

trfieldlisteadadd Similarly but uses \listcseadd.

```

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

```

trfieldlistxadd Similarly but uses \listcsxadd.

```

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

```

Now provide commands to iterate over these lists.

fielddolistloop

```

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

```

ieldforlistloop

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

List element tests:

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

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

rfieldxifinlist Expands item.

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

lsxtrforcsvfield

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

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

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

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

lsxtrifhasfield Unstarred version adds grouping.

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

lsxtrifhasfield Starred version omits grouping.

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

```

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

```

`rIfFieldNonZero` Designed for numeric fields.

```

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

```

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

Designed for numeric fields.

```

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

```

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

Designed for numeric fields.

```

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

```

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

Just uses `\ifcsundef`.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`etfieldifexists`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

xGlsXtrSetField

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

eGlsXtrSetField

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

XtrIfFieldEqStr

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

rIfFieldEqXpStr Like the above but first expands the string.

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

fXpFieldEqXpStr Like the above but also expands the field value.

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

lsXtrForeignText

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

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

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

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

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

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

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

No tracked dialect found for the root language.

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

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

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

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

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

Does a caption hook exist for this?

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

No mapping. Try root language label instead.

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

No mapping. Try root language label instead.

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

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

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

nDialectWarning

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

```

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

```

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

```

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

```

`lossarypreamble`

```

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

```

lossarypreamble

```

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

```

1.3 Modifications to Commands Provided by glossaries

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

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

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

```

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

```

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

ewglossaryentry

```

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

```

ewglossaryentry Starred version.

```

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

```

```

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

```

`\newglossaryentry` Unstarred version.

```

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

```

The following is different from the base glossaries.sty:

```

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

```

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

```

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

```

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

`\ignoredglossary` Redefine to check for star.

```

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

```

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

```

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

```

```

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

```

ignoredglossary Starred form.

```

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

```

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

```

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

```

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

```

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

```

`ignoredglossary` Unstarred version.

```

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

```

```

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

```

`ignoredglossary` Starred form.

```

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

```

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

```

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

```

1.3.1 Existence Checks

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

```

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

```

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

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

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

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

`\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.

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

`\glsdoifexistsordo` Similarly for `\doifglossarynoexistsordo`.

```
1310 \ifdef\doifglossarynoexistsordo
1311 {%
1312    \renewcommand{\doifglossarynoexistsordo}[3]{%
1313      \doifglossarynoexists{#1}{#2}{#3}%
1314    }%
1315 }
```

```

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

```

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

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

```

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

```

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

```

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

```

```

1354 }

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

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

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

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

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

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

```

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

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

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

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

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

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

Add the cross-reference class definition to the hook.

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

Append to class list.

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

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

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

```

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

```

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

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

```

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

```

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

```

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

```

Add to the key mappings.

```

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

```

Set the default value.

```

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

```

Assign the field values.

```

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

```

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

```

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

```

Provide user-level commands to access the values.

```

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

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

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

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

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

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

trseealsolabels
1469 \glsaddstoragekey*{seealso}{}{\glxtrseealsolabels}

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

ryentryposthook Append to the hook to check for the alias and seealso keys.
1470 \appto\@newglossaryentryposthook{%
1471 \ifcsvoid{glo@\@glo@label @alias}%
1472 {%
1473 \ifcsvoid{glo@\@glo@label @seealso}%
1474 }%

```

```

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

```

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

```

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

```

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

```

1492 \AtEndDocument{\if@glstrindexcrossrefs\glstraddallcrossrefs\fi}

```

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

```

1493 \newcommand*\glstraddallcrossrefs{%
1494   \forallglossaries{\@glo@type}%
1495   {%
1496     \forallsentries[\@glo@type]{\@glo@label}%
1497     {%
1498       \ifglused{\@glo@label}%
1499       {\expandafter\glstr@addunusedxrefs\expandafter{\@glo@label}}}%
1500     }%
1501   }%
1502 }

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

`\glsxtrunusedformat`

```

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

```

1.3.2 Document Definitions

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

```

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

```

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

```

1546 \@gls@enablesavenonumberlist

```

```

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

```

Write all currently defined entries.

```

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

```

m@glossaryentry

```

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

```

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

```

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

```

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

```

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

```

Check for docdefs=restricted:

```

1584   \if@glsxtrdocdefrestricted

```

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

```

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

```

Disable document definitions.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`rEnableOnTheFly`

```

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

```

`rEnableOnTheFly`

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

```

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

```

`sxtrstarflywarn`

```

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

```

`rEnableOnTheFly`

```

1650 \newcommand*{\@GlsXtrEnableOnTheFly}{%

```

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

These definitions are all assigned the category given by:

```

\glstrcat
1651 \newcommand*\glstrcat[1]{}

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

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

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

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

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

\@glstrpl

```

```

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

```

\Glsxtrpl

```

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

```

\@Glsxtrpl

```

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

```

\GlsXtrWarning

```

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

```

Disable commands after the glossary:

```

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

```

abledflycommand

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

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

```
1739 \let\GlsXtrEnableOnTheFly\relax
1740 }
1741 \@onlypreamble\GlsXtrEnableOnTheFly
```

1.3.3 Existing Glossary Style Modifications

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

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

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

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

etglossarystyle

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

Only set the current style if it exists.

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

In case we have an old version of glossaries:

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

```

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

```

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

```

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

```

Similarly for \glsdescwidth:

\glsdescwidth

```

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

```

and for \glspagelistwidth:

lspagelistwidth

```

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

```

aryentrynumbers Has the nonumberlist option been used?

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

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

```
1818 \newcommand*{\GlsXtrFormatLocationList}[1]{#1}
```

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

ePreLocationTag

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

```

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

```

glsxtrpreloctag

```

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

```

glsxtrpreloctag

```

1855 \newcommand*{\@glsxtrpreloctag}{%

```

@glsxtr@pagetag

```

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

```

glsxtr@pagetag

```

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

```

lsxtrpostloctag

```

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

```

lsxtrpostloctag

```

1865 \newcommand*{\@glsxtrpostloctag}{%

```

lsxtr@preloctag

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

glxtr@doloctag

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

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

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

1.3.4 Entry Formatting, Hyperlinks and Indexing

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

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

`sxtrregularfont` Font used for regular entries.

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

`bbreviationfont` Font used for abbreviation entries.

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

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

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

```
1899 \renewcommand{\@gls@field@link}[4] [] {%
```

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

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

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

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

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

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

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

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

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

```

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

```

\@Gls@ Save the original definition and redefine.

```

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

```

\@Glspl@ Save the original definition and redefine.

```

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

```

\@GLS@ Save the original definition and redefine.

```

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

```

\@GLSpl@ Save the original definition and redefine.

```

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

```

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

```

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

```

```

1957 \fi
1958 }%
1959 \glspostlinkhook
1960 }

```

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

```

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

```

Post-link hook commands need initialising.

```

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

```

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

```

1975 \newcommand*{\glsxtrinitwrgloss}{%
1976 \glsifattribute{glslabel}{wrgloss}{after}%
1977 {%
1978 \glsxtrinitwrglossbeforefalse
1979 }%
1980 {%
1981 \glsxtrinitwrglossbeforetrue
1982 }%
1983 }

```

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

```

1984 \newif\ifglsxtrinitwrglossbefore
1985 \glsxtrinitwrglossbeforetrue

```

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

```

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

```

```

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

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

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

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

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

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

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

slinkpresetkeys  User hook performed immediately before options are set. Does nothing by default.
2020 \newcommand*{\glslinkpresetkeys}{}

sXtrExpandedFmt  Helper command that (protected) fully expands second argument and then applies it to the
                  first, which must be a command that takes a single argument.
2021 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2022   \protected@edef\@glstr@tmp{#2}%
2023   \expandafter#1\expandafter{\@glstr@tmp}%
2024 }

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

```

```

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

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

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

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

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

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

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

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

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

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

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

    User hook after options are set:
2046   \glslinkpostsetkeys

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

```

Check if the textformat key has been used.

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

Check textformat attribute (new to v1.21).

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

Do write if it should occur before the link text:

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

Do the link text:

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

Do write if it should occur after the link text:

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

Restore original value of \glolinkprefix:

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

As the original definition:

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

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

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

lsaddpresetkeys

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

saddpostsetkeys

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

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

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

```

Implement any default settings (before options are set)

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

```

Implement any default settings (after options are set)

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

```

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

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

```

2130 }%
2131 }

```

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

```

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

```

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

```

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

```

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

`assignfieldfont`

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

`ecknohyperfirst`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

Ensure that \GLSfirst honours the nohyperfirst attribute.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

```

```

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

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

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

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

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

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

```

\@Glsuseri@ First letter uppercase version.

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

\@GLSuseri@ All uppercase version.

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

\@Glsuserii@ First letter uppercase version.

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

\@GLSuserii@ All uppercase version.

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

\@Glsuseriii@ First letter uppercase version.

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

\@GLSuseriii@ All uppercase version.

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

\@Glsuseriv@ First letter uppercase version.

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

```

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

```

\@GLSuseriv@ All uppercase version.

```

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

```

\@Glsuserv@ First letter uppercase version.

```

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

```

\@GLSuserv@ All uppercase version.

```

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

```

\@Glsuservi@ First letter uppercase version.

```

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

```

\@GLSuservi@ All uppercase version.

```

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

```

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

\@acrshort No case change.

```

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

```

```

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

```

\@Acrshort First letter uppercase.

```

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

```

\@ACRshort All uppercase.

```

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

```

\@acrshortpl No case change.

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

\@Acrshortpl First letter uppercase.

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

\@ACRshortpl All uppercase.

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

2503 }

\@acrlong No case change.

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

\@Acrlong First letter uppercase.

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

\@ACRlong All uppercase.

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

```

2546 }%
2547 \glspostlinkhook
2548 }

```

\@acrlongpl No case change.

```

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

```

\@Acrlongpl First letter uppercase.

```

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

```

\@ACRlongpl All uppercase.

```

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

```

```

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

```

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

\@glsaddkey

```

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

```

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

```

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

```

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

```

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

```

```

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

```

Finally the all caps version (modified):

```

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

```

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

```

2673 \providecommand*\@gls@link@nocheckfirsthyper{}

```

checkfirsthyper Modify check to determine if the hyperlink should be automatically suppressed, but save the

original in case the acronyms are restored.

```
2674 \let\@glstr@org@checkfirsthyper\@gls@link@checkfirsthyper
2675 \renewcommand*{\@gls@link@checkfirsthyper}{%
```

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

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

Store the category label for convenience.

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

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

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

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

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

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

lt@glslink@opts

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

```

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

```

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

```

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

```

setaliasnoindex

```

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

```

xtrindexaliased

```

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

```

xtrindexaliased

```

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

```

```

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

```

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

```

2744 \let\glxtrindexaliased\@no\glxtrindexaliased

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

Similarly for the “noidx” version:

s@noidxglossary

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

xtr@do@@wrindex

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

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

```
2780 \newcommand*{\glstrdowrglossaryhook}[1]{}
```

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

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

alt@gls@hyp@opt User version

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

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

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

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

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

rSetAltModifier

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

org@dohyperlink

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

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

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

```
2805 \ifdef\glsnavhyperlink
```

```
2806 {
```

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

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

Scope:

```
2809   {%
```

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

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

```
2812   }%
```

```
2813 }%
```

```
2814 }
```

```
2815 {}
```

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

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

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

```
2818   {%
```

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

```
2820     {%
```

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

```
2822     }%
```

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

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

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

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

```
2827     }%
```

```
2828     {%
```

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

```
2830         {}}%
```

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

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

```
2833     }%
```

```
2834   }%
```

```
2835   {%
```

```

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

```

Check for alias.

```

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

```

Redirect link to the alias target.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

2889 \newcommand*{\glstrprotectlinks}{%
2890   \KV@glslink@hyperfalse
2891   \KV@glslink@noindextrue
2892   \let\@gls@\@glstr@p@text@
2893   \let\@Gls@\@Glsxtr@p@text@
2894   \let\@GLS@\@GLSxtr@p@text@
2895   \let\@glspl@\@glstr@p@plural@
2896   \let\@Glspl@\@Glsxtr@p@plural@
2897   \let\@GLSpl@\@GLSxtr@p@plural@
2898   \let\@glstrshort@\@glstr@p@short@
2899   \let\@Glsxtrshort@\@Glsxtr@p@short@
2900   \let\@GLSxtrshort@\@GLSxtr@p@short@
2901   \let\@glstrlong@\@glstr@p@long@
2902   \let\@Glsxtrlong@\@Glsxtr@p@long@
2903   \let\@GLSxtrlong@\@GLSxtr@p@long@
2904   \let\@glstrshortpl@\@glstr@p@shortpl@
2905   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@

```

```

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

```

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

@glxtrp@text@

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

@Glsxtrp@text@

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

@GLSxtrp@text@

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

lsxtrp@plural@

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

lSxtrp@plural@

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

LSxtrp@plural@

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

glxtrp@short@

```

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

```

Glsxtrp@short@

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

```

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

```

GLSxtr@p@short@

```

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

```

sxtr@p@shortpl@

```

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

```

sxtr@p@shortpl@

```

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

```

Sxtr@p@shortpl@

```

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

```

@glxsxtr@p@long@

```

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

```

@Glsxtr@p@long@

```

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

```

@GLSxtr@p@long@

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

tr@p@acrlongpl@

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

Commands to minimise conflict.

\@glxtrp@opt

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

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

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

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

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

\@@glxtrp

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

Add scope.

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

\@glxtrp

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

\@Glsxtrp

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

```

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

```

\@GLSxtrp

```

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

```

\glsxtr@entry@p

```

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

```

\glsxtrp Not robust as it needs to expand somewhat.

```

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

```

```

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

```

Provide short synonyms for the most common option.

`\glsp`

```

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

```

`\glspt`

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

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

```
3108 \ifdef\texorpdfstring
3109 {
3110   \newcommand{\Glsxtrp}[2]{%
3111     \protect\NoCaseChange
3112     {%
3113       \protect\texorpdfstring
3114       {%
3115         \protect\glsxtrifinmark
3116         {%
3117           \ifcsdef{Glsxtrhead#1}%
3118           {%
3119             {\protect\csuse{Glsxtrhead#1}{#2}}%
3120           }%
3121           {%
3122             \protect\@Gls@entry@field{#2}{#1}%
3123           }%
3124         }%
3125         {%
3126           \@Glsxtrp{#1}{#2}%
3127         }%
3128       }%
3129       {%
3130         \protect\@gls@entry@field{#2}{#1}%
3131       }%
3132     }%
3133   }
3134 }
3135 {
3136   \newcommand{\Glsxtrp}[2]{%
3137     \protect\NoCaseChange
3138     {%
3139       \protect\glsxtrifinmark
3140       {%
3141         \ifcsdef{Glsxtrhead#1}%
3142         {%
3143           {\protect\csuse{Glsxtrhead#1}}%
3144         }%
3145         {%
3146           \protect\@Gls@entry@field{#2}{#1}%
3147         }%
3148       }%
3149       {%
3150         \@Glsxtrp{#1}{#2}%
3151       }%
3152     }%
3153   }
3154 }
```

```

3152     }%
3153 }
3154 }

```

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

```

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

```

```

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

```

1.3.5 Entry Counting

The (use) entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine \gls etc to use \cgl instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceeding the specified threshold. For link counting, see Section 1.4.

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be temporarily disabled, \@glsunset is let to \@glsxtr@unset, which performs the actual unsetting through \@glsunset and then does the hook. This means that the unsetting (and the hook) can be switched off by redefining \@glsunset and then switched back on again by changing the definition back to \@glsxtr@unset.

```

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

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

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

```

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

```

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

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

```

```

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

```

tUnsetBuffering Starred version checks for duplicates.

```

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

```

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

```

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

```

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using \xifinlist as the added complexity might cause problems that the buffering is trying to overcome.)

```

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

```

pUnsetBuffering

```

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

```

pUnsetBuffering Unstarred form (global unset).

```

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

```

pUnsetBuffering Starred form (local unset).

```

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

```

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

```

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

```

```

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

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

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

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

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

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

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

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

```

```

3282     }%
3283     }%
3284 }%
3285 }

```

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

```

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

```

Enable entry counting:

```

3287   \glsenableentrycount

```

Redefine `\gls` etc:

```

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

```

Set the `entrycount` attribute:

```

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

```

In case this command is used again:

```

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

```

`ycountunsetattr`

```

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

```

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

`nableentrycount`

```

3310 \renewcommand*{\glsenableentrycount}{%

```

Enable new fields:

```

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

```

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

```

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

```

New commands to access new fields:

```

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

```

Adjust post unset and reset:

```

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

```

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

```

3349 \let\@cgls@\@cgls@
3350 \let\@cglspl@\@cglspl@

3351 \let\@cGls@\@cGls@
3352 \let\@cGlspl@\@cGlspl@
3353 \let\@cGLS@\@cGLS@

```

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

The rest is as the original definition.

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

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

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

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

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

```

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

```

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

\@@cgl s@

```

3402 \def\@@cgl s@#1#2[#3]{%
3403   \gl sxtrifcounttrigger{#2}%
3404   {%
3405     \cgl sformat{#2}{#3}%
3406     \gl sunset{#2}%
3407   }%
3408   {%
3409     \@gl s@{#1}{#2}[#3]%
3410   }%
3411 }%

```

\@@cgl spl@

```

3412 \def\@@cgl spl@#1#2[#3]{%
3413   \gl sxtrifcounttrigger{#2}%
3414   {%
3415     \cgl splformat{#2}{#3}%
3416     \gl sunset{#2}%
3417   }%
3418   {%
3419     \@gl spl@{#1}{#2}[#3]%
3420   }%
3421 }%

```

\@@cGl s@

```

3422 \def\@@cGl s@#1#2[#3]{%
3423   \gl sxtrifcounttrigger{#2}%
3424   {%
3425     \cGl sformat{#2}{#3}%
3426     \gl sunset{#2}%
3427   }%
3428   {%
3429     \@Gl s@{#1}{#2}[#3]%
3430   }%
3431 }%

```

\@@cGlsp1@

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

\@@cGLS@

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

\@@cGLSp1@

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

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

\@cgl's@

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

\@cGL's@

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

\@cgl'spl@

```
3464 \def\@cgl'spl@#1#2[#3]{\@gl'spl@{#1}{#2}[#3]}
```

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

Add all upper case versions not provided by glossaries.

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

```
\@cGLS Defined the un-starred form. Need to determine if there is a final optional argument
3467 \newcommand*{\@cGLS}[2][\%
3468 \new@ifnextchar[\@cGLS@{#1}{#2}]{\@cGLS@{#1}{#2}[ ]}%
3469 }
```

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

```
\cGLSformat Format used by \cGLS if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.
3471 \newcommand*{\cGLSformat}[2]{\%
3472 \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
3473 }
```

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

```
\@cGLSpl Defined the un-starred form. Need to determine if there is a final optional argument
3475 \newcommand*{\@cGLSpl}[2][\%
3476 \new@ifnextchar[\@cGLSpl@{#1}{#2}]{\@cGLSpl@{#1}{#2}[ ]}%
3477 }
```

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

```
\cGLSplformat Format used by \cGLSpl if entry only used once on previous run. The first argument is the
label, the second argument is the insert text.
3479 \newcommand*{\cGLSplformat}[2]{\%
3480 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3481 }
```

Modify the trigger formats to check for the regular attribute.

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

\cGlsformat

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

\cGlsplformat

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

\cGlsplformat

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

New code similar to above for unit counting.

defunitcounters

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

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

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

@addunitcounter

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

r@ifunitcounter

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

urrentunitcount

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

eviousunitcount

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

t@currunitcount

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

t@currunitcount

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

```

```

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

```

r@currunitcount

```

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

```

r@prevunitcount

```

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

```

eentryunitcount

```

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

```

```

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

```

Access total count:

```

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

```

Access max value:

```

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

```

Adjust post unset and reset:

```

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

```

```

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

```

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

```

3652 \let\@cgls@\@cgls@
3653 \let\@cglspl@\@cglspl@

3654 \let\@cGls@\@cGls@
3655 \let\@cGlspl@\@cGlspl@
3656 \let\@cGLS@\@cGLS@
3657 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3658 \AtEndDocument{\@gls@write@entryunitcounts}%
3659 \renewcommand*{\@gls@entry@unitcount}[3]{%
3660   \csgdef{glo@glsdetoklabel{##1}@prevunit@##3}{##2}%
3661   \ifcsundef{glo@glsdetoklabel{##1}@prevunittotal}%
3662   {\csgdef{glo@glsdetoklabel{##1}@prevunittotal}{##2}}%
3663   {%
3664     \csxdef{glo@glsdetoklabel{##1}@prevunittotal}{
3665       \number\numexpr\csuse{glo@glsdetoklabel{##1}@prevunittotal}+##2}%
3666     }%
3667     \ifcsundef{glo@glsdetoklabel{##1}@prevunitmax}%
3668     {\csgdef{glo@glsdetoklabel{##1}@prevunitmax}{##2}}%
3669     {%
3670       \ifnum\csuse{glo@glsdetoklabel{##1}@prevunitmax}<##2
3671       \csgdef{glo@glsdetoklabel{##1}@prevunitmax}{##2}%
3672       \fi
3673     }%
3674   }%
3675 \let\glsenableentryunitcount\relax
3676 \renewcommand*{\glsenableentrycount}{%
3677   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3678     can't be used with \string\glsenableentryunitcount}%
3679   {Use one or other but not both commands}%
3680 }%
3681 }
3682 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3683 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3684 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3685   \immediate\write\@auxout

```

```

3686   {\string\@gls@entry@unitcount
3687     {\@glsentry}%
3688     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3689     }%
3690     {#1}}%
3691 }

```

entryunitcounts

```

3692 \newcommand*{\@gls@write@entryunitcounts}{%
3693   \immediate\write\@auxout
3694   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3695   \count@=0\relax
3696   \forallglsentries{\@glsentry}{%
3697     \glshasattribute{\@glsentry}{unitcount}%
3698     {%
3699       \ifglsused{\@glsentry}%
3700       {%
3701         \forlistcsloop
3702           {\@gls@write@entryunitcounts@do}%
3703           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3704         }%
3705         {}%
3706         \advance\count@ by \@ne
3707       }%
3708     }%
3709   }%
3710   \ifnum\count@=0
3711     \GlossariesExtraWarningNoLine{Entry counting has been enabled
3712     \MessageBreak with \string\glsenableentryunitcount\space but the
3713     \MessageBreak attribute ‘unitcount’ hasn’t
3714     \MessageBreak been assigned to any of the defined
3715     \MessageBreak entries}%
3716   \fi
3717 }

```

tryUnitCounting The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

```

3718 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3719   \glsenableentryunitcount

```

Redefine \gls etc:

```

3720   \renewcommand*{\gls}{\cgl}%
3721   \renewcommand*{\Gls}{\cGls}%
3722   \renewcommand*{\glspl}{\cglspl}%
3723   \renewcommand*{\Glspl}{\cGlspl}%
3724   \renewcommand*{\GLS}{\cGLS}%
3725   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```
3726 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
3727 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
3728 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3729 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3730 can't be used with \string\GlsXtrEnableEntryUnitCounting}%
3731 {Use one or other but not both commands}}%
3732 }
```

tcountunsetattr

```
3733 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3734 \@for\@glsxtr@cat:=#1\do
3735 {%
3736 \ifdefempty{\@glsxtr@cat}{}%
3737 {%
3738 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3739 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3740 }%
3741 }%
3742 }
```

1.3.6 Acronym Modifications

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

nericNewAcronym

```
3743 \renewcommand*{\SetGenericNewAcronym}{%
3744 \let\@Gls@entryname\@Gls@acrenryname
3745 \renewcommand{\newacronym}[4][ ]{%
3746 \ifdefempty{\@glsacronymlists}%
3747 {%
3748 \def\@glo@type{\acronymtype}%
3749 \setkeys{glossentry}{##1}%
3750 \DeclareAcronymList{\@glo@type}%
3751 }%
3752 {}%
3753 \glskeylisttok{##1}%
3754 \glslabeltok{##2}%
3755 \glsshorttok{##3}%
3756 \glslongtok{##4}%
3757 \newacronymhook
```

```

3758 \protected@edef\@do@newglossaryentry{%
3759 \noexpand\newglossaryentry{\the\glslabeltok}%
3760 {%
3761 type=\acronymtype,%
3762 name={\expandonce{\acronymentry{##2}}},%
3763 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3764 text={\the\glsshorttok},%
3765 short={\the\glsshorttok},%
3766 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3767 long={\the\glslongtok},%
3768 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3769 category=acronym,
3770 \GenericAcronymFields,%
3771 \the\glskeylisttok
3772 }%
3773 }%
3774 \@do@newglossaryentry
3775 }%
3776 \renewcommand*{\acrfullfmt}[3]{%
3777 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3778 \renewcommand*{\Acrfullfmt}[3]{%
3779 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3780 \renewcommand*{\ACRfullfmt}[3]{%
3781 \glslink[##1]{##2}{%
3782 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
3783 \renewcommand*{\acrfullplfmt}[3]{%
3784 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3785 \renewcommand*{\Acrfullplfmt}[3]{%
3786 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3787 \renewcommand*{\ACRfullplfmt}[3]{%
3788 \glslink[##1]{##2}{%
3789 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
3790 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
3791 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
3792 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
3793 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
3794 }

```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```

3795 \let\@glxstr@org@setacronymstyle\setacronymstyle
3796 \let\@glxstr@org@newacronymstyle\newacronymstyle

```

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

```

3797 \newcommand*{\MakeAcronymsAbbreviations}{%
3798 \renewcommand*{\newacronym}[4][]{%

```

```

3799 \glstr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3800 }%
3801 \renewcommand*{\firstacronymfont}[1]{\glstrfirstabbrvfont{##1}}%
3802 \renewcommand*{\acronymfont}[1]{\glstrabbrvfont{##1}}%
3803 \renewcommand*{\setacronymstyle}[1]{%
3804 \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
3805 unavailable.
3806 Use \string\setabbreviationstyle\space instead.
3807 The original acronym interface can be restored with
3808 \string\RestoreAcronyms}{}}%
3809 }%
3810 \renewcommand*{\newacronymstyle}[1]{%
3811 \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3812 available unless you restore the original acronym interface with
3813 \string\RestoreAcronyms}}%
3814 \@glstr@org@newacronymstyle{##1}%
3815 }%
3816 }

```

Switch acronyms to abbreviations:

```
3817 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3818 \newcommand*{\RestoreAcronyms}{%
3819 \SetGenericNewAcronym
3820 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3821 \renewcommand{\acronymfont}[1]{##1}%
3822 \let\setacronymstyle\@glstr@org@setacronymstyle
3823 \let\newacronymstyle\@glstr@org@newacronymstyle

```

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

```

3824 \renewcommand*{\@gls@link@checkfirsthyper{%
3825 \ifglstrused{\glslabel}}%
3826 {\let\glstrifwasfirstuse\@secondoftwo}
3827 {\let\glstrifwasfirstuse\@firstoftwo}}%
3828 \@glstr@org@checkfirsthyper
3829 }
3830 \glssetcategoryattribute{acronym}{regular}{false}%
3831 \setacronymstyle{long-short}%
3832 }

```

\glsacspace Allow the user to customise the maximum value.

```

3833 \renewcommand*{\glsacspace}[1]{%
3834 \settowidth{\dimen@}{(\firstacronymfont{\glstrentryshort{#1}})}%
3835 \ifdim\dimen@<\glsacspacemax~\else\space\fi
3836 }

```

\glsacspacemax Value used in the above.

```
3837 \newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the “noidx” commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require `makeindex/xindy`.

r@reg@glosslist

```
3838 \newcommand*{\@glxtr@reg@glosslist}{{}
```

Save the original definition of `\makeglossaries`:

```
3839 \let\@glxtr@org@makeglossaries\makeglossaries
```

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

\makeglossaries

```
3840 \renewcommand*{\makeglossaries}[1][]{%
3841   \ifx\@glxtr@record@setting\@glxtr@record@setting@only
3842     \PackageError{glossaries-extra}{\string\makeglossaries\space
3843       not permitted\MessageBreak with record=only package option}%
3844     {You may only use \string\makeglossaries\space with
3845       record=off or record=alsoindex options}%
3846   \else
3847     \ifblank{#1}%
3848       {\@glxtr@org@makeglossaries}%
3849       {%
3850         \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3851           \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3852             not permitted\MessageBreak with record=alsoindex package option}%
3853           {You may only use the hybrid \string\makeglossaries[...]\space with
3854             record=off option}%
3855         \else
3856           \edef\@glxtr@reg@glosslist{#1}%
3857           \ifundef{\glswrite}{\newwrite\glswrite}{}%
3858           \protected@write\@auxout{}{\string\providecommand
3859             \string\@glxtr@order[1]}{}
3860           \protected@write\@auxout{}{\string\providecommand
3861             \string\@istfilename[1]}{}
3862           \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3863           \protected@write\@auxout{}{\string\@glxtr@order{\glxtr@order}}
3864           \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}
3865           \write\@auxout{\string\providecommand\string\@glxtr@reference[3]}{}%
3866         }%
3867       }%
3868     }
```

Iterate through each supplied glossary type and activate it.

```
3866   \@for\@glo@type:=#1\do{%
3867     \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
3868   }
```

New glossaries must be created before `\makeglossaries`:

```
3869 \renewcommand*\newglossary[4] [] {%
3870 \PackageError{glossaries}{New glossaries
3871 must be created before \string\makeglossaries}{You need
3872 to move \string\makeglossaries\space after all your
3873 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
3874 \let\@makeglossary\relax
3875 \let\makeglossary\relax
3876 \renewcommand\makeglossaries[1] [] {}%
```

Disable all commands that have no effect after `\makeglossaries`

```
3877 \@disable@onlypremakeg
```

Allow see key:

```
3878 \let\gls@checkseeallowed\relax
```

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

```
3879 \renewcommand*\@do@seeglossary[2] {%
3880 \glsdoifexists{##1}%
3881 {%
3882 \edef\@gls@label{\glsdetoklabel{##1}}%
3883 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3884 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3885 {\@glsxtr@org@doseeglossary{##1}{##2}}%
3886 {%
3887 \@@glsxtrwrglossmark
3888 \protected@write\@auxout{}{%
3889 \string\@gls@reference
3890 {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
3891 }%
3892 }%
3893 }%
3894 }%
```

Adjust `\@do@@wrglossary`

```
3895 \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
3896 \def\@do@@wrglossary{%
3897 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3898 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3899 {\@glsxtr@@do@@wrglossary}%
3900 {\@gls@noidxglossary}%
3901 }%
```

Suppress warning about no `\makeglossaries`

```
3902 \let\warn@nomakeglossaries\relax
3903 \def\warn@noprintglossary{%
3904 \GlossariesWarningNoLine{No \string\printglossary\space
3905 or \string\printglossaries\space
```

```

3906         found.^^J(Remove \string\makeglossaries\space if you don't want
3907         any glossaries.)^^JThis document will not have a glossary}%
3908     }%

```

Only warn for glossaries not listed.

```

3909     \renewcommand{\@gls@noref@warn}[1]{%
3910         \edef\@gls@type{##1}%
3911         \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3912         {%
3913             \GlossariesExtraWarning{Can't use
3914                 \string\printnoidxglossary[type={\@gls@type}]
3915                 when '\@gls@type' is listed in the optional argument of
3916                 \string\makeglossaries}%
3917         }%
3918     }%
3919     \GlossariesWarning{Empty glossary for
3920         \string\printnoidxglossary[type={##1}].
3921         Rerun may be required (or you may have forgotten to use
3922         commands like \string\gls)}%
3923 }%
3924 }%

```

Adjust display number list to check for type:

```

3925     \renewcommand*{\glsdisplaynumberlist}[1]{%
3926         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3927         {\@glsxtr@idx@displaynumberlist{##1}}%
3928         {\@glsxtr@noidx@displaynumberlist{##1}}%
3929     }%

```

Adjust entry list:

```

3930     \renewcommand*{\glsentrynumberlist}[1]{%
3931         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3932         {\@glsxtr@idx@entrynumberlist{##1}}%
3933         {\@glsxtr@noidx@entrynumberlist{##1}}%
3934     }%

```

Adjust number list loop

```

3935     \renewcommand*{\glsnumberlistloop}[2]{%
3936         \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3937         {%
3938             \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3939                 not available for glossary '##1'}{%
3940             }%
3941             {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3942         }%

```

Only sanitize sort for normal indexing glossaries.

```

3943     \renewcommand*{\glsprestandardsort}[3]{%
3944         \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3945         {%
3946             \glsdosanitizesort

```

```

3947     }%
3948     {%
3949         \ifglssanitizesort
3950         \@gls@noidx@sanitizesort
3951         \else
3952         \@gls@noidx@nosanitizesort
3953         \fi
3954     }%
3955 }%

```

Unlike `\makenoidxglossaries` we can't automatically set `sanitizesort=false`. All entries must be defined in the preamble.

```

3956     \renewcommand*\new@glossaryentry[2]{%
3957         \PackageError{glossaries-extra}{Glossary entries must be defined
3958             in the preamble\MessageBreak when you use the optional argument
3959             of \string\makeglossaries}{Either move your definitions to the
3960             preamble or don't use the optional argument of
3961             \string\makeglossaries}%
3962     }%

```

Only activate sort key for glossaries that aren't listed in #1 (glossary label is stored in `\@glo@type` but this defaults to `\glsdefaulttype` so some expansion is required).

```

3963     \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
3964     \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3965         \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
3966         type=\glsdefaulttype,\@end@glsxtr@gettype
3967         \def\@glo@sorttype{\@glo@default@sorttype}%
3968     }%

```

Check automake setting:

```

3969     \ifglssautomake
3970     \renewcommand*\@gls@doautomake{%
3971         \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
3972             \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3973         }%
3974     }%
3975 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3976     \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3977     \fi
3978 }%
3979 \fi
3980 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\rgprintglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes

\provideignoredglossary to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

3981 \newcommand{\@glxtr@orgprintglossary}[2]{%
3982   \def\@glo@type{\glsdefaulttype}%
3983   \def\glossarytitle{%
3984     \ifcsdef{\@glo@type\@glo@type @title}%
3985       {\csuse{\@glo@type\@glo@type @title}}%
3986       {\glossaryname}}%
3987   \def\glossarytoctitle{\glossarytitle}%
3988   \let\org@glossarytitle\glossarytitle
3989   \def\@glossarystyle{%
3990     \ifx\@glossary@default@style\relax
3991       \GlossariesWarning{No default glossary style provided \MessageBreak
3992         for the glossary '\@glo@type'. \MessageBreak
3993         Using deprecated fallback. \MessageBreak
3994         To fix this set the style with \MessageBreak
3995         \string\setglossarystyle\space or use the \MessageBreak
3996         style key=value option}%
3997     \fi
3998   }%
3999   \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
4000   \let\@org@glossaryentrynumbers\glossaryentrynumbers
4001   \bgroup
4002     \@printgloss@setsort
4003     \setkeys{printgloss}{#1}%
4004     \ifx\glossarytitle\org@glossarytitle
4005     \else
4006       \cslet{\@glo@type\@glo@type @title}{\glossarytitle}%
4007     \fi
4008     \let\currentglossary\@glo@type
4009     \let\org@glossaryentrynumbers\glossaryentrynumbers
4010     \let\glsnonextpages\@glsnonextpages
4011     \let\glsnextpages\@glsnextpages
4012     \glsxtractivatenopost
4013     \gls@dotoc@title
4014     \@glossarystyle
4015     \let\gls@org@glossaryentryfield\glossentry
4016     \let\gls@org@glossarysubentryfield\subglossentry
4017     \renewcommand{\glossentry}[1]{%
4018       \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4019       \gls@org@glossaryentryfield{##1}%
4020     }%
4021     \renewcommand{\subglossentry}[2]{%
4022       \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4023       \gls@org@glossarysubentryfield{##1}{##2}%
4024     }%
4025     \@gls@preglossaryhook

```

```

4026      #2%
4027 \egroup
4028 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4029 \global\let\warn@noprintglossary\relax
4030 }

```

ractivatenopost Change \nopostdesc and \glxtrnopostpunc to behave as they do in the glossary.

```

4031 \newcommand*{\glxtractivatenopost}{%
4032 \let\nopostdesc\@nopostdesc
4033 \let\glxtrnopostpunc\@glxtr@nopostpunc
4034 }

```

lsxtrnopostpunc

```

4035 \newrobustcmd*{\glxtrnopostpunc}{%

```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

4036 \newcommand{\@glxtr@nopostpunc}{%
4037 \let\@glxtr@org@postdescription\glspostdescription
4038 \ifglsnopostdot
4039 \renewcommand{\glspostdescription}{%
4040 \glsnopostdottrue
4041 \let\glspostdescription\@glxtr@org@postdescription
4042 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4043 \glxtrpostdescription
4044 \@glxtr@nopostpunc@postdesc}%
4045 \else
4046 \renewcommand{\glspostdescription}{%
4047 \let\glspostdescription\@glxtr@org@postdescription
4048 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4049 \glxtrpostdescription
4050 \@glxtr@nopostpunc@postdesc}%
4051 \fi
4052 \glsnopostdotfalse
4053 }

```

stpunc@postdesc

```

4054 \newcommand*{\@glxtr@nopostpunc@postdesc}{%

```

estore@postpunc

```

4055 \newcommand*{\@glxtr@restore@postpunc}{%
4056 \def\@glxtr@nopostpunc@postdesc{%
4057 \@glxtr@org@postdescription
4058 \let\@glxtr@nopostpunc@postdesc\@empty
4059 \let\glxtrrestorepostpunc\@empty
4060 }%
4061 }

```

restorepostpunc Does nothing outside of glossary.

```
4062 \newcommand*{\glxtrrestorepostpunc}{}

\@printglossary Redefine.
```

```
4063 \renewcommand{\@printglossary}[2]{%
4064   \def\@glxtr@printglossopts{#1}%
4065   \@glxtr@orgprintglossary{#1}{#2}%
4066 }
```

Add a key that switches off the entry targets:

```
4067 \define@choicekey{printgloss}{target}
4068 [\@glxtr@printglossval\@glxtr@printglossnr]%
4069 {true,false}[true]%
4070 {%
4071   \ifcase\@glxtr@printglossnr

4072   \def\@glstarget{\glsdohypertarget}%
4073   \else
4074   \let\@glstarget\@secondoftwo
4075   \fi
4076 }
```

hypernameprefix

```
4077 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:
```

```
4078 \define@key{printgloss}{targetnameprefix}{%
4079   \renewcommand{\@glxtrhypernameprefix}{#1}%
4080 }

4081 \define@key{printgloss}{prefix}{%
4082   \renewcommand{\glolinkprefix}{#1}%
4083 }
```

glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```
4084 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4085 \renewcommand{\glsdohypertarget}[2]{%
4086   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4087 }
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
4088 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4089   \def\@glstarget{\glsdohypertarget}%
4090 \fi
4091 \end{macro}
```

@makeglossaries For the benefit of makeglossaries

```
4092 \newcommand*{\glxtr@makeglossaries}[1]{}

122
```

@glxtr@gettype Get just the type.

```
4093 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4094   \def\@glo@type{#2}%
4095 }
```

@assign@sortkey Assign the sort key.

```
4096 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4097   \edef\@glo@type{\@glo@type}%
4098   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4099   {%
4100     \@glo@no@assign@sortkey{#1}%
4101   }%
4102   {%
4103     \@glo@assign@sortkey{#1}%
4104   }%
4105 }
```

Display number list for the regular version:

splaynumberlist

```
4106 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
4107 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4108   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4109   \ifdef\@gls@loclist
4110   {%
4111     \def\@gls@noidxloclist@sep{%
4112       \def\@gls@noidxloclist@sep{%
4113         \def\@gls@noidxloclist@sep{%
4114           \glsnumlistsep
4115         }%
4116         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4117       }%
4118     }%
4119     \def\@gls@noidxloclist@finalsep{}%
4120     \def\@gls@noidxloclist@prev{}%
4121     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4122     \@gls@noidxloclist@finalsep
4123     \@gls@noidxloclist@prev
4124   }%
4125   {%
4126     \glxtrundeftag
4127     \glsdoifexists{#1}%
4128     {%
4129       \GlossariesWarning{Missing location list for ‘#1’. Either
4130         a rerun is required or you haven’t referenced the entry.}%

```

```

4131 }%
4132 }%
4133 }%
4134

```

And for the number list loop:

@numberlistloop

```

4135 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
4136   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4137   \let\@gls@org@glsnoidxdisplayloc@glsnoidxdisplayloc
4138   \let\@gls@org@glsseeformat@glsseeformat
4139   \let\glsnoidxdisplayloc#2\relax
4140   \let\glsseeformat#3\relax
4141   \ifdef\@gls@loclist
4142   {%
4143     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4144   }%
4145   {%

4146     \glsxtrundeftag
4147     \glsdoifexists{#1}%
4148     {%
4149       \GlossariesWarning{Missing location list for ‘##1’. Either
4150         a rerun is required or you haven’t referenced the entry.}%
4151     }%
4152   }%
4153   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4154   \let\glsseeformat\@gls@org@glsseeformat
4155 }%

```

Same for entry number list.

entrynumberlist

```

4156 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4157   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4158   \ifdef\@gls@loclist
4159   {%
4160     \glsnoidxloclist{\@gls@loclist}%
4161   }%
4162   {%

4163     \glsxtrundeftag
4164     \glsdoifexists{#1}%
4165     {%
4166       \GlossariesWarning{Missing location list for ‘#1’. Either
4167         a rerun is required or you haven’t referenced the entry.}%
4168     }%
4169   }%
4170 }%

```

entrynumberlist

```
4171 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
4172 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4173   \protected@edef\@glsxtr@titlelabel{#1}%
4174   \ifdefvoid\@glsxtr@titlelabel
4175   {%
4176     {%
4177       \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4178     }%
4179     \ifdefvoid{\@glsxtr@titlelabel}%
4180     {%
4181       \DTLifint{#1}%
4182       {%
4183         \ifnum#1<256\relax
4184           \edef#2{\char#1\relax}%
4185         \else
4186           \edef#2{#1}%
4187         \fi
4188       }%
4189       {%
4190         \ifcsundef{#1groupname}%
4191         {\def#2{#1}}%
4192         {\letcs#2{#1groupname}}%
4193       }%
4194     }%
4195     {%
4196       \let#2\@glsxtr@titlelabel
4197     }%
4198 }
```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```
4199 \let\glsxtr@org@getgrouptitle\@gls@getgrouptitle
```

trgetgrouptitle Provide a user-level command to fetch the group title. The first argument is the group label.
The second argument is a control sequence in which to store the title.

```
4200 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4201   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
4202   \@onelevel@sanitize\@glsxtr@titlelabel
4203   \ifcsdef{\@glsxtr@titlelabel}
4204   {\letcs{#2}{\@glsxtr@titlelabel}}%
4205   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4206 }
4207 \let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

trsetgrouptitle Sets the title for the given group label.

```
4208 \newcommand{\glsxtrsetgrouptitle}[2]{%
```

```

4209 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
4210 \onelevel@sanitize\@glstr@titlelabel
4211 \protected@csxdef{\@glstr@titlelabel}{#2}%
4212 }

```

`\alsetgrouptitle` As above put only locally defines the title.

```

4213 \newcommand{\glstrlocalsetgrouptitle}[2]{%
4214 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
4215 \onelevel@sanitize\@glstr@titlelabel
4216 \protected@csedef{\@glstr@titlelabel}{#2}%
4217 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4218 \renewcommand*{\glsnavigation}{%
4219 \def\@gls@between{%
4220 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
4221 {%
4222 \def\@gls@list{%
4223 }%
4224 {%
4225 \expandafter\let\expandafter\@gls@list
4226 \csname \@gls@hypergroup@list@\@glo@type\endcsname
4227 }%
4228 \@for\@gls@tmp:=\@gls@list\do{%
4229 \@gls@between
4230 \glstrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4231 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4232 \let\@gls@between\glshypernavsep
4233 }%
4234 }

```

`@noidx@glossary`

```

4235 \renewcommand*{\@print@noidx@glossary}{%
4236 \ifcsdef{\@glsref@\@glo@type}%
4237 {%
4238 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
4239 {%
4240 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4241 }%
4242 {%
4243 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4244 }%
4245 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4246 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4247 \def\@gls@currentlettergroup{%
4248 \begin{theglossary}%

```

```

4249 \glossaryheader
4250 \glsresetentrylist
4251 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
4252 \end{theglossary}%
4253 \glossarypostamble
4254 }%
4255 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4256 \glstrifemptyglossary{\@glo@type}%
4257 }%
4258 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4259 \@gls@noref@warn{\@glo@type}%
4260 }%
4261 }

```

`noidxdisplayloc` Patch to check for range formations.

```

4262 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4263 \setentrycounter[#1]{#2}%
4264 \@glstr@display@loc#3\empty\end@glstr@display@loc{#4}%
4265 }

```

`xtr@display@loc` Patch to check for range formations.

```

4266 \def\@glstr@display@loc#1#2\end@glstr@display@loc#3{%
4267 \ifx#1\relax
4268 \glstrdisplaystartloc{#2}{#3}%
4269 \else
4270 \ifx#1)\relax
4271 \glstrdisplayendloc{#2}{#3}%
4272 \else
4273 \glstrdisplaysingleloc{#1#2}{#3}%
4274 \fi
4275 \fi
4276 }

```

`isplaysingleloc` Single location.

```

4277 \newcommand*{\glstrdisplaysingleloc}[2]{%
4278 \csuse{#1}{#2}%
4279 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glstrlocrangefmt`.

`displaystartloc` Start of a location range.

```

4280 \newcommand*{\glstrdisplaystartloc}[2]{%
4281 \edef\glstrlocrangefmt{#1}%
4282 \ifx\glstrlocrangefmt\empty
4283 \def\glstrlocrangefmt{\glsnumberformat}%

```

```

4284 \fi
4285 \expandafter\glxtrdisplaysingleloc
4286 \expandafter{\glxtrlocrangefmt}{#2}%
4287 }

```

`trdisplayendloc` End of a location range.

```

4288 \newcommand*{\glxtrdisplayendloc}[2]{%
4289 \edef\@glxtr@tmp{#1}%
4290 \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{glnumberformat}}{}%
4291 \ifx\glxtrlocrangefmt\@glxtr@tmp
4292 \else
4293 \GlossariesExtraWarning{Mismatched end location range
4294 (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
4295 \fi
4296 \expandafter\glxtrdisplayendloohook\expandafter{\@glxtr@tmp}{#2}%
4297 \expandafter\glxtrdisplaysingleloc
4298 \expandafter{\glxtrlocrangefmt}{#2}%
4299 \def\glxtrlocrangefmt{}%
4300 }

```

`splayendloohook` Allow the user to hook into the end of range command.

```

4301 \newcommand*{\glxtrdisplayendloohook}[2]{

```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```

4302 \newcommand*{\glxtrlocrangefmt}{}

```

`setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4303 \renewcommand*{\setentrycounter}[2][{}]{%
4304 \def\glxtrcounterprefix{#1}%
4305 \ifx\glxtrcounterprefix\@empty
4306 \def\@glo@counterprefix{.}%
4307 \else
4308 \def\@glo@counterprefix{.#1.}%
4309 \fi
4310 \def\glsetentrycounter{#2}%
4311 }

```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4312 \def\@gls@removespaces#1 #2\@nil{%
4313 \toks@=\expandafter{\the\toks@#1}%
4314 \ifx\@#2\%
4315 \edef\x{\the\toks@}%
4316 \ifx\x\empty
4317 \else
4318 \expandafter\glxtrlocationhyperlink\expandafter
4319 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4320 \fi

```

```

4321 \else
4322   \@gls@ReturnAfterFi{%
4323     \@gls@removespaces#2\@nil
4324   }%
4325 \fi
4326 }

```

cationhyperlink

```

4327 \newcommand*{\glstrlocationhyperlink}[3]{%
4328   \ifdefvoid\glstrsupplocationurl
4329   {%
4330     \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4331   }%
4332   {%
4333     \hyperref{\glstrsupplocationurl}{#1#2#3}{#3}%
4334   }%
4335 }

```

supphypernumber

```

4336 \newcommand*{\glstrsupphypernumber}[1]{%
4337   {%
4338     \glshasattribute{\glscurrententrylabel}{externallocation}%
4339     {%
4340       \def\glstrsupplocationurl{%
4341         \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4342     }%
4343     {%
4344       \def\glstrsupplocationurl{}%
4345     }%
4346     \glshypernumber{#1}%
4347   }%
4348 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4349 \renewcommand{\@print@glossary}{%
4350   \makeatletter
4351   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4352   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4353   {%
4354     {\glstrNoGlossaryWarning{\@glo@type}}%
4355     \ifglsxindy
4356       \ifcsundef{@xdy@\@glo@type @language}%
4357       {%
4358         \edef\@do@auxoutstuff{%
4359           \noexpand\AtEndDocument{%
4360             \noexpand\immediate\noexpand\write\@auxout{%

```

```

4361         \string\providecommand\string\@xdylanguage[2]{}%
4362         \noexpand\immediate\noexpand\write\@auxout{%
4363         \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4364     }%
4365 }%
4366 }%
4367 {%
4368     \edef\@do@auxoutstuff{%
4369     \noexpand\AtEndDocument{%
4370     \noexpand\immediate\noexpand\write\@auxout{%
4371     \string\providecommand\string\@xdylanguage[2]{}%
4372     \noexpand\immediate\noexpand\write\@auxout{%
4373     \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4374     @language\endcsname}}%
4375     }%
4376     }%
4377     }%
4378     \@do@auxoutstuff
4379     \edef\@do@auxoutstuff{%
4380     \noexpand\AtEndDocument{%
4381     \noexpand\immediate\noexpand\write\@auxout{%
4382     \string\providecommand\string\@gls@codepage[2]{}%
4383     \noexpand\immediate\noexpand\write\@auxout{%
4384     \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4385     }%
4386     }%
4387     \@do@auxoutstuff
4388     \fi
4389     \renewcommand*{\@warn@nomakeglossaries}{%
4390     \GlossariesWarningNoLine{\string\makeglossaries\space
4391     hasn't been used,^^Jthe glossaries will not be updated}%
4392     }%
4393 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4394 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4395 This document is incomplete. The external file associated with
4396 the glossary '#1' (which should be called \texttt{#2})
4397 hasn't been created.%
4398 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4399 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4400 This has probably happened because there are no entries defined
4401 in this glossary.%
4402 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4403 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4404   If you don't want this glossary,
4405   add \texttt{nomain} to your package option list when you load
4406   \texttt{glossaries-extra.sty}. For example:%
4407 }

```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```

4408 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4409   Did you forget to use \texttt{type=#1} when you defined your
4410   entries? If you tried to load entries into this glossary with
4411   \texttt{\string\loadglsentries} did you remember to use
4412   \texttt{[#1]} as the optional argument? If you did, check that
4413   the definitions in the file you loaded all had the type set
4414   to \texttt{\string\glsdefaulttype}.%
4415 }

```

arningCheckFile Advisory message to check the file contents.

```

4416 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4417   Check the contents of the file \texttt{#1}. If
4418   it's empty, that means you haven't indexed any of your entries in this
4419   glossary (using commands like \texttt{\string\gls} or
4420   \texttt{\string\glsadd}) so this list can't be generated.
4421   If the file isn't empty, the document build process hasn't been
4422   completed.%
4423 }

```

WarningAutoMake Message when automake option has been used.

```

4424 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4425   You may need to rerun \LaTeX. If you already have, it may be that
4426   \TeX's shell escape doesn't allow you to run
4427   \ifglxindy xindy\else makeindex\fi. Check the
4428   transcript file \texttt{\jobname.log}. If the shell escape is
4429   disabled, try one of the following:
4430
4431   \begin{itemize}
4432     \item Run the external (Lua) application:
4433
4434       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4435
4436     \item Run the external (Perl) application:
4437
4438       \texttt{makeglossaries \string"\jobname\string"}
4439   \end{itemize}
4440
4441   Then rerun \LaTeX\ on this document.
4442   \GlossariesExtraWarning{Rerun required to build the
4443   glossary '#1' or check TeX's shell escape allows
4444   you to run \ifglxindy xindy\else makeindex\fi}%
4445 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4446 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4447   You need to either replace \texttt{\string\makenoidxglossaries}
4448   with \texttt{\string\makeglossaries} or replace
4449   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4450   \texttt{\string\printnoidxglossary}
4451   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4452   this document.%
4453 }

```

arningBuildInfo Build advice.

```

4454 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4455   Try one of the following:
4456   \begin{itemize}
4457     \item Add \texttt{automake} to your package option list when you load
4458       \texttt{glossaries-extra.sty}. For example:
4459
4460       \texttt{\string\usepackage[automake]%
4461         \glsopenbrace glossaries-extra\glsclosebrace}
4462
4463     \item Run the external (Lua) application:
4464
4465     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4466
4467     \item Run the external (Perl) application:
4468
4469     \texttt{makeglossaries \string"\jobname\string"}
4470   \end{itemize}
4471
4472   Then rerun \LaTeX\ on this document.%
4473 }

```

trRecordWarning Paragraph for record=only.

```

4474 \newcommand{\GlsXtrRecordWarning}[1]{%
4475   \texttt{\string\printglossary} doesn't work
4476   with the \texttt{record=only} package option
4477   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4478   instead (or change the package option).%
4479 }

```

oGlsWarningTail Final paragraph.

```

4480 \newcommand{\GlsXtrNoGlsWarningTail}{%
4481   This message will be removed once the problem has been fixed.%
4482 }

```

GlsWarningNoOut No out file created. Build advice.

```

4483 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4484   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4485   \texttt{\string\makeglossaries} or you have used

```

```

4486 \texttt{\string\nofiles}. If this is just a draft version of the
4487 document, you can suppress this message using the
4488 \texttt{nomissingglstext} package option.%
4489 }

```

glossarywarning

```

4490 \newcommand*{\@glstr@defaultnoglossarywarning}[1]{%
4491 \glossarysection[\glossarytoctitle]{\glossarytitle}
4492 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4493 \par
4494 \glstrifemptyglossary{#1}%
4495 {%
4496 \GlsXtrNoGlsWarningEmptyStart\space
4497 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4498 \medskip
4499 \noindent\texttt{\string\usepackage[nomain\ifglstacronym ,acronym\fi]%
4500 \glstopenbrace glossaries-extra\glstclosebrace}
4501 \medskip
4502 }%
4503 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4504 }%
4505 {%
4506 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4507 {%
4508 \GlsXtrNoGlsWarningCheckFile
4509 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4510
4511 \ifglstautomake
4512
4513 \GlsXtrNoGlsWarningAutoMake{#1}
4514
4515 \else
4516
4517 \ifthenelse{\equal{#1}{main}}%
4518 {%
4519 \GlsXtrNoGlsWarningEmptyMain\par
4520 \medskip
4521 \noindent\texttt{\string\usepackage[nomain]%
4522 \glstopenbrace glossaries-extra\glstclosebrace}
4523 \medskip
4524 }%
4525 {}%
4526
4527 \ifdequal\makeglossaries\@no@makeglossaries
4528 {%
4529 \GlsXtrNoGlsWarningMisMatch
4530 }%
4531 {}%
4532 \GlsXtrNoGlsWarningBuildInfo

```

```

4533     }%
4534   \fi
4535 }%
4536 {%
4537   \GlsXtrNoGlsWarningNoOut
4538   {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4539 }%
4540 }%
4541 \par
4542 \GlsXtrNoGlsWarningTail
4543 }

```

`glossarywarning` Warn about using `\printglossary` with `record`

```

4544 \newcommand*{\@glxtr@record@noglossarywarning}[1]{%
4545   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4546     with record=only package option\MessageBreak(use
4547     \string\printunsrtglossary[type=#1])\MessageBreak
4548     instead (or change the package option)}%
4549   \glossarysection[\glossarytoctitle]{\glossarytitle}
4550   \GlsXtrRecordWarning{#1}
4551   \GlsXtrNoGlsWarningTail
4552 }

```

Provide some commands to accompany the `record` option for use with **bib2gls**.

`xltrresourcefile` Since it's dangerous for an external application to create a file with a `.tex` extension, as from v1.11 this enforces a `.glstex` extension to avoid conflict.

```

4553 \newcommand*{\glxtrresourcefile}[2] []{%
  The record option can't be set after this command.
4554   \disable@keys{glossaries-extra.sty}{record}%
4555   \glxtr@writefields
4556   \protected@write\@auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%
4557   \let\@glxtr@org@see@noindex\@gl@see@noindex
4558   \let\@gl@see@noindex\relax
4559   \IfFileExists{#2.glstex}%
4560   {%

```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```

4561     \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4562     \makeatletter
4563     \@input{#2.glstex}%
4564     \@bibgls@restoreat
4565   }%
4566   {%
4567     \GlossariesExtraWarning{No file '#2.glstex'}%
4568   }%
4569   \let\@gl@see@noindex\@glxtr@org@see@noindex
4570 }

```

```

4571 \@onlypreamble\glxtrresourcefile

xtrresourceinit  Code used during the protected write operation.
4572 \newcommand*{\glxtrresourceinit}{%

trresourcecount
4573 \newcount\glxtrresourcecount

trLoadResources  Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.
4574 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
4575   \ifnum\glxtrresourcecount=0\relax
4576     \glxtrresourcefile[#1]{\jobname}%
4577   \else
4578     \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4579   \fi
4580   \advance\glxtrresourcecount by 1\relax
4581 }

glxtr@resource
4582 \newcommand*{\glxtr@resource}[2]{%

\glxtr@fields
4583 \newcommand*{\glxtr@fields}[1]{%

xtr@texencoding
4584 \newcommand*{\glxtr@texencoding}[1]{%

\glxtr@langtag
4585 \newcommand*{\glxtr@langtag}[1]{%

@pluralsuffixes
4586 \newcommand*{\glxtr@pluralsuffixes}[4]{%

tr@shortcutsval
4587 \newcommand*{\glxtr@shortcutsval}[1]{%

sxtr@linkprefix
4588 \newcommand*{\glxtr@linkprefix}[1]{%

xtr@writefields  This information only needs to be written once, so disable it after it's been used.
4589 \newcommand*{\glxtr@writefields}{%

4590   \protected@write\@auxout{%
4591     {\string\providecommand*{\string\glxtr@fields}[1]{}}%
4592   \protected@write\@auxout{%
4593     {\string\providecommand*{\string\glxtr@resource}[2]{}}%
4594   \protected@write\@auxout{%
4595     {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}%

```

```

4596 \protected@write\@auxout{}\%
4597   {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}%
4598 \protected@write\@auxout{}\%
4599   {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}%
4600 \protected@write\@auxout{}\{\string\glxtr@fields{\@glx@keymap}}\%

4601 \protected@write\@auxout{}\%
4602   {\string\providecommand*{\string\glxtr@record}[5]{}}%

```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by `tracklang`). For multilingual documents, the required locale will have to be indicated in the sort key when using `\glxtrresourcefile`.

```

4603 \ifdef\CurrentTrackedLanguageTag
4604 {%
4605   \protected@write\@auxout{}\%
4606     \string\glxtr@langtag{\CurrentTrackedLanguageTag}%
4607 }%
4608 {}%
4609 \protected@write\@auxout{}\{\string\glxtr@pluralsuffixes
4610   {\glxpluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4611   {\glxtrabbrvpluralsuffix}}\%
4612 \ifdef\inputencodingname
4613 {%
4614   \protected@write\@auxout{}\{\string\glxtr@texencoding{\inputencodingname}}\%
4615 }%
4616 {%

```

If `fontspec` has been loaded, assume UTF-8. (The encoding can be changed with `\XeTeXinputencoding`, but I can't work out how to determine the current encoding.)

```

4617   \@ifpackageloaded{fontspec}%
4618     {\protected@write\@auxout{}\{\string\glxtr@texencoding{utf8}}\%
4619     }%
4620 }%
4621 \protected@write\@auxout{}\{\string\glxtr@shortcutsval{\@glxtr@shortcutsval}}\%

```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by `bib2gls` when the external option is used.

```

4622 \AtBeginDocument
4623   {\protected@write\@auxout{}\{\string\glxtr@linkprefix{\glolinkprefix}}\%
4624   \let\glxtr@writefields\relax

```

If the `automake` option is on, try running `bib2gls` if the aux file exists. The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```

4625 \ifglautomake
4626   \IfFileExists{\jobname.aux}%
4627     {\immediate\write18{bib2gls \jobname}}\%

```

If `\makeglossaries` is also used, allow `makeindex/xindy` to also be run, otherwise disable the error message about requiring `\makeglossaries` with `automake=true`.

```

4628 \ifx\@gls@doautomake\@gls@doautomake@err
4629 \let\@gls@doautomake\relax
4630 \fi
4631 \fi
4632 }

```

do@automake@err

```

4633 \newcommand*{\@gls@doautomake@err}{%
4634 \PackageError{glossaries}{You must use
4635 \string\makeglossaries\space with automake=true}
4636 {%
4637 Either remove the automake=true setting or
4638 add \string\makeglossaries\space to your document preamble.%
4639 }%
4640 }

```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```

4641 \newcommand*{\glsxtr@record}[5]{}

```

r@counterrecord Aux file command.

```

4642 \newcommand*{\glsxtr@counterrecord}[3]{%
4643 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4644 }

```

unterrecordhook Hook used by \@glsxtr@dorecord.

```

4645 \newcommand*{\@glsxtr@counterrecordhook}{}

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

4646 \newcommand*{\GlsXtrRecordCounter}[1]{%
4647 \@glsxtr@recordcounter{#1}%
4648 }
4649 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

4650 \newcommand*{\@glsxtr@docounterrecord}[1]{%
4651 \protected@write\@auxout{}{\string\glsxtr@counterrecord
4652 {\@gls@label}{#1}{\csuse{the#1}}}%
4653 }

```

lsxtrglossentry Users may prefer to have entries displayed throughout the document rather than gathered together in a list. This command emulates the way \glossentry behaves (without the style formatting commands like \item). This needs to define \currentglossary to the current glossary type (normally set at the start of \@printglossary) and needs to define \glscurrententrylabel to the entry's label (normally set before \glossentry and \subglossentry). This needs some protection in case it's used in a section heading.

```

4654 \newcommand*{\glsxtrglossentry}[1]{%

```

```

4655 \glxstrtitleorpdforheading
4656 {\@glxstrglossentry{#1}}%
4657 {\glsenentryname{#1}}%
4658 {\glxstrheadname{#1}}%
4659 }

```

`\glxstrglossentry` Another test is needed in case `\@glxstrglossentry` has been written to the table of contents.

```

4660 \newrobustcmd*{\@glxstrglossentry}[1]{%
4661   \glxstrtitleorpdforheading
4662   {%
4663     \glsdodefexists{#1}%
4664     {%
4665       \begingroup
4666         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4667         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4668         \ifglshasparent{#1}%
4669           {\GlsXtrStandaloneSubEntryItem{#1}}%
4670           {\glsenentryitem{#1}}%
4671           \glstarget{#1}{\glssentryname{#1}}%
4672         \endgroup
4673       }%
4674     }%
4675     {\glsenentryname{#1}}%
4676     {\glxstrheadname{#1}}%
4677 }

```

`\oneGlossaryType` To make it easier to adjust the definition of `\currentglossary` within `\glxstrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4678 \newcommand{\GlsXtrStandaloneGlossaryType}{\glssentrytype{\glscurrententrylabel}}

```

`\oneSubEntryItem` Used for sub-entries in standalone format. The argument is the entry's label.

```

4679 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4680   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
4681 }

```

`\glossentryother` As `\glxstrglossentry` but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```

4682 \newcommand*{\glxstrglossentryother}[3]{%
4683   \ifstrempy{#1}%
4684   {%
4685     \ifcsdef{glxstrhead#3}%
4686     {%
4687       \glxstrtitleorpdforheading
4688       {\@glxstrglossentryother{#2}{#3}{#1}}%

```

```

4689     {\@gls@entry@field{#2}{#3}}%
4690     {\csuse{glsxtrhead#3}{#2}}%
4691 }%
4692 {%
4693     \glsxtrtitleorpdforheading
4694     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4695     {\@gls@entry@field{#2}{#3}}%
4696     {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4697 }%
4698 }%
4699 {%
4700     \glsxtrtitleorpdforheading
4701     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4702     {\@gls@entry@field{#2}{#3}}%
4703     {#1}%
4704 }%
4705 }

```

`glossentryother` As `\@glsxtrglossentry` but uses a different field.

```

4706 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
4707     \glsxtrtitleorpdforheading
4708     {%
4709         \glsdoifexists{#1}%
4710         {%
4711             \begingroup
4712                 \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4713                 \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4714                 \ifglshasparent{#1}%
4715                     {\GlsXtrStandaloneSubEntryItem{#1}}%
4716                     {\glsentryitem{#1}}%
4717                     \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4718                 \endgroup
4719             }%
4720         }%
4721         {\@gls@entry@field{#1}{#2}}%
4722         {#3}%
4723 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4724 \newcommand*{\printunsrtglossary}{%
4725     \@ifstar\s@printunsrtglossary\@printunsrtglossary
4726 }

```

`ntunsrtglossary` Unstarred version.

```

4727 \newcommand*{\@printunsrtglossary}[1][\]{%
4728     \@printglossary{type=\glsdefaultttype,#1}{\@print@unsrt@glossary}%
4729 }

```

ntunsrtglossary Starred version.

```
4730 \newcommand*{\s@printunsrtglossary}[2][\{%
4731   \begingroup
4732   #2%
4733   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4734   \endgroup
4735 }
```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```
4736 \newcommand*{\printunsrtglossaries}{%
4737   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4738 }
```

@unsrt@glossary

```
4739 \newcommand*{\@print@unsrt@glossary}{%
4740   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4741   \glossarypreamble
4742   check for empty list
4743   \glsxtrifemptyglossary{\@glo@type}%
4744   {%
4745     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4746   }%
4747   \key@ifundefined{glossentry}{group}%
4748   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4749   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4750   \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
4751 \def\@glsxtr@doglossary{%
4752   \begin{theglossary}%
4753   \glossaryheader
4754   \glsresetentrylist
4755 }%
4756 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4757   :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
4758   \ifdefempty{\glscurrententrylabel}
4759   }%
4760 }
```

Provide a hook (for example to measure width).

```
4761 \let\glsxtr@process\@firstofone
4762 \let\printunsrtglossaryskipentry
4763 \let\glsxtr@printunsrtglossaryskipentry
4764 \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```

4765 \glstr@process
4766 {%
4767 \ifglshasparent{\glscurrententrylabel}{}%
4768 {%
4769 \@glstr@checkgroup\glscurrententrylabel
4770 \expandafter\appto\expandafter\@glstr@doglossary\expandafter
4771 {\@glstr@groupheading}%
4772 }%
4773 \eappto\@glstr@doglossary{%
4774 \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4775 }%
4776 }%
4777 }%
4778 \appto\@glstr@doglossary{\end{theglossary}}%
4779 \printunsrtglossarypredoglossary
4780 \@glstr@doglossary
4781 }%
4782 \glossarypostamble
4783 }

```

entryprocesshook

```

4784 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

glossaryskipentry

```

4785 \newcommand*{\printunsrtglossaryskipentry}{%
4786 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4787 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4788 }

```

entryprocesshook

```

4789 \newcommand*{\@glstr@printunsrtglossaryskipentry}{%
4790 \let\glstr@process\@gobble
4791 }

```

rypredoglossary

```

4792 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```

4793 \newcommand{\@printunsrt@glossary@handler}[1]{%
4794 \xdef\glscurrententrylabel{#1}%
4795 \printunsrtglossaryhandler\glscurrententrylabel
4796 }

```

glossaryhandler

```

4797 \newcommand{\printunsrtglossaryhandler}[1]{%
4798 \glstrunsrtdo{#1}%
4799 }

```

xtriflabelinlist

```
\glxtriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```
4800 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4801   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
4802   \@glxtr@doiflabelinlist{#3}{#4}}%
4803 }
```

srtglossaryunit

```
4804 \newcommand{\print@op@unsrtglossaryunit}[2][{}]{%
4805   \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
4806     \printunsrtglossaryunitsetup{#2}}%
4807   }%
4808 }
```

ossaryunitsetup

```
4809 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4810   \renewcommand{\printunsrtglossaryhandler}[1]{%
4811     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4812     {\glxtrunsrtdo{##1}}}%
4813   }%
4814   }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glslinkprefix`. The `\@gobble` part discards `\glslinkprefix`.

```
4815   \ifcsundef{theH#1}%
4816     {%
4817       \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4818     }%
4819     {%
4820       \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4821     }%
4822   \renewcommand*{\glossarysection}[2][{}]{%
4823     \appto\glossarypostamble{\glspar\medskip\glspar}%
4824   }
```

srtglossaryunit

```
4825 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
4826   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4827     requires the record=only or record=alsoindex package option}{}%
4828 }
```

t@getgrouptitle

```
4829 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4830   \protected@edef\@glxtr@titlelabel{\glxtr@grouptitle@#1}%
4831 }
```

```

4831 \@onelevel@sanitize\@glxtr@titlelabel
4832 \ifcsdef{\@glxtr@titlelabel}
4833 {\letcs{#2}{\@glxtr@titlelabel}}%
4834 {\def#2{#1}}%
4835 }

```

`\glxtrunsrtdo` Provide a user-level call to `\@glxtr@noidx@do` to make it easier to define a new handler.

```

4836 \newcommand{\glxtrunsrtdo}{\@glxtr@noidx@do}

```

`lsxtrgroupfield` `bib2gls` provides a supplementary field labelled `secondarygroup` for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```

4837 \newcommand*{\glxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while `\@glxtr@doglossary` is being constructed rather than in the handler.

`lsxtr@checkgroup` The argument is the entry's label. (This block of code was formerly in `\@glxtr@noidx@do`.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in `\@glxtr@groupheading`, which will be empty if no heading is required.

```

4838 \newcommand*{\@glxtr@checkgroup}[1]{%
4839   \def\@glxtr@groupheading{}%
4840   \key@ifundefined{glossentry}{group}%
4841   {%
4842     \letcs{\@gls@sort}{glo@glstdetoklabel{#1}@sort}%
4843     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
4844   }%
4845   {%
4846     \protected@edef\@glo@thislettergrp{%
4847       \csuse{glo@glstdetoklabel{#1}@glxtrgroupfield}}%
4848     }%
4849     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4850     {}%
4851     {%
4852       \ifdefempty{\@gls@currentlettergroup}{}%
4853       {\def\@glxtr@groupheading{\glsgroupskip}}%
4854       \eappto\@glxtr@groupheading{%
4855         \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
4856       }%
4857     }%
4858     \let\@gls@currentlettergroup\@glo@thislettergrp
4859 }

```

`glxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```

4860 \newcommand{\@glsxtr@noidx@do}[1]{%
4861   \ifglstryexists{#1}%
4862   {%
4863     \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
4864     \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@location}%
4865     \ifglshasparent{#1}%
4866     {%
4867       \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
4868       \ifdefvoid{\@gls@location}%
4869       {%
4870         \ifdefvoid{\@gls@loclist}%
4871         {%
4872           \subglossentry{\gls@level}{#1}{}%
4873         }%
4874         {%
4875           \subglossentry{\gls@level}{#1}%
4876           {%
4877             \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4878           }%
4879         }%
4880       }%
4881       {%
4882         \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4883       }%
4884     }%
4885   {%
4886     \ifdefvoid{\@gls@location}%
4887     {%
4888       \ifdefvoid{\@gls@loclist}
4889       {%
4890         \glossentry{#1}{}%
4891       }%
4892       {%
4893         \glossentry{#1}%
4894         {%
4895           \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4896         }%
4897       }%
4898     }%
4899     {%
4900       \glossentry{#1}%
4901       {%
4902         \glossaryentrynumbers{\@gls@location}%
4903       }%
4904     }%
4905   }%
4906 }%
4907 {}%
4908 }

```

Provide a way to conveniently define commands that behaves like `\gls` with a label prefix.
 It's possible that the user might want minor variations with the same prefix but different default options, so use a counter to provide unique inner commands.

`\glsxtrnewgls`

4909 `\newcount\@glsxtrnewgls@inner`

(The default options supplied in `\options` below could possibly be used to form the inner control sequence name to help make it unique, but it might feasibly contain the value where the value might contain commands.)

`\@glsxtrnewgls`

`\glsxtrnewgls[\options]{\prefix}{\cs}{\inner cs name}`

```

4910 \newcommand*\@glsxtrnewgls[4]{%
4911   \ifdef{#3}%
4912   {%
4913     \PackageError{glossaries-extra}{Command \string#3\space already
4914 defined}{}%
4915   }%
4916   {%
4917     \ifcsdef{@#4like@#2}%
4918     {%
4919       \advance\@glsxtrnewgls@inner by \@ne
4920       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4921     }%
4922     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4923     \expandafter\newrobustcmd\expandafter*\expandafter
4924       #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4925     \ifstrempy{#1}%
4926     {%
4927       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
4928         \new@ifnextchar[%
4929           {\csname @#4@\endcsname{##1}{#2##2}}%
4930           {\csname @#4@\endcsname{##1}{#2##2} []}%
4931         }%
4932       }%
4933     {%
4934       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
4935         \new@ifnextchar[%
4936           {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4937           {\csname @#4@\endcsname{#1,##1}{#2##2} []}%
4938         }%
4939       }%
4940     }%
4941 }

```

`\glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```
4942 \newrobustcmd*{\glxstrnewgls}[3] [] {%
4943   \@glxstrnewgls{#1}{#2}{#3}{gls}%
4944 }
```

`\glxstrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4945 \newrobustcmd*{\glxstrnewglslike}[6] [] {%
4946   \@glxstrnewgls{#1}{#2}{#3}{gls}%
4947   \@glxstrnewgls{#1}{#2}{#4}{glspl}%
4948   \@glxstrnewgls{#1}{#2}{#5}{Gls}%
4949   \@glxstrnewgls{#1}{#2}{#6}{Glspl}%
4950 }
```

`\glxstrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4951 \newrobustcmd*{\glxstrnewGLSlike}[4] [] {%
4952   \@glxstrnewgls{#1}{#2}{#3}{GLS}%
4953   \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
4954 }
```

`\glxstrnewrgls` As `\glxstrnewgls` but for `\rgls`.

```
4955 \newrobustcmd*{\glxstrnewrgls}[3] [] {%
4956   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4957 }
```

`\glxstrnewrglslike` As `\glxstrnewglslike` but for `\rgls` etc.

```
4958 \newrobustcmd*{\glxstrnewrglslike}[6] [] {%
4959   \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4960   \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
4961   \@glxstrnewgls{#1}{#2}{#5}{rGls}%
4962   \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
4963 }
```

`\glxstrnewrGLSlike` As `\glxstrnewGLSlike` but for `\rGLS` etc.

```
4964 \newrobustcmd*{\glxstrnewrGLSlike}[4] [] {%
4965   \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
4966   \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
4967 }
```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
4968 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
4969 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount}%
4970 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
4971 {0}%
4972 }
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4973 \newcommand*{\GlsXtrRecordCount}[2]{%
4974 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
4975 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
4976 {0}%
4977 }
```

tionRecordCount Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless \glsxtrdetoklocation can be set to something sensible.

```
4978 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4979 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}}%
4980 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}\endcsname}%
4981 {0}%
4982 }
```

trdetoklocation

```
4983 \newcommand*{\glsxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
4984 \newcommand*{\glsxtrenablerecordcount}{%
4985 \renewcommand*{\gls}{\rgls}%
4986 \renewcommand*{\Gls}{\rGls}%
4987 \renewcommand*{\glspl}{\rglspl}%
4988 \renewcommand*{\Glspl}{\rGlspl}%
4989 \renewcommand*{\GLS}{\rGLS}%
4990 \renewcommand*{\GLSpl}{\rGLSpl}%
4991 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
4992 \newcommand*{\glsxtrrecordtriggervalue}[1]{%
4993 \GlsXtrTotalRecordCount{#1}%
4994 }
```

dCountAttribute

```
4995 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4996 \@for\@glsxtr@cat:=#1\do
4997 {%
4998 \ifdefempty{\@glsxtr@cat}{}%
4999 }
```

```

4999   {%
5000     \glssetcategoryattribute{\@glxstr@cat}{recordcount}{#2}%
5001   }%
5002 }%
5003 }

```

`trifrecordtrigger` `\glxstrifrecordtrigger{<label>}{<trigger format>}{<normal>}`

```

5004 \newcommand*{\glxstrifrecordtrigger}[3]{%
5005   \glshasattribute{#1}{recordcount}%
5006   {%
5007     \ifnum\glxstrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
5008     #3%
5009   \else
5010     #2%
5011   \fi
5012 }%
5013 {#3}%
5014 }

```

`strigger@record` Still need a record to ensure that bib2gls selects the entry.

```

5015 \newcommand*{\@glxstr@rglstrigger@record}[3]{%
5016   \edef\glslabel{\glsdetoklabel{#2}}%
5017   \let\@gls@link@label\glslabel
5018   \def\@glxstr@thevalue{%
5019     \def\@glxstr@theHvalue{\@glxstr@thevalue}%
5020     \def\@glsnumberformat{glstriggerrecordformat}%
5021     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5022     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5023     \def\@glxstr@thevalue{%
5024       \def\@glxstr@theHvalue{\@glxstr@thevalue}%
5025       \glxstrinitwrgloss
5026       \glslinkpresetkeys
5027       \setkeys{glslink}{#1}%
5028       \glslinkpostsetkeys
5029       \ifdefempty{\@glxstr@thevalue}%
5030       {%
5031         \@gls@saveentrycounter
5032       }%
5033     }%
5034     \let\thegl Sentrycounter\@glxstr@thevalue
5035     \def\theHglSentrycounter{\@glxstr@theHvalue}%
5036   }%
5037   \ifglxstrinitwrglossbefore
5038     \do@wrglossary{#2}%
5039   \fi

```

```

5040 #3%
5041 \ifglxtrinitwrglossbefore
5042 \else
5043   \@do@wrglossary{#2}%
5044 \fi
5045 \ifKV@glslink@local
5046   \glsllocalunset{#2}%
5047 \else
5048   \glsunset{#2}%
5049 \fi
5050 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5051 \newcommand*{\glstriggerrecordformat}[1]{%

```

\rgls

```

5052 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

5053 \newcommand*{\@rgls}[2][{}]{%
5054   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5055 }

```

\@rgls@

```

5056 \def\@rgls@#1#2[#3]{%
5057   \glxtrifrecordtrigger{#2}%
5058   {%
5059     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5060   }%
5061   {%
5062     \@gls@{#1}{#2}[#3]%
5063   }%
5064 }%

```

\rglsp1

```

5065 \newrobustcmd*{\rglsp1}{\@gls@hyp@opt\@rglsp1}

```

\@rglsp1

```

5066 \newcommand*{\@rglsp1}[2][{}]{%
5067   \new@ifnextchar[{\@rglsp1@{#1}{#2}}{\@rglsp1@{#1}{#2}[]}%
5068 }

```

\@rglsp1@

```

5069 \def\@rglsp1@#1#2[#3]{%
5070   \glxtrifrecordtrigger{#2}%
5071   {%
5072     \@glxtr@rglstrigger@record{#1}{#2}{\rglsp1format{#2}{#3}}%

```

```

5073 }%
5074 {%
5075 \@glspl@{#1}{#2}[#3]%
5076 }%
5077 }%

```

\rGls

```
5078 \newrobustcmd*{\rGls}{\@gls@hyp@opt\rGls}
```

\@rGls

```

5079 \newcommand*{\@rGls}[2][{}]{%
5080 \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5081 }

```

\@rGls@

```

5082 \def\rGls@#1#2[#3]{%
5083 \glsxtrifrecordtrigger{#2}%
5084 {%
5085 \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5086 }%
5087 {%
5088 \@Gls@{#1}{#2}[#3]%
5089 }%
5090 }%

```

\rGlspl

```
5091 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}
```

\@rGlspl

```

5092 \newcommand*{\@rGlspl}[2][{}]{%
5093 \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5094 }

```

\@rGlspl@

```

5095 \def\rGlspl@#1#2[#3]{%
5096 \glsxtrifrecordtrigger{#2}%
5097 {%
5098 \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5099 }%
5100 {%
5101 \@Glspl@{#1}{#2}[#3]%
5102 }%
5103 }%

```

\rGLS

```
5104 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}
```

\@rGLS

```
5105 \newcommand*{\@rGLS}[2][\%  
5106   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}}]{\%  
5107 }
```

\@rGLS@

```
5108 \def\@rGLS@#1#2[#3]{\%  
5109   \glstrifrecordtrigger{#2}%  
5110   {\%  
5111     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%  
5112   }%  
5113   {\%  
5114     \@GLS@{#1}{#2}[#3}%  
5115   }%  
5116 }%
```

\rGLSpl

```
5117 \newrobustcmd*{\rGLSpl}{\@glshyp@opt\rGLSpl}
```

\@rGLSpl

```
5118 \newcommand*{\@rGLSpl}[2][\%  
5119   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}}]{\%  
5120 }
```

\@rGLSpl@

```
5121 \def\@rGLSpl@#1#2[#3]{\%  
5122   \glstrifrecordtrigger{#2}%  
5123   {\%  
5124     \@glstr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%  
5125   }%  
5126   {\%  
5127     \@GLSpl@{#1}{#2}[#3}%  
5128   }%  
5129 }%
```

\rglsformat

```
5130 \newcommand*{\rglsformat}[2]{\%  
5131   \glrifregular{#1}  
5132   {\glentryfirst{#1}}%  
5133   {\ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}}%#2%  
5134 }
```

\rglsplformat

```
5135 \newcommand*{\rglsplformat}[2]{\%  
5136   \glrifregular{#1}  
5137   {\glentryfirstplural{#1}}%  
5138   {\ifglshaslong{#1}{\glentrylongplural{#1}}{\glentryfirstplural{#1}}}%#2%  
5139 }
```

`\rGlsformat`

```
5140 \newcommand*{\rGlsformat}[2]{%
5141   \glsifregular{#1}
5142   {\Glsentryfirst{#1}}%
5143   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5144 }
```

`\rGlsplformat`

```
5145 \newcommand*{\rGlsplformat}[2]{%
5146   \glsifregular{#1}
5147   {\Glsentryfirstplural{#1}}%
5148   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5149 }
```

`\rGLSformat`

```
5150 \newcommand*{\rGLSformat}[2]{%
5151   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsformat{#1}{#2}}%
5152 }
```

`\rGLSplformat`

```
5153 \newcommand*{\rGLSplformat}[2]{%
5154   \expandafter\mfirstucMakeUppercase\expandafter{\rGlsplformat{#1}{#2}}%
5155 }
```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into `\@gls@link` (through `\glsxtr@inc@linkcount`) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The “link” part of the name refers to `\@gls@link` not `\hyperlink`.)

`\@inc@linkcount` This performs the actual incrementing and counter definition. The counter is given by `\c@glsxtr@linkcount@<label>` where *<label>* is the entry’s label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```
5156 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
5157   \glsifattribute{\glslabel}{linkcount}{true}%
5158   {%
    Does the counter exist?
5159     \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5160     {%
      Counter doesn't exist, so define it.
5161       \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
5162 \glshasattribute{\glslabel}{linkcountmaster}%
5163 {%
```

Need to ensure values are fully expanded.

```
5164 \begingroup
5165 \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@glslabel}%
5166 {\glsggetattribute{glslabel}{linkcountmaster}}}%
5167 \x
5168 }%
5169 {}%
5170 }%
```

Increment counter:

```
5171 \glxtrinlinkcounter{glxtr@linkcount@glslabel}%
5172 }%
5173 {}%
5174 }
```

`\reflinkcounter` May be redefined to use `\refstepcounter` if required.

```
5175 \newcommand*{\glxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
5176 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5177 \ifcsundef{c@glxtr@linkcount@#1}{0}{\csname c@glxtr@linkcount@#1\endcsname}%
5178 }
```

`\rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5179 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5180 \ifcsundef{theglxtr@linkcount@#1}{0}%
5181 {\csname theglxtr@linkcount@#1\endcsname}%
5182 }
```

`\ifLinkCounterDef` Tests if the counter has been defined

```
5183 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5184 \ifcsundef{theglxtr@linkcount@#1}{#3}{#2}%
5185 }
```

`\LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
5186 \newcommand*{\GlsXtrLinkCounterName}[1]{glxtr@linkcount@#1}
```

`\enableLinkCounting` `\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
5187 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5188 \let\glxtr@inc@linkcount\@glxtr@do@inc@linkcount
```

```

5189 \@for\@glstr@label:=#2\do
5190 {%
5191   \glsetcategoryattribute{\@glstr@label}{linkcount}{true}%
5192   \ifstrepty{#1}{}%
5193   {%
5194     \ifcsundef{c@#1}%
5195     {\@nocounterr{#1}}%
5196     {\glsetcategoryattribute{\@glstr@label}{linkcountmaster}{#1}}%
5197   }%
5198 }%
5199 }
5200 \@onlypreamble\GlsXtrEnableLinkCounting

```

1.5 Integration with glossaries-accsupp

Provide better integration with the glossaries-accsupp package. (Must be loaded before the main code of glossaries-extra either explicitly or through the accsupp package option.)

These commands have their definitions set according to whether or not glossaries-extra has been loaded.

```

5201 \@ifpackageloaded{glossaries-accsupp}
5202 {

```

Define (or redefine) commands to use the accessibility information.

\glsaccessname Display the name value (no link and no check for existence).

```

5203   \newcommand*\glsaccessname[1]{%
5204     \glsnameaccessdisplay
5205     {%
5206       \glstentryname{#1}%
5207     }%
5208     {#1}%
5209   }

```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

5210   \newcommand*\Glsaccessname[1]{%
5211     \glsnameaccessdisplay
5212     {%
5213       \Glstentryname{#1}%
5214     }%
5215     {#1}%
5216   }

```

\GLSaccessname Display the name value (no link and no check for existence) converted to upper case.

```

5217   \newcommand*\GLSaccessname[1]{%
5218     \glsnameaccessdisplay
5219     {%
5220       \mfirstucMakeUppercase{\glstentryname{#1}}%

```

```

5221     }%
5222     {#1}%
5223 }

```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

5224 \newcommand*{\glsaccesstext}[1]{%
5225   \glstextaccessdisplay
5226   {%
5227     \glentrytext{#1}%
5228   }%
5229   {#1}%
5230 }

```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

5231 \newcommand*{\Glsaccesstext}[1]{%
5232   \glstextaccessdisplay
5233   {%
5234     \Glsentrytext{#1}%
5235   }%
5236   {#1}%
5237 }

```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5238 \newcommand*{\GLSaccesstext}[1]{%
5239   \glstextaccessdisplay
5240   {%
5241     \mfirstucMakeUppercase{\glentrytext{#1}}%
5242   }%
5243   {#1}%
5244 }

```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

5245 \newcommand*{\glsaccessplural}[1]{%
5246   \glspluralaccessdisplay
5247   {%
5248     \glentryplural{#1}%
5249   }%
5250   {#1}%
5251 }

```

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5252 \newcommand*{\Glsaccessplural}[1]{%
5253   \glspluralaccessdisplay
5254   {%
5255     \Glsentryplural{#1}%
5256   }%

```

```

5257     {#1}%
5258 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

5259 \newcommand*{\GLSaccessplural}[1]{%
5260   \glspluralaccessdisplay
5261   {%
5262     \mfirstucMakeUppercase{\glsentryplural{#1}}%
5263   }%
5264   {#1}%
5265 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

5266 \newcommand*{\glsaccessfirst}[1]{%
5267   \glsfirstaccessdisplay
5268   {%
5269     \glsentryfirst{#1}%
5270   }%
5271   {#1}%
5272 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5273 \newcommand*{\Glsaccessfirst}[1]{%
5274   \glsfirstaccessdisplay
5275   {%
5276     \Glsentryfirst{#1}%
5277   }%
5278   {#1}%
5279 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

5280 \newcommand*{\GLSaccessfirst}[1]{%
5281   \glsfirstaccessdisplay
5282   {%
5283     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
5284   }%
5285   {#1}%
5286 }

```

cessfirstplural Display the firstplural value (no link and no check for existence).

```

5287 \newcommand*{\glsaccessfirstplural}[1]{%
5288   \glsfirstpluralaccessdisplay
5289   {%
5290     \glsentryfirstplural{#1}%
5291   }%
5292   {#1}%
5293 }

```

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5294 \newcommand*\Glsaccessfirstplural}[1]{%
5295 \glsfirstpluralaccessdisplay
5296 {%
5297 \Glsentryfirstplural{#1}%
5298 }%
5299 {#1}%
5300 }

```

cessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```

5301 \newcommand*\GLSaccessfirstplural}[1]{%
5302 \glsfirstpluralaccessdisplay
5303 {%
5304 \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5305 }%
5306 {#1}%
5307 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

5308 \newcommand*\glsaccesssymbol}[1]{%
5309 \glssymbolaccessdisplay
5310 {%
5311 \glsentrysymbol{#1}%
5312 }%
5313 {#1}%
5314 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5315 \newcommand*\GLSaccesssymbol}[1]{%
5316 \glssymbolaccessdisplay
5317 {%
5318 \Glsentrysymbol{#1}%
5319 }%
5320 {#1}%
5321 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5322 \newcommand*\GLSaccesssymbol}[1]{%
5323 \glssymbolaccessdisplay
5324 {%
5325 \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5326 }%
5327 {#1}%
5328 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5329 \newcommand*{\glsaccesssymbolplural}[1]{%
5330   \glssymbolpluralaccessdisplay
5331   {%
5332     \glsentrysymbolplural{#1}%
5333   }%
5334   {#1}%
5335 }

```

`\esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5336 \newcommand*{\Glsaccesssymbolplural}[1]{%
5337   \glssymbolpluralaccessdisplay
5338   {%
5339     \Glsentrysymbolplural{#1}%
5340   }%
5341   {#1}%
5342 }

```

`\esssymbolplural` Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5343 \newcommand*{\GLSaccesssymbolplural}[1]{%
5344   \glssymbolpluralaccessdisplay
5345   {%
5346     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5347   }%
5348   {#1}%
5349 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5350 \newcommand*{\glsaccessdesc}[1]{%
5351   \glsdescriptionaccessdisplay
5352   {%
5353     \glsentrydesc{#1}%
5354   }%
5355   {#1}%
5356 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5357 \newcommand*{\Glsaccessdesc}[1]{%
5358   \glsdescriptionaccessdisplay
5359   {%
5360     \Glsentrydesc{#1}%
5361   }%
5362   {#1}%
5363 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5364 \newcommand*{\GLSaccessdesc}[1]{%

```

```

5365 \glsdescriptionaccessdisplay
5366 {%
5367     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5368 }%
5369 {#1}%
5370 }

```

`\glsaccessdescplural` Display the descplural value (no link and no check for existence).

```

5371 \newcommand*{\glsaccessdescplural}[1]{%
5372     \glsdescriptionpluralaccessdisplay
5373     {%
5374         \glsentrydescplural{#1}%
5375     }%
5376     {#1}%
5377 }

```

`\Glsaccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5378 \newcommand*{\Glsaccessdescplural}[1]{%
5379     \glsdescriptionpluralaccessdisplay
5380     {%
5381         \Glsentrydescplural{#1}%
5382     }%
5383     {#1}%
5384 }

```

`\glsaccessdescplurl` Display the descplurl value (no link and no check for existence) converted to upper case.

```

5385 \newcommand*{\GLSaccessdescplurl}[1]{%
5386     \glsdescriptionpluralaccessdisplay
5387     {%
5388         \mfirstucMakeUppercase{\glsentrydescplurl{#1}}%
5389     }%
5390     {#1}%
5391 }

```

`\glsaccesssshort` Display the short form (no link and no check for existence).

```

5392 \newcommand*{\glsaccesssshort}[1]{%
5393     \glsshortaccessdisplay
5394     {%
5395         \glsentryshort{#1}%
5396     }%
5397     {#1}%
5398 }

```

`\Glsaccesssshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5399 \newcommand*{\Glsaccesssshort}[1]{%
5400     \glsshortaccessdisplay

```

```

5401     {%
5402         \Glsentryshort{#1}%
5403     }%
5404     {#1}%
5405 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5406 \newcommand*{\GLSaccessshort}[1]{%
5407     \glsshortaccessdisplay
5408     {%
5409         \mfirstucMakeUppercase{\Glsentryshort{#1}}%
5410     }%
5411     {#1}%
5412 }

```

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5413 \newcommand*{\lsaccessshortpl}[1]{%
5414     \glsshortpluralaccessdisplay
5415     {%
5416         \Glsentryshortpl{#1}%
5417     }%
5418     {#1}%
5419 }

```

`lSaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5420 \newcommand*{\lSaccessshortpl}[1]{%
5421     \glsshortpluralaccessdisplay
5422     {%
5423         \Glsentryshortpl{#1}%
5424     }%
5425     {#1}%
5426 }

```

`LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5427 \newcommand*{\LSaccessshortpl}[1]{%
5428     \glsshortpluralaccessdisplay
5429     {%
5430         \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5431     }%
5432     {#1}%
5433 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

5434 \newcommand*{\glsaccesslong}[1]{%
5435     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5436 }

```

```

\GLsaccesslong   Display the long form (no link and no check for existence).
5437
5438   \newcommand*\GLsaccesslong}[1]{%
5439     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5440   }

\GLSaccesslong   Display the long value (no link and no check for existence) converted to upper case.
5441   \newcommand*\GLSaccesslong}[1]{%
5442     \glslongaccessdisplay
5443     {%
5444       \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5445     }%
5446     {#1}%
5447   }

glsaccesslongpl   Display the long plural form (no link and no check for existence).
5448   \newcommand*\glsaccesslongpl}[1]{%
5449     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5450   }

Glsaccesslongpl   Display the long plural form (no link and no check for existence).
5451
5452   \newcommand*\Glsaccesslongpl}[1]{%
5453     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5454   }

GLSaccesslongpl   Display the longplural value (no link and no check for existence) converted to upper case.
5455   \newcommand*\GLSaccesslongpl}[1]{%
5456     \glslongpluralaccessdisplay
5457     {%
5458       \mfirstucMakeUppercase{\Glsentrylongpl{#1}}%
5459     }%
5460     {#1}%
5461   }

    Keys for accessibility support.
5462   \define@key{glsxtrabrv}{access}{%
5463     \def\@gls@nameaccess{#1}%
5464   }

5465   \define@key{glsxtrabrv}{textaccess}{%
5466     \def\@gls@textaccess{#1}%
5467   }

5468   \define@key{glsxtrabrv}{firstaccess}{%
5469     \def\@gls@firstaccess{#1}%
5470   }

5471   \define@key{glsxtrabrv}{shortaccess}{%
5472     \def\@gls@shortaccess{#1}%
5473   }

```

```

5474 \define@key{glsextrabrv}{shortpluralaccess}{%
5475   \def\@gls@shortaccesspl{#1}%
5476 }

```

@initaccesskeys

```

5477 \newcommand*\@gls@initaccesskeys{%
5478   \def\@gls@nameaccess{%
5479     \def\@gls@textaccess{%
5480       \def\@gls@firstaccess{%
5481         \def\@gls@shortaccess{%
5482           \def\@gls@shortaccesspl{}%
5483         }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```

5484 \newcommand*\@gls@ifaccessattribute@set}[3]{%
5485   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5486   {#2}%
5487   {%
5488     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5489     {#3}%
5490     {%
5491       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5492       {#2}%
5493       {#3}%
5494     }%
5495   }%
5496 }

```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5497 \newcommand*\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```

5498   \ifdefempty\@gls@shortaccess
5499   {%
5500     \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5501     {%
5502       \@glsextr@insertdots\@gls@shortaccess{#1}%
5503       \eappto\ExtraCustomAbbreviationFields{%
5504         shortaccess={\expandonce\@gls@shortaccess},}%
5505       }%
5506     }%
5507   }%
5508 }%

```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```

5509 \ifdefempty\@gls@shortaccess
5510 {}%
5511 {%
5512 \ifdefempty\@gls@shortaccesspl
5513 {%
5514 \@gls@ifaccessattribute@set{aposplural}%
5515 {%
5516 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5517 \@gls@shortaccess'\abbrvpluralsuffix}%
5518 }%
5519 {%
5520 \@gls@ifaccessattribute@set{noshortplural}%
5521 {%
5522 \let\@gls@shortaccesspl\@gls@shortaccess
5523 }%
5524 {%
5525 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5526 \@gls@shortaccess\abbrvpluralsuffix}%
5527 }%
5528 }%
5529 \eappto\ExtraCustomAbbreviationFields{%
5530 shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5531 }%
5532 {}%
5533 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5534 \ifdefempty\@gls@nameaccess
5535 {%
5536 \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5537 {%

```

Do nothing if the shortaccess key hasn't been set.

```

5538 \ifdefempty\@gls@shortaccess
5539 {}%
5540 {%
5541 \eappto\ExtraCustomAbbreviationFields{%
5542 access={\expandonce\@gls@shortaccess},%
5543 }%
5544 }%
5545 }%
5546 {}%
5547 }%
5548 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5549 \ifdefempty\@gls@textaccess
5550 {%
5551 \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5552 {%

```

Do nothing if the shortaccess key hasn't been set.

```

5553     \ifdefempty\@gls@shortaccess
5554     {}%
5555     {%
5556         \eappto\ExtraCustomAbbreviationFields{%
5557             textaccess={\expandonce\@gls@shortaccess},%
5558         }%
5559     }%
5560 }%
5561 {}%
5562 }%
5563 {}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```

5564     \ifdefempty\@gls@firstaccess
5565     {%
5566         \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5567         {%
```

Do nothing if the shortaccess key hasn't been set.

```

5568         \ifdefempty\@gls@shortaccess
5569         {}%
5570         {%
5571             \eappto\ExtraCustomAbbreviationFields{%
5572                 firstaccess={\expandonce\@gls@shortaccess},%
5573             }%
5574         }%
5575     }%
5576     {}%
5577 }%
5578 {}%
5579 }
```

End of if accsupp part

```

5580 }
5581 {
```

No accessibility support. Just define these commands to do \glsentry<xxx>

\glsaccessname Display the name value (no link and no check for existence).

```

5582 \newcommand*{\glsaccessname}[1]{\glsentryname{#1}}
```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

5583 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```

5584 \newcommand*{\GLSaccessname}[1]{%
5585     \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

`\glsaccesstext` Display the text value (no link and no check for existence).
5586 `\newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}`

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.
5587 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.
5588 `\newcommand*{\GLSaccesstext}[1]{%`
5589 `\protect\mfirstucMakeUppercase{\glsentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).
5590 `\newcommand*{\glsaccessplural}[1]{\glsentryplural{#1}}`

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.
5591 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
5592 `\newcommand*{\GLSaccessplural}[1]{%`
5593 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
5594 `\newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}`

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.
5595 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
5596 `\newcommand*{\GLSaccessfirst}[1]{%`
5597 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
5598 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`Glscessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5599 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`GLScessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5600 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5601 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).
5602 `\newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}`

`\Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

5603 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`\GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

5604 `\newcommand*{\GLSaccesssymbol}[1]{%`
5605 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`\essymbolplural` Display the symbolplural value (no link and no check for existence).

5606 `\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}`

`\essymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

5607 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`\essymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.

5608 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5609 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).

5610 `\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

5611 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.

5612 `\newcommand*{\GLSaccessdesc}[1]{%`
5613 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence).

5614 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

5615 `\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`\ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.

5616 `\newcommand*{\GLSaccessdescplural}[1]{%`
5617 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).

5618 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

5619 `\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.

5620 `\newcommand*{\GLSaccessshort}[1]{%`
5621 `\protect\mfirstucMakeUppercase{\glsentryshort{#1}}}`

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

5622 `\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}`

`lSaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

5623 `\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.

5624 `\newcommand*{\GLSaccessshortpl}[1]{%`
5625 `\protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).

5626 `\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}`

`\Glsaccesslong` Display the long form (no link and no check for existence).

5627 `\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

5628 `\newcommand*{\GLSaccesslong}[1]{%`
5629 `\protect\mfirstucMakeUppercase{\glsentrylong{#1}}}`

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

5630 `\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}`

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

5631 `\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}`

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

5632 `\newcommand*{\GLSaccesslongpl}[1]{%`
5633 `\protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}`

`@initaccesskeys` This does nothing if there's no accessibility support.

5634 `\newcommand*{\@gls@initaccesskeys}{}`

`lt@short@access` This does nothing if there's no accessibility support.

5635 `\newcommand{\@gls@setup@default@short@access}[1]{}%`
End of else part
5636 }

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
5637 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
5638 \newcommand{\glsifcategory}[4]{%
5639   \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
5640 }
```

Categories can have attributes.

`categoryattribute`

```
\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
5641 \newcommand*{\glssetcategoryattribute}[3]{%
5642   \csdef{@glxtr@categoryattr@@#1@#2}{#3}%
5643 }
```

`categoryattribute`

```
\glsgetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5644 \newcommand*{\glsgetcategoryattribute}[2]{%
5645   \csuse{@glxtr@categoryattr@@#1@#2}%
5646 }
```

`categoryattribute`

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5647 \newcommand*{\glshascategoryattribute}[4]{%
5648   \ifcvoid{@glxtr@categoryattr@@#1@#2}{#4}{#3}%
5649 }
```

`\glssetattribute`

```
\glssetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5650 \newcommand*{\glssetattribute}[3]{%
```

```

5651 \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5652 }

```

`\glsggetattribute` `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```

5653 \newcommand*{\glsggetattribute}[2]{%
5654 \glsssetcategoryattribute{\glscategory{#1}}{#2}%
5655 }

```

`\glshasattribute` `\glshasattribute{<entry label>}{<attribute-label>}{<true>}{<false>}`

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```

5656 \newcommand*{\glshasattribute}[4]{%
5657 \ifglssentryexists{#1}%
5658 {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5659 {#4}%
5660 }

```

`\glssifcategoryattribute` `\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```

5661 \newcommand{\glssifcategoryattribute}[5]{%
5662 \ifcsundef{@glssxtr@categoryattr@#1@#2}%
5663 {#5}%
5664 {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5665 }

```

`\glssifattribute` `\glssifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

5666 \newcommand{\glssifattribute}[5]{%
5667 \ifglssentryexists{#1}%
5668 {\glssifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5669 {#5}%
5670 }

```

Set attributes for the default general category:

```
5671 \glsssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5672 \glsssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
5673 \newcommand*{\glsssetregularcategory}[1]{%
5674   \glsssetcategoryattribute{#1}{regular}{true}%
5675 }
```

`ifregularcategory` `\glssifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5676 \newcommand{\glssifregularcategory}[3]{%
5677   \glssifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5678 }
```

`ifnotregularcategory` `\glssifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5679 \newcommand{\glssifnotregularcategory}[3]{%
5680   \glssifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5681 }
```

`\glssifregular` `\glssifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5682 \newcommand{\glssifregular}[3]{%
5683   \glssifregularcategory{\glscategory{#1}}{#2}{#3}%
5684 }
```

`\glssifnotregular` `\glssifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5685 \newcommand{\glssifnotregular}[3]{%
5686   \glssifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5687 }
```

oreachincategory

```
\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}
{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category matches *⟨category-label⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
5688 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5689   \forallglossaries[#1]{#3}%
5690   {%
5691     \forlgsentries[#3]{#4}%
5692     {%
5693       \glsifcategory{#4}{#2}{#5}{}%
5694     }%
5695   }%
5696 }
```

achwithattribute

```
\glsforeachwithattribute[⟨glossary labels⟩]{⟨attribute-label⟩}
{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category attribute *⟨attribute-label⟩* matches *⟨attribute-value⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
5697 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5698   \forallglossaries[#1]{#4}%
5699   {%
5700     \forlgsentries[#4]{#5}%
5701     {%
5702       \glsifattribute{#5}{#2}{#3}{#6}{}%
5703     }%
5704   }%
5705 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```
5706 \ifdef\newterm
5707 {%
```

`\newterm`

```
5708   \renewcommand*{\newterm}[2][ ]{%
5709     \newglossaryentry{#2}%
5710     {type={index},category=index,name={#2},%
```

```

5711     description={\glxtrpostdescription\nopostdesc},#1}%
5712 }

```

Indexed terms are regular by default.

```

5713 \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5714 \newcommand*{\glxtrpostdescindex}{}

5715 }
5716 {}

```

If the symbols package option was used, define a similar command for symbols, but set the default sort to the label rather than the name as the symbols will typically contain commands that will confuse makeindex and xindy.

```

5717 \ifdef\printsymbols
5718 {%

```

glxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```

5719 \newcommand*{\glxtrnewsymbol}[3] [] {%
5720 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5721 }

```

Symbols are regular by default.

```

5722 \glsssetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

5723 \newcommand*{\glxtrpostdescsymbol}{}

5724 }
5725 {}

```

Similar for the numbers option.

```

5726 \ifdef\printnumbers
5727 {%

```

glxtrnewnumber

```

5728 \ifdef\printnumbers
5729 \newcommand*{\glxtrnewnumber}[3] [] {%
5730 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5731 }

```

Numbers are regular by default.

```

5732 \glsssetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

5733 \newcommand*{\glxtrpostdescnumber}{}

```

```
5734 }
5735 {}
```

`\glstrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5736 \newcommand*{\glstrsetcategory}[2]{%
5737   \@for\@glstr@label:=#1\do
5738   {%
5739     \glsfieldxdef{\@glstr@label}{category}{#2}%
5740   }%
5741 }
```

`\glstrcategoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
5742 \newcommand*{\glstrcategoryforall}[2]{%
5743   \forallglossaries[#1]{\@glstr@type}{%
5744     \forallsentries[\@glstr@type]{\@glstr@label}%
5745     {%
5746       \glsfieldxdef{\@glstr@label}{category}{#2}%
5747     }%
5748   }%
5749 }
```

`\glstrfieldtitlecase` `\glstrfieldtitlecase{\langle label \rangle}{\langle field \rangle}`

Apply title casing to the contents of the given field.

```
5750 \newcommand*{\glstrfieldtitlecase}[2]{%
5751   \expandafter\glstrfieldtitlecasesecs\expandafter
5752   {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
5753 }
```

`\glstrfieldtitlecasesecs` The command used by `\glstrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
5754 \newcommand*{\glstrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
5755 \@ifpackageloaded{glossaries-accsupp}
5756 {
5757   \renewcommand*{\glossentrydesc}[1]{%
5758     \glsdoifexistsorwarn{#1}%
5759     {%
5760       \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

5761 \glshasattribute{#1}{glossdescfont}%
5762 {%
5763 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5764 \ifcsdef{\@glxtr@attrval}%
5765 {%
5766 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5767 }%
5768 {%
5769 \GlossariesExtraWarning{Unknown control sequence name
5770 '\@glxtr@attrval' supplied in glossdescfont attribute
5771 for entry '#1'. Ignoring}%
5772 \let\@glxtr@glossdescfont\@firstofone
5773 }%
5774 }%
5775 {\let\@glxtr@glossdescfont\@firstofone}%
5776 \glusifattribute{#1}{glossdesc}{firstuc}%
5777 {%
5778 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5779 }%
5780 {%
5781 \glusifattribute{#1}{glossdesc}{title}%
5782 {%
5783 \@glxtr@do@titlecaps@warn
5784 \glsdescriptionaccessdisplay
5785 {%
5786 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5787 }%
5788 {#1}%
5789 }%
5790 {%
5791 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5792 }%
5793 }%
5794 }%
5795 }
5796 }
5797 {
5798 \renewcommand*{\glossentrydesc}[1]{%
5799 \glsdoifexistsorwarn{#1}%
5800 {%
5801 \glissetabbrvfmt{\glscategory{#1}}%
5802 \glshasattribute{#1}{glossdescfont}%
5803 {%
5804 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5805 \ifcsdef{\@glxtr@attrval}%
5806 {%
5807 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5808 }%

```

```

5809      {%
5810      \GlossariesExtraWarning{Unknown control sequence name
5811      '\@glsxtr@attrval' supplied in glossdescfont attribute
5812      for entry '#1'. Ignoring}%
5813      \let\@glsxtr@glossdescfont\@firstofone
5814      }%
5815  }%
5816  {\let\@glsxtr@glossdescfont\@firstofone}%
5817  \glsifattribute{#1}{glossdesc}{firstuc}%
5818  {%
5819  \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5820  }%
5821  {%
5822  \glsifattribute{#1}{glossdesc}{title}%
5823  {%
5824  \@glsxtr@do@titlecaps@warn
5825  \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5826  }%
5827  {%
5828  \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5829  }%
5830  }%
5831  }%
5832  }
5833  }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5834 \ifpackageloaded{glossaries-accsupp}
5835 {
5836   \renewcommand*{\glossentryname}[1]{%
5837     \@glsdoifexistsorwarn{#1}%
5838     {%
5839       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5840   \glshasattribute{#1}{glossnamefont}%
5841   {%
5842   \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5843   \ifcsdef{\@glsxtr@attrval}%
5844   {%
5845   \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5846   }%
5847   {%
5848   \GlossariesExtraWarning{Unknown control sequence name
5849   '\@glsxtr@attrval' supplied in glossnamefont attribute
5850   for entry '#1'. Reverting to default \string\glsnamefont}%
5851   \let\@glsxtr@glossnamefont\glsnamefont
5852   }%
5853   }%

```

```

5854      {\let\@glsxtr@glossnamefont\glsnamefont}%
5855      \glsifattribute{#1}{glossname}{firstuc}%
5856      {%
5857        \glsnameaccessdisplay
5858        {%
5859          \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5860        }%
5861        {#1}%
5862      }%
5863      {%
5864        \glsifattribute{#1}{glossname}{title}%
5865        {%
5866          \@glsxtr@do@titlecaps@warn
5867          \glsnameaccessdisplay
5868          {%
5869            \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
5870          }%
5871          {#1}%
5872        }%
5873        {%
5874          \glsifattribute{#1}{glossname}{uc}%
5875          {%
5876            \glsnameaccessdisplay
5877            {%

```

Hide the label from the upper-casing command.

```

5878          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5879          \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
5880        }%
5881        {#1}%
5882      }%
5883      {%
5884        \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5885        \glsnameaccessdisplay
5886        {%
5887          \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
5888        }%
5889        {#1}%
5890      }%
5891    }%
5892  }%

```

Do post-name hook:

```

5893      \glsxtrpostnamehook{#1}%
5894    }%
5895  }
5896 }
5897 {
5898   \renewcommand*{\glossentryname}[1]{%
5899     \@glsdoifexistsorwarn{#1}%

```

```

5900   {%
5901     \glsetabbrvfmt{\glscategory{#1}}%
5902     \glshasattribute{#1}{glossnamefont}%
5903     {%
5904       \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5905       \ifcsdef{\@glxtr@attrval}%
5906       {%
5907         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5908       }%
5909       {%
5910         \GlossariesExtraWarning{Unknown control sequence name
5911           '\@glxtr@attrval' supplied in glossnamefont attribute
5912           for entry '#1'. Reverting to default \string\glsglossnamefont}%
5913         \let\@glxtr@glossnamefont\glsglossnamefont
5914       }%
5915     }%
5916     {\let\@glxtr@glossnamefont\glsglossnamefont}%
5917     \glsgifattribute{#1}{glossname}{firstuc}%
5918     {%
5919       \@glxtr@glossnamefont{\Glsentryname{#1}}%
5920     }%
5921     {%
5922       \glsgifattribute{#1}{glossname}{title}%
5923     }%
5924     \@glxtr@do@titlecaps@warn
5925     \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5926   }%
5927   {%
5928     \glsgifattribute{#1}{glossname}{uc}%
5929   }%

```

Hide the label from the upper-casing command.

```

5930       \letcs{\glo@name}{\glo@glsgdetoklabel{#1}@name}%
5931       \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5932     }%
5933     {%

```

This little trick is used by glossaries to allow the user to redefine \glsglossnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```

5934       \letcs{\glo@name}{\glo@glsgdetoklabel{#1}@name}%
5935       \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5936     }%
5937   }%
5938 }%

```

Do post-name hook.

```

5939   \glxtrpostnamehook{#1}%
5940 }%
5941 }
5942 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
5943 \ifpackageloaded{glossaries-accsupp}
5944 {
5945   \renewcommand*{\Glossentryname}[1]{%
5946     \@glsdoifexistsorwarn{#1}%
5947     {%
5948       \glsetabbrvfmt{\glscategory{#1}}%
5949       \glshasattribute{#1}{glossnamefont}%
5950       {%
5951         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5952         \ifcsdef{\@glstr@attrval}%
5953         {%
5954           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5955         }%
5956         {%
5957           \GlossariesExtraWarning{Unknown control sequence name
5958             '\@glstr@attrval' supplied in glossnamefont attribute
5959             for entry '#1'. Reverting to default \string\glnamefont}%
5960           \let\@glstr@glossnamefont\glnamefont
5961         }%
5962       }%
5963       {\let\@glstr@glossnamefont\glnamefont}%
5964       \glnameaccessdisplay
5965       {%
5966         \@glstr@glossnamefont{\Glsentryname{#1}}%
5967       }%
5968     }%
5969     \glstrpostnamehook{#1}%
5970   }%
5971 }
5972 }
5973 {
5974   \renewcommand*{\Glossentryname}[1]{%
5975     \@glsdoifexistsorwarn{#1}%
5976     {%
5977       \glsetabbrvfmt{\glscategory{#1}}%
5978       \glshasattribute{#1}{glossnamefont}%
5979       {%
5980         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5981         \ifcsdef{\@glstr@attrval}%
5982         {%
5983           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5984         }%
5985         {%
5986           \GlossariesExtraWarning{Unknown control sequence name
5987             '\@glstr@attrval' supplied in glossnamefont attribute
```

Do post-name hook:

```
5969   \glstrpostnamehook{#1}%
5970 }%
5971 }
5972 }
5973 {
5974   \renewcommand*{\Glossentryname}[1]{%
5975     \@glsdoifexistsorwarn{#1}%
5976     {%
5977       \glsetabbrvfmt{\glscategory{#1}}%
5978       \glshasattribute{#1}{glossnamefont}%
5979       {%
5980         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5981         \ifcsdef{\@glstr@attrval}%
5982         {%
5983           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5984         }%
5985         {%
5986           \GlossariesExtraWarning{Unknown control sequence name
5987             '\@glstr@attrval' supplied in glossnamefont attribute
```

```

5988         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5989         \let\@glsxtr@glossnamefont\glsnamefont
5990     }%
5991 }%
5992 {\let\@glsxtr@glossnamefont\glsnamefont}%
5993 \@glsxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

5994     \glsxtrpostnamehook{#1}%
5995 }%
5996 }
5997 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

5998 \newcommand*{\glsxtrpostnamehook}[1]{%
5999   \let\@glsnumberformat\@glsxtr@defaultnumberformat
6000   \glsxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

6001   \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

6002   \csuse{glsxtrpostname\glscategory{#1}}%
6003 }

```

`trapostnamehook`

```

6004 \newcommand*{\glsextrapostnamehook}[1]{%

```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```

6005 \newcommand*{\glsdefpostname}[2]{%
6006   \csdef{glsxtrpostname#1}{#2}%
6007 }

```

`etaccessdisplay`

```

6008 \@ifpackageloaded{glossaries-accsupp}
6009 {
6010   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6011     \ifcsdef{gls#1accessdisplay}%
6012     {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
6013     {%

```

This is essentially the reverse of `\@gls@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gls<field>accessdisplay` commands use the key name.

```

6014     \edef\@gls@thisval{#1}%
6015     \@for\@gls@map:=\@gls@keymap\do{%

```

```

6016      \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6017      \ifdefequal{\@this@key}{\@gls@thisval}%
6018      {%
6019          \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6020          \@endfortrue
6021      }%
6022      {}%
6023      }%
6024      \ifcsdef{gls\@gls@thisval accessdisplay}%
6025      {\letcs\@glxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
6026      {\let\@glxtr@accessdisplay\@firstoftwo}%
6027      }%
6028  }
6029 }
6030 {%
6031     \newcommand*{\glxtr@setaccessdisplay}[1]{%
6032         \let\@glxtr@accessdisplay\@firstoftwo
6033     }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

6034 \newrobustcmd*{\glossentrynameother}[2]{%
6035     \@glsdoifexistsorwarn{#1}%
6036     {%

```

Accessibility support:

```

6037     \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6038     \glsssetabbrvfmt{\glscategory{#1}}%
6039     \glshasattribute{#1}{glossnamefont}%
6040     {%
6041         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6042         \ifcsdef{\@glxtr@attrval}%
6043         {%
6044             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6045             }%
6046             {%
6047                 \GlossariesExtraWarning{Unknown control sequence name
6048                 '@@glxtr@attrval' supplied in glossnamefont attribute
6049                 for entry '#1'. Reverting to default \string\glssnamefont}%
6050                 \let\@glxtr@glossnamefont\glssnamefont
6051             }%
6052             }%
6053             {\let\@glxtr@glossnamefont\glssnamefont}%
6054             \glssifattribute{#1}{glossname}{firstuc}%
6055             {%
6056                 \@glxtr@accessdisplay
6057                 {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6058                 {#1}%

```

```

6059 }%
6060 {%
6061   \glsifattribute{#1}{glossname}{title}%
6062   {%
6063     \@glsxtr@do@titlecaps@warn
6064     \@glsxtr@accessdisplay
6065     {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
6066     {#1}%
6067   }%
6068   {%
6069     \glsifattribute{#1}{glossname}{uc}%
6070     {%
6071       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6072       \@glsxtr@accessdisplay
6073       {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6074       {#1}%
6075     }%
6076     {%
6077       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6078       \@glsxtr@accessdisplay
6079       {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6080       {#1}%
6081     }%
6082   }%
6083 }%

```

Do post-name hook.

```

6084   \glsxtrpostnamehook{#1}%
6085 }%
6086 }

```

`format@override` Determines if the `format` key should override the indexing attribute value.

```

6087 \newif\if@glsxtr@format@override
6088 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6089 \@ifpackageloaded{hyperref}
6090 {

```

If `hyperref`'s `hyperindex` option is on, then `hyperref` will automatically add `\hyperpage`, so don't add it.

```

6091   \ifHy@hyperindex
6092     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6093       \@glsxtr@format@override true
6094       \appto\theindex{\let\glshypernumber\@firstofone}%
6095     }
6096   \else

```

```

6097 \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6098 \@glsextr@format@overridetrue
6099 \appto\theindex{\let\glshypernumber\hyperpage}%
6100 }
6101 \fi
6102 }
6103 {
6104 \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6105 \@glsextr@format@overridetrue
6106 }
6107 }
6108 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6109 \newcommand*{\glsextrdoautoindexname}[2]{%
6110 \glshasattribute{#1}{#2}%
6111 {%

```

Escape any makeindex/xindy characters in the value of the name field. Take care with babel as this won't work if the category code has changed for those characters.

```

6112 \@glsextr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

6113 \protected@edef\@glsextr@attrval{\glsggetattribute{#1}{#2}}%
6114 \if@glsextr@format@override

6115 \ifx\@glsnumberformat\@glsextr@defaultnumberformat
6116 \else
6117 \let\@glsextr@attrval\@glsnumberformat
6118 \fi
6119 \fi
6120 \ifdefstring{\@glsextr@attrval}{true}%
6121 {}%
6122 {\eappto\@glo@name{\@glsextr@autoindex@encap\@glsextr@attrval}}%
6123 \expandafter\glsextrautoindex\expandafter{\@glo@name}%
6124 }%
6125 {}%
6126 }

```

glsextrautoindex

```

6127 \newcommand*{\glsextrautoindex}{\index}

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6128 \newcommand*{\@glsextr@autoindex@setname}[1]{%
6129 \protected@edef\@glo@name{\glsextrautoindexentry{#1}}%
6130 \glsextrautoindexassignsort{\@glo@sort}{#1}%
6131 \@gls@checkmkidxchars\@glo@sort
6132 \@glsextr@autoindex@doextra@esc\@glo@sort
6133 \epreto\@glo@name{\@glo@sort\@glsextr@autoindex@at}%
6134 }

```

autoindexentry Command used for the actual part when auto-indexing.

```
6135 \newcommand*{\glxtrautoindexentry}[1]{\string\glstryname{#1}}
```

indexassignsort Used to assign the sort value when auto-indexing.

```
6136 \newcommand*{\glxtrautoindexassignsort}[2]{%
6137   \glsletentryfield{#1}{#2}{sort}%
6138 }
```

doextra@esc

```
6139 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6140   \ifx\@glxtr@autoindex@esc\@gl@quotechar
6141   \else
6142     \def\@gl@checkedmkidx{}%
6143     \edef\@glxtr@checkspch{%
6144       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
6145       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
6146       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
6147     \@glxtr@checkspch
6148     \let#1\@gl@checkedmkidx\relax
6149   \fi
```

Escape actual character unless it has already been escaped.

```
6150   \ifx\@glxtr@autoindex@at\@gl@actualchar
6151   \else
6152     \def\@gl@checkedmkidx{}%
6153     \edef\@glxtr@checkspch{%
6154       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
6155       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
6156       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
6157     \@glxtr@checkspch
6158     \let#1\@gl@checkedmkidx\relax
6159   \fi
```

Escape level character unless it has already been escaped.

```
6160   \ifx\@glxtr@autoindex@level\@gl@levelchar
6161   \else
6162     \def\@gl@checkedmkidx{}%
6163     \edef\@glxtr@checkspch{%
6164       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
6165       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
6166       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
6167     \@glxtr@checkspch
6168     \let#1\@gl@checkedmkidx\relax
6169   \fi
```

Escape encap character unless it has already been escaped.

```
6170   \ifx\@glxtr@autoindex@encap\@gl@encapchar
6171   \else
```

```

6172 \def\@gls@checkedmkidx{}%
6173 \edef\@glsxtr@checkspch{%
6174 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6175 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6176 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6177 \@glsxtr@checkspch
6178 \let#1\@gls@checkedmkidx\relax
6179 \fi
6180 }

```

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`.

```
6181 \newcommand*{\@glsxtr@autoindex@at}{}

```

`trSetActualChar` Set the actual character.

```

6182 \newcommand*{\GlsXtrSetActualChar}[1]{%
6183 \gdef\@glsxtr@autoindex@at{#1}%
6184 \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
6185 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6186 }%
6187 }
6188 \@onlypreamble\GlsXtrSetActualChar
6189 \makeatother
6190 \GlsXtrSetActualChar{@}
6191 \makeatletter

```

`autoindex@encap` Encap character for use with `\index`.

```
6192 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`XtrSetEncapChar` Set the encap character.

```

6193 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6194 \gdef\@glsxtr@autoindex@encap{#1}%
6195 \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
6196 \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
6197 }%
6198 }
6199 \GlsXtrSetEncapChar{|}
6200 \@onlypreamble\GlsXtrSetEncapChar

```

`autoindex@level` Level character for use with `\index`.

```
6201 \newcommand*{\@glsxtr@autoindex@level}{}

```

`XtrSetLevelChar` Set the encap character.

```

6202 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6203 \gdef\@glsxtr@autoindex@level{#1}%
6204 \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%

```

```

6205 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
6206 }%
6207 }
6208 \GlsXtrSetLevelChar{!}
6209 \@onlypreamble\GlsXtrSetLevelChar

```

r@autoindex@esc Escape character for use with \index.

```

6210 \newcommand*{\@glsxtr@autoindex@esc}{"}

```

lsXtrSetEscChar Set the escape character.

```

6211 \newcommand*{\GlsXtrSetEscChar}[1]{%
6212 \gdef\@glsxtr@autoindex@esc{#1}%
6213 \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6214 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6215 }%
6216 }
6217 \GlsXtrSetEscChar{"}
6218 \@onlypreamble\GlsXtrSetEscChar

```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```

6219 \ifdef\actualchar
6220 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6221 {}

```

Quote character \quotechar:

```

6222 \ifdef\quotechar
6223 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6224 {}

```

Level character \levelchar:

```

6225 \ifdef\levelchar
6226 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6227 {}

```

Encap character \encapchar:

```

6228 \ifdef\encapchar
6229 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6230 {}

```

leto@endescspch

```

6231 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}

```

toindex@esc@spch \@@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}

```

6232 \newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
6233 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
6234 \toks@={#3}%

```

```

6235 \ifx\@nnil#3\relax
6236 \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch#5\@glxstr@endescspch}%
6237 \else
6238 \ifx\@nnil#4\relax
6239 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
6240 \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch
6241 #4#5\@glxstr@endescspch}%
6242 \else
6243 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
6244 \@glxstr@autoindex@esc#1}%
6245 \def\@glxstr@checkspch{#2#5#1\@nnil#1\@glxstr@endescspch}%
6246 \fi
6247 \fi
6248 \@glxstr@checkspch
6249 }

```

`\Glossentrydesc` Redefine to set the abbreviation format and accessibility support.

```

6250 \renewcommand*{\Glossentrydesc}[1]{%
6251 \glsdoifexistsorwarn{#1}%
6252 {%
6253 \glssetabbrvfmt{\glscategory{#1}}%
6254 \Glsaccessdesc{#1}%
6255 }%
6256 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

6257 \renewcommand*{\Glossentrysymbol}[1]{%
6258 \glsdoifexistsorwarn{#1}%
6259 {%
6260 \glssetabbrvfmt{\glscategory{#1}}%
6261 \Glsaccesssymbol{#1}%
6262 }%
6263 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

6264 \renewcommand*{\Glossentrysymbol}[1]{%
6265 \glsdoifexistsorwarn{#1}%
6266 {%
6267 \glssetabbrvfmt{\glscategory{#1}}%
6268 \Glsaccesssymbol{#1}%
6269 }%
6270 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\GlsXtrEnableInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```

6271 \newcommand*{\GlsXtrEnableInitialTagging}{%

```

```

6272 \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6273 }
6274 \@onlypreamble\GlsXtrEnableInitialTagging

```

r@enabletagging Starred version undefines command.

```

6275 \newcommand*{\s@glxtr@enabletagging}[2]{%
6276 \undef#2%
6277 \@glxtr@enabletagging{#1}{#2}%
6278 }

```

r@enabletagging Internal command.

```

6279 \newcommand*{\@glxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6280 \@for\@glxtr@cat:=#1\do
6281 {%
6282 \ifdefempty\@glxtr@cat
6283 {}%
6284 {\glsssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%
6285 }%
6286 \newrobustcmd*#2[1]{##1}%
6287 \def\@glxtr@taggingcs{#2}%
6288 \renewcommand*\@glxtr@activate@initialtagging{%
6289 \let#2\@glxtr@tag
6290 }%
6291 \ifundef\@glx@preglossaryhook
6292 {\GlossariesExtraWarning{Initial tagging requires at least
6293 glossaries.sty v4.19 to work correctly}}%
6294 {}%
6295 }

```

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

```

6296 \ifundef\mfu@checkword@do
6297 {
6298 \newcommand*{\mfu@checkword@do}[1]{%
6299 \ifdefstring{\mfu@checkword@arg}{#1}%
6300 {%
6301 \let\@mfu@domakefirstuc\@firstofone
6302 \listbreak
6303 }%
6304 {}%
6305 }

```

\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.

```

6306 \ifundef\mfu@checkword

```

```

6307 {
6308   \newcommand{\@glxtr@do@titlecaps@warn}{%
6309     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6310       support not available}%
        One warning should suffice.
6311     \let\@glxtr@do@titlecaps@warn\relax
6312   }
6313 }
6314 {
6315   \renewcommand*\mfu@checkword}[1]{%
6316     \def\mfu@checkword@arg{#1}%
6317     \let\@mfu@domakefirstuc\makefirstuc
6318     \forlistloop\mfu@checkword@do\@mfu@nocaplist
6319   }
6320 }
6321 }
6322 {}% no patch required

```

@titlecaps@warn Do warning if title case not supported.

```
6323 \newcommand*\@glxtr@do@titlecaps@warn{}
```

@initialtagging Used in \printglossary but at least v4.19 of glossaries required.

```
6324 \newcommand*\@glxtr@activate@initialtagging{}
```

\@glxtr@tag Definition of tagging command when used in glossary.

```

6325 \newrobustcmd*\@glxtr@tag}[1]{%
6326   \glsifattribute{\glscurrententrylabel}{tagging}{true}%
6327   {\glxtrtagfont{#1}}{#1}%
6328 }

```

\glxtrtagfont Used in the glossary.

```
6329 \newcommand*\glxtrtagfont}[1]{\underline{#1}}
```

preglossaryhook This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```

6330 \ifdef\@gls@preglossaryhook
6331 {
6332   \renewcommand*\@gls@preglossaryhook{%
6333     \@glxtr@activate@initialtagging

```

Since the glossaries are automatically scoped, \@glxtr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```

6334   \ifundef\@glxtr@org@postdescription
6335   {%
6336     \let\@glxtr@org@postdescription\glspostdescription
6337     \renewcommand*\glspostdescription{%

```

```

6338     \ifglentryexists{\glscurrententrylabel}%
6339     {%
6340         \glxtrpostdescription
6341         \@glxtr@org@postdescription
6342     }%
6343     {}%
6344 }%
6345 }%
6346 {}%

```

Enable the options used by \@glxtrp:

```

6347     \glossxtrsetpopts
6348 }%
6349 }
6350 {}

```

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.

```

6351 \newcommand*{\glxtrpostdescription}{%
6352     \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
6353 }

```

postdescgeneral

```

6354 \newcommand*{\glxtrpostdescgeneral}{}

```

xtrpostdescterm

```

6355 \newcommand*{\glxtrpostdescterm}{}

```

postdescacronym

```

6356 \newcommand*{\glxtrpostdescacronym}{}

```

escabbreviation

```

6357 \newcommand*{\glxtrpostdescabbreviation}{}

```

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.

```

6358 \newcommand*{\glsdefpostdesc}[2]{%
6359     \csdef{glxtrpostdesc#1}{#2}%
6360 }

```

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.

```

6361 \renewcommand*{\glspostlinkhook}{%
6362     \ifglentryexists{\glslabel}{\glxtrpostlinkhook}{}%
6363 }

```

trpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.

```
6364 \newcommand*{\glsxtrpostlinkhook}{%
6365   \glsxtrdiscardperiod{\glslabel}%
6366   {\glsxtrpostlinkendsentence}%
6367   {\glsxtrifcustomdiscardperiod
6368     {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
6369   {\glsxtrpostlink}%
6370 }%
6371 }
```

omdiscardperiod Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6372 \newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

\glsxtrpostlink

```
6373 \newcommand*{\glsxtrpostlink}{%
6374   \csuse{\glsxtrpostlink\glscategory{\glslabel}}%
6375 }
```

\glsdefpostlink Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite \glsxtrpostlink.

```
6376 \newcommand*{\glsdefpostlink}[2]{%
    \ifthenelse is used to ensure that the expanded value is tested. (The category label must
    be fully expandable.)
6377   \ifthenelse{\equal{#1}{}}{%
6378     {\PackageError{glossaries-extra}
6379       {Invalid empty category label in \string\glsdefpostlink}{}}%
6380     {\csdef{\glsxtrpostlink#1}{#2}}%
6381 }
```

linkendsentence Done by \glsxtrpostlinkhook if a full stop is discarded.

```
6382 \newcommand*{\glsxtrpostlinkendsentence}{%
6383   \ifcsdef{\glsxtrpostlink\glscategory{\glslabel}}
6384   {%
6385     \csuse{\glsxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
6386     .\spacefactor\sfcode'\. \relax
6387   }%
6388   {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
6389     \spacefactor\sfcode'\. \relax
6390   }%
6391 }
```

dDescOnFirstUse Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6392 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6393   \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}%
6394 }
```

ymbolOnFirstUse Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6395 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6396   \glxtrifwasfirstuse
6397   {%
6398     \ifglshassymbol{\glslabel}%
6399     {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
6400     }%
6401   }%
6402   {}%
6403 }
```

lDescOnFirstUse Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6404 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6405   \glxtrifwasfirstuse
6406   {%
6407     \space\glxtrparen
6408     {%
6409       \ifglshassymbol{\glslabel}%
6410       {\glssaccesssymbol{\glslabel}, }%
6411       }%
6412       \glssaccessdesc{\glslabel}%
6413     }%
6414   }%
6415   {}%
6416 }
```

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).

```
6417 \newcommand*{\glxtrdiscardperiod}[3]{%
6418   \glxtrifwasfirstuse
6419   {%
6420     \glusifattribute{#1}{retainfirstuseperiod}{true}%
6421     {#3}%
6422     {%
6423       \glusifattribute{#1}{discardperiod}{true}%
6424       {%
6425         \glusifplural
6426         {%
```

```

6427      \glsifattribute{#1}{pluraldiscardperiod}{true}%
6428      {\glsxtrifperiod{#2}{#3}}%
6429      {#3}%
6430      }%
6431      {%
6432      \glsxtrifperiod{#2}{#3}%
6433      }%
6434      }%
6435      {#3}%
6436      }%
6437  }%
6438  {%
6439  \glsifattribute{#1}{discardperiod}{true}%
6440  {%
6441  \glsifplural
6442  {%
6443  \glsifattribute{#1}{pluraldiscardperiod}{true}%
6444  {\glsxtrifperiod{#2}{#3}}%
6445  {#3}%
6446  }%
6447  {%
6448  \glsxtrifperiod{#2}{#3}%
6449  }%
6450  }%
6451  {#3}%
6452  }%
6453  }

```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```
6454 \newcommand*{\glsxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}
```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```
6455 \newcommand*{\glsxtr@punclist}{.,:;?!}
```

`\punctuationmark` Add character to punctuation list.

```
6456 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`\punctuationmarks` Reset the punctuation list.

```
6457 \newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}
```

`\glsxtrifpunc` `\glsxtrifnextpunc{<true part>}{<>false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```
6458 \newcommand*{\glxtrifnextpunc}[2]{%
6459   \def\reserved@a{#1}%
6460   \def\reserved@b{#2}%
6461   \futurelet\@glspunc@token\glxtr@ifnextpunc
6462 }
```

`\glxtr@ifnextpunc`

```
6463 \newcommand*{\glxtr@ifnextpunc}{%
6464   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6465   \reserved@b
6466 }
```

`\glxtr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```
6467 \newcommand*{\glxtr@ifpunctoken}[1]{%
6468   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6469 }
```

`\glxtr@ifpunctoken`

```
6470 \def\@glxtr@ifpunctoken#1#2{%
6471   \let\reserved@d=#2%
6472   \ifx\reserved@d\@nnil
6473     \let\glxtr@next\@glxtr@notfoundinlist
6474   \else
6475     \ifx#1\reserved@d
6476       \let\glxtr@next\@glxtr@foundinlist
6477     \else
6478       \let\glxtr@next\@glxtr@ifpunctoken
6479     \fi
6480   \fi
6481   \glxtr@next#1%
6482 }
```

`\glxtr@foundinlist`

```
6483 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

`\glxtr@notfoundinlist`

```
6484 \def\@glxtr@notfoundinlist#1{\@secondoftwo}
```

`\glxtr@dopostpunc`

```
\glxtr@dopostpunc{<code>}
```

If this is followed by a punctuation character, do `<code>` after the character otherwise do `<code>` before whatever comes next.

```
6485 \newcommand{\glxtr@dopostpunc}[1]{%
6486   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6487 }
```

@glxtr@swaptwo

```
6488 \newcommand{\@glxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
6489 \define@key{glxtrabbrv}{category}{%
6490   \edef\glscategorylabel{#1}%
6491   \ifcsdef{@glsabbrv@current@#1}%
6492   {%
```

Warning should already have been issued.

```
6493   \let\@glxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6494   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6495   \glxtr@applyabbrvstyle{\cname @glsabbrv@current@#1\endcsname}%
6496   \let\GlsXtrWarnDeprecatedAbbrStyle\@glxtr@orgwarndep
6497 }%
6498 {}%
6499 }
```

Save the short plural form. This may be needed before the entry is defined.

```
6500 \define@key{glxtrabbrv}{shortplural}{%
6501   \def\@gl@shortpl{#1}%
6502 }
```

Similarly for the long plural form.

```
6503 \define@key{glxtrabbrv}{longplural}{%
6504   \def\@gl@longpl{#1}%
6505 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
6506 \newtoks\glsshortpltok
```

\glslongpltok

```
6507 \newtoks\glslongpltok
```

`\glxtr@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.

```

6508 \newcommand*{\@glxtr@insertdots}[2]{%
6509   \def#1{}%
6510   \@glxtr@insert@dots#1#2\@nnil
6511 }

```

xtr@insert@dots

```

6512 \newcommand*{\@glxtr@insert@dots}[2]{%
6513   \ifx\@nnil#2\relax
6514   \let\@glxtr@insert@dots@next\@gobble
6515   \else
6516   \ifx\relax#2\relax
6517   \else
6518     \appto#1{#2.}%
6519   \fi
6520   \let\@glxtr@insert@dots@next\@glxtr@insert@dots
6521   \fi
6522   \@glxtr@insert@dots@next#1%
6523 }

```

Similarly provide a way of replacing spaces with \glxtrwordsep, which first needs to be defined:

\glxtrwordsep

```

6524 \newcommand*{\glxtrwordsep}{\space}

```

Each word is marked with

\glxtrword

```

6525 \newcommand*{\glxtrword}[1]{#1}

```

tr@markwordseps

```

6526 \newcommand*{\@glxtr@markwordseps}[2]{%
6527   \def#1{}%
6528   \@glxtr@mark@wordseps#1#2 \@nnil
6529 }

```

r@mark@wordseps

```

6530 \def\@glxtr@mark@wordseps#1#2 #3{%
6531   \ifdefempty{#1}%
6532   {\def#1{\protect\glxtrword{#2}}}%
6533   {\appto#1{\protect\glxtrwordsep\protect\glxtrword{#2}}}%
6534   \ifx\@nnil#3\relax
6535   \let\@glxtr@mark@wordseps@next\relax
6536   \else
6537   \def\@glxtr@mark@wordseps@next{%
6538     \@glxtr@mark@wordseps#1#3}%
6539   \fi
6540   \@glxtr@mark@wordseps@next
6541 }

```

`newabbreviation` Define a new generic abbreviation.

```

6542 \newcommand*{\newabbreviation}[4] [] {%
6543   \glstr@newabbreviation{#1}{#2}{#3}{#4}%
6544 }

```

`newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This just makes it easier to save and restore the original definition.)

```

6545 \newcommand*{\glstr@newabbreviation}[4] {%
6546   \glstok{#1}%
6547   \glstok{#2}%
6548   \glstok{#3}%
6549   \glstok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```

6550   \def\glstr@short{#3}%
6551   \def\glstr@long{#4}%

```

Provide extra settings for hooks (if modified, this command must end with a comma).

```

6552   \def\ExtraCustomAbbreviationFields{}%

```

Initialise accessibility settings if required.

```

6553   \@glst@initaccesskeys

```

Get the category.

```

6554   \def\glscategorylabel{abbreviation}%
6555   \glstr@applyabbrvstyle{\@glst@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```

6556   \setkeys*{\glstr@abbrv}{shortplural,longplural}{#1}%

```

Set the default long plural

```

6557   \def\@glst@longpl{#4\glstpluralsuffix}%
6558   \let\@glst@default@longpl\@glst@longpl

```

Has the markwords attribute been set?

```

6559   \glstifcategoryattribute{\glscategorylabel}{markwords}{true}%
6560   {%
6561     \@glstr@markwordseps\@glst@long{#4}%
6562     \expandafter\def\expandafter\@glst@longpl\expandafter
6563       {\@glst@long\glstpluralsuffix}%
6564     \let\@glst@default@longpl\@glst@longpl

```

Update `\glstlongtok`.

```

6565     \expandafter\glstlongtok\expandafter{\@glst@long}%
6566   }%
6567 }%

```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```

6568   \glstifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6569   {%
6570     \@glstr@markwordseps\@glst@short{#3}%
6571   }%
6572   {%

```

Has the insertdots attribute been set?

```
6573 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6574 {%
6575 \@glstr@insertdots\@gls@short{#3}%
6576 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6577 }%
6578 {\def\@gls@short{#3}}%
6579 }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6580 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6581 {%
6582 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6583 '\abbrvpluralsuffix}%
6584 }%
6585 {%
```

Has the noshortplural attribute been set?

```
6586 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6587 {%
6588 \let\@gls@shortpl\@gls@short
6589 }%
6590 {%
6591 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6592 \abbrvpluralsuffix}%
6593 }%
6594 }%
```

Update \glsshorttok:

```
6595 \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
6596 \glstrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
6597 \setkeys*{glstrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
6598 \ifx\@gls@default@longpl\@gls@longpl
6599 \else
```

Has the markwords attribute been set?

```
6600 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6601 {%
6602 \expandafter\@glstr@markwordseps\expandafter\@gls@longpl\expandafter
6603 {\@gls@longpl}%
6604 }%
6605 {}%
6606 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
6607 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6608 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6609 \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6610 \newabbreviationhook
```

Define this entry:

```
6611 \protected@edef\@do@newglossaryentry{%
6612 \noexpand\newglossaryentry{\the\glslabeltok}%
6613 {%
6614 type=\glstrabbrvtype,%
6615 category=abbreviation,%
6616 short={\the\glsshorttok},%
6617 shortplural={\the\glsshortpltok},%
6618 long={\the\glslongtok},%
6619 longplural={\the\glslongpltok},%
6620 name={\the\glsshorttok},%
6621 \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6622 \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6623 \the\glskeylisttok
6624 }%
6625 }%
6626 \@do@newglossaryentry
6627 \GlsXtrPostNewAbbreviation
6628 }
```

evpresetkeyhook Hook for extra stuff in `\newabbreviation`

```
6629 \newcommand*{\glstrnewabbspresetkeyhook}[3]{}%
```

NewAbbreviation Hook used by abbreviation styles.

```
6630 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

bbreviationhook Hook for use with `\newabbreviation`.

```
6631 \newcommand*{\newabbreviationhook}{}%
```

reviationFields

```
6632 \newcommand*{\CustomAbbreviationFields}{}%
```

\glstrparen For the parenthetical styles.

```
6633 \newcommand*{\glstrparen}[1]{(#1)}
```

`\lsxtrfullformat` Full format without case change.

```

6634 \newcommand*{\lsxtrfullformat}[2]{%
6635   \glsfirstlongfont{\glsaccesslong{#1}}#2\lsxtrfullsep{#1}%
6636   \lsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6637 }

```

`\lsxtrfullformat` Full format with case change.

```

6638 \newcommand*{\Glsxtrfullformat}[2]{%
6639   \glsfirstlongfont{\Glsaccesslong{#1}}#2\lsxtrfullsep{#1}%
6640   \lsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6641 }

```

`\xtrfullplformat` Plural full format without case change.

```

6642 \newcommand*{\xtrfullplformat}[2]{%
6643   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\lsxtrfullsep{#1}%
6644   \lsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
6645 }

```

`\xtrfullplformat` Plural full format with case change.

```

6646 \newcommand*{\Glxtrfullplformat}[2]{%
6647   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\lsxtrfullsep{#1}%
6648   \lsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
6649 }

```

`\glsxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```

6650 \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.

```

6651 \newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}

```

`\inlinefullformat` Full format with case change.

```

6652 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}

```

`\xtrfullplformat` Plural full format without case change.

```

6653 \newcommand*{\xtrinlinefullplformat}{\glsxtrfullplformat}

```

`\inefullplformat` Plural full format with case change.

```

6654 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}

```

Redefine `\glsentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glsxtrfull` set of commands instead.

`\glsentryfull`

```

6655 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}

```

`\Glsentryfull`
6656 `\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}`

`\glsentryfullpl`
6657 `\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}`

`\Glsentryfullpl`
6658 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
6659 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
6660 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.
6661 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`
6662 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
6663 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.
6664 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.
6665 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`
6666 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.
6667 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glsbrvpluralsuffix` Default plural suffix.
6668 `\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).
6669 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`
6670 `\newcommand*\ns@glsxtrfull[2][\%]`
6671 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{\%`
6672 `{\@glsxtr@full{#1}{#2}[]}\%`
6673 `}`

`\@glxtr@full` Low-level macro:

```
6674 \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).

```
6675 \@glxtr@record{#1}{#2}{glslink}%
6676 \glstoifexists{#2}%
6677 {%
6678 \glsetabbrfmt{\glscategory{#2}}%
6679 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6680 \let\gl@ifplural\@secondoftwo
6681 \let\gl@scapscase\@firstofthree
6682 \let\gl@insert\@empty
6683 \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.

```
6684 \glxtrsetupfulldefs
6685 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6686 }%
6687 \glspostlinkhook
6688 }
```

`trsetupfulldefs`

```
6689 \newcommand*{\glxtrsetupfulldefs}{%
6690 \let\glxtrifwasfirstuse\@firstoftwo
6691 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
6692 \newrobustcmd*{\Glsxtrfull}{\@gl@hyp@opt\@ns@Glsxtrfull}
6693 \newcommand*{\ns@Glsxtrfull}[2][{}]{%
6694 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
6695 {\@Glsxtr@full{#1}{#2}[]}%
6696 }
```

`\@Glsxtr@full` Low-level macro:

```
6697 \def\@Glsxtr@full#1#2[#3]{%
6698 \glstoifexists{#2}%
6699 {%
6700 \glsetabbrfmt{\glscategory{#2}}%
6701 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6702 \let\gl@ifplural\@secondoftwo
6703 \let\gl@scapscase\@secondofthree
6704 \let\gl@insert\@empty
6705 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6706 \glxtrsetupfulldefs
6707 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%

```

```

6708 }%
6709 \glspostlinkhook
6710 }

```

`\GLSxtrfull` Full form (all uppercase).

```

6711 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\@ns@GLSxtrfull}
6712 \newcommand*\ns@GLSxtrfull[2][{}]{%
6713   \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}]{%
6714     {\@GLSxtr@full{#1}{#2}[]}%
6715 }

```

`\@GLSxtr@full` Low-level macro:

```

6716 \def\@GLSxtr@full#1#2[#3]{%
6717   \glstoifexists{#2}%
6718   {%
6719     \glsssetabbrvfmt{\glscategory{#2}}%
6720     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6721     \let\glsifplural\@secondoftwo
6722     \let\glscapscase\@thirdofthree
6723     \let\glsinsert\@empty
6724     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6725     \glsxtrsetupfulldefs
6726     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6727   }%
6728   \glspostlinkhook
6729 }

```

`\glsxtrfullpl` Plural full form (no case-change).

```

6730 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\@ns@glsxtrfullpl}
6731 \newcommand*\ns@glsxtrfullpl[2][{}]{%
6732   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}]{%
6733     {\@glsxtr@fullpl{#1}{#2}[]}%
6734 }

```

`\@glsxtr@fullpl` Low-level macro:

```

6735 \def\@glsxtr@fullpl#1#2[#3]{%
  If the record option has been used, the information needs to be written to the aux file regard-
  less of whether the entry exists (unless indexing has been switched off).
6736   \@glsxtr@record{#1}{#2}{\glslink}%
6737   \glstoifexists{#2}%
6738   {%
6739     \glsssetabbrvfmt{\glscategory{#2}}%
6740     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6741     \let\glsifplural\@firstoftwo
6742     \let\glscapscase\@firstofthree
6743     \let\glsinsert\@empty
6744     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6745     \glsxtrsetupfulldefs

```

```

6746 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6747 }%
6748 \glspostlinkhook
6749 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

6750 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
6751 \newcommand*\ns@Glsxtrfullpl[2][\%
6752 \new@ifnextchar[\@Glsxtr@fullpl{#1}{#2}]{\%
6753 \@Glsxtr@fullpl{#1}{#2}[]}%
6754 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6755 \def\@Glsxtr@fullpl#1#2[#3]{\%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6756 \glsxtr@record{#1}{#2}{glslink}%
6757 \glsdoifexists{#2}%
6758 {%
6759 \glssetabbrvfmt{\gls@category{#2}}}%
6760 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6761 \let\glsifplural\@firstoftwo
6762 \let\gls@scapscase\@secondofthree
6763 \let\glsinsert\@empty
6764 \def\gls@customtext{\Glsxtr@inlinefullplformat{#2}{#3}}%
6765 \glsxtr@setupfulldefs
6766 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6767 }%
6768 \glspostlinkhook
6769 }

```

`\GLSxtrfullpl` Plural full form (all upper case).

```

6770 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
6771 \newcommand*\ns@GLSxtrfullpl[2][\%
6772 \new@ifnextchar[\@GLSxtr@fullpl{#1}{#2}]{\%
6773 \@GLSxtr@fullpl{#1}{#2}[]}%
6774 }

```

`\@GLSxtr@fullpl` Low-level macro:

```

6775 \def\@GLSxtr@fullpl#1#2[#3]{\%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6776 \glsxtr@record{#1}{#2}{glslink}%
6777 \glsdoifexists{#2}%
6778 {%
6779 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6780 \let\glsifplural\@firstoftwo

```

```

6781 \let\glscapscase\@thirdofthree
6782 \let\glsinsert\@empty
6783 \def\glscustomtext{%
6784     \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
6785 \glsxtrsetupfulldefs
6786 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6787 }%
6788 \glspostlinkhook
6789 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

6790 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\ns@glsxtrshort}

```

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

```

6791 \newcommand*{\ns@glsxtrshort}[2] [] {%
6792 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2} []}%
6793 }

```

Read in the final optional argument:

```

6794 \def\@glsxtrshort#1#2[#3] {%

```

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

```

6795 \@glsxtr@record{#1}{#2}{glslink}%
6796 \glsdoifexists{#2}%
6797 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

6798 \glssetabbrvfmt{\glscategory{#2}}%
6799 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6800 \let\glsxtrifwasfirstuse\@secondoftwo
6801 \let\glsifplural\@secondoftwo
6802 \let\glscapscase\@firstofthree
6803 \let\glsinsert\@empty
6804 \def\glscustomtext{%
6805     \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
6806     \ifglsxtrininsertinside\else#3\fi
6807 }%
6808 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6809 }%
6810 \glspostlinkhook
6811 }

```

`\Glsxtrshort`

```

6812 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}

```

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

```

6813 \newcommand*{\ns@Glsxtrshort}[2] [] {%
6814 \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
6815 }

```

Read in the final optional argument:

```
6816 \def\@Glsxtrshort#1#2[#3]{%
```

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

```
6817 \@glxtr@record{#1}{#2}{glslink}%
6818 \glsoifexists{#2}%
6819 {%
6820 \glsetabbrvfmt{\glscategory{#2}}%
6821 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6822 \let\glxtrifwasfirstuse\@secondoftwo
6823 \let\gl@ifplural\@secondoftwo
6824 \let\glscapscase\@secondofthree
6825 \let\glinsert\@empty
6826 \def\glscustomtext{%
6827 \glabbrvfont{\Glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
6828 \ifglxtrininsertinside\else#3\fi
6829 }%
6830 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6831 }%
6832 \glspostlinkhook
6833 }
```

\GLSxtrshort

```
6834 \newrobustcmd*{\GLSxtrshort}{\@gl@hyp@opt\@ns@GLSxtrshort}
```

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

```
6835 \newcommand*{\ns@GLSxtrshort}[2][ ]{%
6836 \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[ ]}%
6837 }
```

Read in the final optional argument:

```
6838 \def\@GLSxtrshort#1#2[#3]{%
```

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

```
6839 \@glxtr@record{#1}{#2}{glslink}%
6840 \glsoifexists{#2}%
6841 {%
6842 \glsetabbrvfmt{\glscategory{#2}}%
6843 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6844 \let\glxtrifwasfirstuse\@secondoftwo
6845 \let\gl@ifplural\@secondoftwo
6846 \let\glscapscase\@thirdofthree
6847 \let\glinsert\@empty
6848 \def\glscustomtext{%
6849 \mfirstucMakeUppercase
6850 {\glabbrvfont{\Glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
6851 \ifglxtrininsertinside\else#3\fi
6852 }%
```

```

6853 }%
6854 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6855 }%
6856 \glspostlinkhook
6857 }

```

\glsxtrlong

```

6858 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\ns@glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6859 \newcommand*{\ns@glsxtrlong}[2] [] {%
6860   \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2} []}%
6861 }

```

Read in the final optional argument:

```

6862 \def\@glsxtrlong#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6863   \@glsxtr@record{#1}{#2}{glslink}%
6864   \glsdoifexists{#2}%
6865   {%
6866     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6867     \let\glsxtrifwasfirstuse\@secondoftwo
6868     \let\glsifplural\@secondoftwo
6869     \let\glsapscase\@firstofthree
6870     \let\glsinsert\@empty
6871     \def\glscustomtext{%
6872       \glsfont{\glsaccesslong{#2}\ifglsxtrininsertinside#3\fi}%
6873       \ifglsxtrininsertinside\else#3\fi
6874     }%
6875     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6876   }%
6877   \glspostlinkhook
6878 }

```

\Glsxtrlong

```

6879 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6880 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6881   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6882 }

```

Read in the final optional argument:

```

6883 \def\@Glsxtrlong#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6884   \@glsxtr@record{#1}{#2}{glslink}%
6885   \glsdoifexists{#2}%

```

```

6886 {%
6887   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6888   \let\glxtrifwasfirstuse\@secondoftwo
6889   \let\gl@ifplural\@secondoftwo
6890   \let\glscapscase\@secondofthree
6891   \let\glinsert\@empty
6892   \def\glscustomtext{%
6893     \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6894     \ifglxtrininsertinside\else#3\fi
6895   }%
6896   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6897 }%
6898 \glspostlinkhook
6899 }

```

\GLSxtrlong

```

6900 \newrobustcmd*{\GLSxtrlong}{\@gl@hyp@opt\ns@GLSxtrlong}

```

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

```

6901 \newcommand*{\ns@GLSxtrlong}[2] [] {%
6902   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
6903 }

```

Read in the final optional argument:

```

6904 \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).

```

6905   \@glxtr@record{#1}{#2}{glslink}%
6906   \glsoifexists{#2}%
6907   {%
6908     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6909     \let\glxtrifwasfirstuse\@secondoftwo
6910     \let\gl@ifplural\@secondoftwo
6911     \let\glscapscase\@thirdofthree
6912     \let\glinsert\@empty
6913     \def\glscustomtext{%
6914       \mfirstucMakeUppercase
6915       {\glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6916       \ifglxtrininsertinside\else#3\fi
6917     }%
6918     }%
6919     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6920   }%
6921   \glspostlinkhook
6922 }

```

Plural short forms:

\glxtrshortpl

```

6923 \newrobustcmd*{\glxtrshortpl}{\@gl@hyp@opt\ns@glxtrshortpl}

```

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

```
6924 \newcommand*{\ns@glxtrshortpl}[2] [] {%
6925   \new@ifnextchar[{\@glxtrshortpl{#1}{#2}}{\@glxtrshortpl{#1}{#2} []}%
6926 }
```

Read in the final optional argument:

```
6927 \def\@glxtrshortpl#1#2[#3] {%
```

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

```
6928   \@glxtr@record{#1}{#2}{glslink}%
6929   \glsoifexists{#2}%
6930   {%
6931     \glsetabbrfmt{\glscategory{#2}}%
6932     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6933     \let\glxtrifwasfirstuse\@secondoftwo
6934     \let\gl@ifplural\@firstoftwo
6935     \let\glscapscase\@firstofthree
6936     \let\glinsert\@empty
6937     \def\glscustomtext{%
6938       \glabbrvfont{\glaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6939       \ifglxtrininsertinside\else#3\fi
6940     }%
6941     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6942   }%
6943   \glspostlinkhook
6944 }
```

\Glsxtrshortpl

```
6945 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\ns@Glsxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
6946 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
6947   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
6948 }
```

Read in the final optional argument:

```
6949 \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).

```
6950   \@glxtr@record{#1}{#2}{glslink}%
6951   \glsoifexists{#2}%
6952   {%
6953     \glsetabbrfmt{\glscategory{#2}}%
6954     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6955     \let\glxtrifwasfirstuse\@secondoftwo
6956     \let\gl@ifplural\@firstoftwo
6957     \let\glscapscase\@secondofthree
6958     \let\glinsert\@empty
6959     \def\glscustomtext{%
```

```

6960      \glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrinsertinside#3\fi}%
6961      \ifglxtrinsertinside\else#3\fi
6962  }%
6963  \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6964  }%
6965  \glspostlinkhook
6966 }

```

\GLSxtrshortpl

```

6967 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
      Define the un-starred form. Need to determine if there is a final optional argument
6968 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
6969   \new@ifnextchar[\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6970 }

```

Read in the final optional argument:

```

6971 \def\@GLSxtrshortpl#1#2[#3] {%
      If the record option has been used, the information needs to be written to the aux file regard-
      less of whether the entry exists (unless indexing has been switched off).
6972   \@glsxtr@record{#1}{#2}{glslink}%
6973   \glsdoifexists{#2}%
6974   {%
6975     \glsssetabbrvfmt{\glscategory{#2}}%
6976     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6977     \let\glxtrifwasfirstuse\@secondoftwo
6978     \let\glsifplural\@firstoftwo
6979     \let\glscapscase\@thirdofthree
6980     \let\glsinsert\@empty
6981     \def\glscustomtext{%
6982       \mfirstucMakeUppercase
6983       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrinsertinside#3\fi}%
6984       \ifglxtrinsertinside\else#3\fi
6985     }%
6986   }%
6987   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6988   }%
6989   \glspostlinkhook
6990 }

```

Plural long forms:

\glsxtrlongpl

```

6991 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
      Define the un-starred form. Need to determine if there is a final optional argument
6992 \newcommand*{\ns@glsxtrlongpl}[2] [] {%
6993   \new@ifnextchar[\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%
6994 }

```

Read in the final optional argument:

```
6995 \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).

```
6996 \@glxstr@record{#1}{#2}{glslink}%
6997 \glstoifexists{#2}%
6998 {%
6999 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7000 \let\glxstrifwasfirstuse\@secondoftwo
7001 \let\gl@ifplural\@firstoftwo
7002 \let\glscapscase\@firstofthree
7003 \let\glinsert\@empty
7004 \def\glscustomtext{%
7005 \glslongfont{\gl@saccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
7006 \ifglxtrininsertinside\else#3\fi
7007 }%
7008 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7009 }%
7010 \glspostlinkhook
7011 }
```

\Glsxtrlongpl

```
7012 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
7013 \newcommand*{\ns@Glsxtrlongpl}[2][ ]{%
7014 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2}[ ]}%
7015 }
```

Read in the final optional argument:

```
7016 \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).

```
7017 \@glxstr@record{#1}{#2}{glslink}%
7018 \glstoifexists{#2}%
7019 {%
7020 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7021 \let\glxstrifwasfirstuse\@secondoftwo
7022 \let\gl@ifplural\@firstoftwo
7023 \let\glscapscase\@secondofthree
7024 \let\glinsert\@empty
7025 \def\glscustomtext{%
7026 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
7027 \ifglxtrininsertinside\else#3\fi
7028 }%
7029 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7030 }%
7031 \glspostlinkhook
7032 }
```

```

\GLSxtrlongpl
7033 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7034 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7035   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7036 }

    Read in the final optional argument:
7037 \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).
7038   \@glsxtr@record{#1}{#2}{glslink}%
7039   \glsdoifexists{#2}%
7040   {%
7041     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7042     \let\glsxtrifwasfirstuse\@secondoftwo
7043     \let\glsifplural\@firstoftwo
7044     \let\glscapscase\@thirdofthree
7045     \let\glsinsert\@empty
7046     \def\glscustomtext{%
7047       \mfirstucMakeUppercase
7048       {\glslongfont{\glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7049       \ifglsxtrininsertinside\else#3\fi
7050     }%
7051   }%
7052   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7053 }%
7054 \glspostlinkhook
7055 }

\glssetabbrvfmt   Set the current format for the given category (or the abbreviation category if unset).
7056 \newcommand*{\glssetabbrvfmt}[1] {%
7057   \ifcsdef{@glsabbrv@current@#1}%
7058   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7059   {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7060 }

glsuseabbrvfont   Provide a way to use the abbreviation font for a given category for arbitrary text.
7061 \newrobustcmd*{\glsuseabbrvfont}[2] {{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}

\glsuselongfont   Provide a way to use the long font for a given category for arbitrary text.
7062 \newrobustcmd*{\glsuselongfont}[2] {{\glssetabbrvfmt{#2}\glslongfont{#1}}}

sxtrgenabbrvfmt   Similar to \glsgenacfmt, but for abbreviations.
7063 \newcommand*{\glsxtrgenabbrvfmt}{%
7064   \ifdefempty\glscustomtext
7065   {%

```

7066 \ifglused\glslabel
7067 {%

Subsequent use:

7068 \glcifplural
7069 {%

Subsequent plural form:

7070 \glscapscase
7071 {%

Subsequent plural form, don't adjust case:

7072 \glxtrsubsequentplfmt{\glslabel}{\glinsert}%
7073 }%
7074 {%

Subsequent plural form, make first letter upper case:

7075 \Glsxtrsubsequentplfmt{\glslabel}{\glinsert}%
7076 }%
7077 {%

Subsequent plural form, all caps:

7078 \mfirstucMakeUppercase
7079 {\glxtrsubsequentplfmt{\glslabel}{\glinsert}}%
7080 }%
7081 }%
7082 {%

Subsequent singular form

7083 \glscapscase
7084 {%

Subsequent singular form, don't adjust case:

7085 \glxtrsubsequentfmt{\glslabel}{\glinsert}%
7086 }%
7087 {%

Subsequent singular form, make first letter upper case:

7088 \Glsxtrsubsequentfmt{\glslabel}{\glinsert}%
7089 }%
7090 {%

Subsequent singular form, all caps:

7091 \mfirstucMakeUppercase
7092 {\glxtrsubsequentfmt{\glslabel}{\glinsert}}%
7093 }%
7094 }%
7095 }%
7096 {%

First use:

7097 \glcifplural
7098 {%

First use plural form:

```
7099      \glscapscase
7100      {%
```

First use plural form, don't adjust case:

```
7101      \glxtrfullplformat{\glslabel}{\glsinsert}%
7102      }%
7103      {%
```

First use plural form, make first letter upper case:

```
7104      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
7105      }%
7106      {%
```

First use plural form, all caps:

```
7107      \mfirstucMakeUppercase
7108      {\glxtrfullplformat{\glslabel}{\glsinsert}}%
7109      }%
7110      }%
7111      {%
```

First use singular form

```
7112      \glscapscase
7113      {%
```

First use singular form, don't adjust case:

```
7114      \glxtrfullformat{\glslabel}{\glsinsert}%
7115      }%
7116      {%
```

First use singular form, make first letter upper case:

```
7117      \Glsxtrfullformat{\glslabel}{\glsinsert}%
7118      }%
7119      {%
```

First use singular form, all caps:

```
7120      \mfirstucMakeUppercase
7121      {\glxtrfullformat{\glslabel}{\glsinsert}}%
7122      }%
7123      }%
7124      }%
7125      }%
7126      {%
```

User supplied text.

```
7127      \glscustomtext
7128      }%
7129 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7130 \newcommand*{\glxtrsubsequentfmt}[2]{%
7131   \glsabbrvfont{\glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
```

```

7132 \ifglxtrinsertinside \else#2\fi
7133 }
7134 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural no case change).

```

7135 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7136 \glssabrvfont{\glssaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7137 \ifglxtrinsertinside \else#2\fi
7138 }
7139 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```

7140 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7141 \glssabrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
7142 \ifglxtrinsertinside \else#2\fi
7143 }
7144 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```

7145 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7146 \glssabrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7147 \ifglxtrinsertinside \else#2\fi
7148 }
7149 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

breivationstyle

```

7150 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7151 \ifcsundef{@glssabrv@dispstyle@setup@#2}
7152 {%
7153 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7154 }%
7155 {%

```

Have abbreviations already been defined for this category?

```

7156 \ifcsstring{@glssabrv@current@#1}{#2}%
7157 {%

```

Style already set.

```

7158 }%
7159 {%
7160 \def\@glxtr@dostylewarn{%
7161 \glssforeachincategory{#1}{\@glss@type}{\@glss@label}%
7162 {%
7163 \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7164 style has been switched \MessageBreak
7165 for category ‘#1’, \MessageBreak
7166 but there have already been entries \MessageBreak

```

```

7167         defined for this category. Unwanted \MessageBreak
7168         side-effects may result}}}%
7169     \@endfortrue
7170 }%
7171 \@glsxtr@dostylewarn

```

Set up the style for the given category.

```

7172     \csdef{@glsabbrv@current@#1}{#2}%
7173     \glsxtr@applyabbrvstyle{#2}%
7174 }%
7175 }%
7176 }

```

applyabbrvstyle Apply the abbreviation style without existence check.

```

7177 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7178   \csuse{@glsabbrv@dispstyle@setup@#1}%
7179   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7180 }

```

r@applyabbrvfmt Only apply the style formats.

```

7181 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7182   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7183 }

```

breivationstyle This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```

7184 \newcommand*{\newabbreviationstyle}[3]{%
7185   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7186   {%
7187     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7188       defined}}}%
7189   }%
7190   {%
7191     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7192     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7193     #2}%
7194     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7195     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7196     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7197     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7198     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7199     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7200     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt

```

```

7201 \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7202 \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7203 #3}%
7204 }%
7205 }

```

breivationstyle

```

7206 \newcommand*{\renewabbreviationstyle}[3]{%
7207 \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7208 {%
7209 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7210 }%
7211 {%
7212 \csdef{@glsabbrv@dispstyle@setup@#1}{%
    Initialise hook to do nothing. The style may change this.
7213 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7214 #2}%
7215 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
    Assume in-line form is the same as first use. The style may change this.
7216 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7217 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7218 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7219 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7220 #3}%
7221 }%
7222 }

```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

7223 \newcommand*{\letabbreviationstyle}[2]{%
7224 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7225 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7226 }

```

ecated@abbrstyle `\@glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```

7227 \newcommand*{\@glxtr@deprecated@abbrstyle}[2]{%
7228 \csdef{@glsabbrv@dispstyle@setup@#1}{%
7229 \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7230 \csuse{@glsabbrv@dispstyle@setup@#2}%
7231 }%
7232 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7233 }

```

ecatedAbbrStyle Generate warning for deprecated style use.

```
7234 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7235   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7236   use ‘#2’ instead}%
7237 }
```

eAbbrStyleSetup

```
7238 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7239   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7240   {%
7241     \PackageError{glossaries-extra}%
7242     {Unknown abbreviation style definitions ‘#1’}{}%
7243   }%
7244   {%
7245     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7246   }%
7247 }
```

seAbbrStyleFmts

```
7248 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7249   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7250   {%
7251     \PackageError{glossaries-extra}%
7252     {Unknown abbreviation style formats ‘#1’}{}%
7253   }%
7254   {%
7255     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7256   }%
7257 }
```

1.7.2 Predefined Styles (Default Font)

Define some common styles. These will set the first, firstplural, text and plural keys, even if the regular attribute isn't set to "true". If this attribute is set, commands like `\gls` will use them as per a regular entry, otherwise those keys will be ignored unless explicitly invoked by the user with commands like `\glsfirst`. In order for the first letter uppercase versions to work correctly, `\glsxtrfullformat` needs to be expanded when those keys are set. The final optional argument of `\glsfirst` will behave differently to the final optional argument of `\gls` with some styles.

xtrinsertinside Switch to determine if the insert text should be inside or outside the font changing command.
The default is outside.

```
7258 \newif\ifglsxtrinsertinside
7259 \glsxtrinsertinsidefalse
```

trlongshortname

```
7260 \newcommand*{\glsxtrlongshortname}{%
```

```

7261 \protect\glsabbrvfont{\the\glsshorttok}%
7262 }

```

long-short

```

7263 \newabbreviationstyle{long-short}%
7264 {%
7265   \renewcommand*{\CustomAbbreviationFields}{%
7266     name={\glsxtrlongshortname},
7267     sort={\the\glsshorttok},
7268     first={\protect\glsfirstlongfont{\the\glslongtok}%
7269       \protect\glsxtrfullsep{\the\glslabeltok}%
7270       \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7271     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7272       \protect\glsxtrfullsep{\the\glslabeltok}%
7273       \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7274     plural={\protect\glsabbrvfont{\the\glsshortpltok}}},%
7275     description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7276 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7277   \glsattribute{\the\glslabeltok}{regular}%
7278   {%
7279     \glssetattribute{\the\glslabeltok}{regular}{false}%
7280   }%
7281   {}%
7282 }%
7283 }%
7284 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7285 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7286 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7287 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7288 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7289 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7290 \renewcommand*{\glsxtrfullformat}[2]{%
7291   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7292   \ifglsxtrininsertinside\else##2\fi
7293   \glsxtrfullsep{##1}%
7294   \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7295 }%
7296 \renewcommand*{\glsxtrfullplformat}[2]{%
7297   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7298   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7299   \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7300 }%
7301 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

7302 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7303 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7304 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7305 }%
7306 \renewcommand*{\Glsxtrfullplformat}[2]{%
7307 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7308 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7309 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7310 }%
7311 }

```

Set this as the default style for general abbreviations:

```
7312 \setabbreviationstyle{long-short}
```

ngshortdescsort

```

7313 \newcommand*{\glxtrlongshortdescsort}{%
7314 \expandonce\glxtrorglong\space (\expandonce\glxtrorgshort)%
7315 }

```

ngshortdescname

```

7316 \newcommand*{\glxtrlongshortdescname}{%
7317 \protect\glslongfont{\the\glslongtok}
7318 \glxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}}%
7319 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7320 \newabbreviationstyle{long-short-desc}%
7321 {%
7322 \renewcommand*{\CustomAbbreviationFields}{%
7323 name={\glxtrlongshortdescname},
7324 sort={\glxtrlongshortdescsort},%
7325 first={\protect\glsfirstlongfont{\the\glslongtok}%
7326 \protect\glxtrfullsep{\the\glslabeltok}%
7327 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7328 firstplural={\protect\glsfirstlongfont{\the\glslongtok}%
7329 \protect\glxtrfullsep{\the\glslabeltok}%
7330 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7331 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7332 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7333 }%

```

Unset the regular attribute if it has been set.

```

7334 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7335 \glshasattribute{\the\glslabeltok}{regular}%
7336 {%
7337 \glssetattribute{\the\glslabeltok}{regular}{false}%
7338 }%

```

```

7339    {}%
7340  }%
7341 }%
7342 {%
7343   \GlsXtrUseAbbrStyleFmts{long-short}%
7344 }

```

trshortlongname

```

7345 \newcommand*{\glxtrshortlongname}{%
7346   \protect\glsabbrvfont{\the\glsshorttok}%
7347 }

```

short-long Short form followed by long form in parenthesis on first use.

```

7348 \newabbreviationstyle{short-long}%
7349 {%
7350   \renewcommand*{\CustomAbbreviationFields}{%
7351     name={\glxtrshortlongname},
7352     sort={\the\glsshorttok},
7353     description={\the\glslongtok},%
7354     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7355       \protect\glxtrfullsep{\the\glslabeltok}%
7356       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7357     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7358       \protect\glxtrfullsep{\the\glslabeltok}%
7359       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7360     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}

```

Unset the regular attribute if it has been set.

```

7361 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7362   \glshasattribute{\the\glslabeltok}{regular}%
7363   {%
7364     \glissetattribute{\the\glslabeltok}{regular}{false}%
7365   }%
7366   {}%
7367 }%
7368 }%
7369 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7370 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7371 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7372 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7373 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7374 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7375 \renewcommand*{\glxtrfullformat}[2]{%
7376   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7377   \ifglxtrininsertinside\else##2\fi

```

```

7378 \glxtrfullsep{##1}%
7379 \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7380 }%
7381 \renewcommand*{\glxtrfullplformat}[2]{%
7382 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7383 \ifglxtrininsertinside\else##2\fi
7384 \glxtrfullsep{##1}%
7385 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7386 }%
7387 \renewcommand*{\Glsxtrfullformat}[2]{%
7388 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7389 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7390 \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7391 }%
7392 \renewcommand*{\Glsxtrfullplformat}[2]{%
7393 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7394 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7395 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7396 }%
7397 }

```

ortlongdescsort

```
7398 \newcommand*{\glxtrshortlongdescsort}{\the\glssshorttok}
```

ortlongdescname

```

7399 \newcommand*{\glxtrshortlongdescname}{%
7400 \protect\glsabbrvfont{\the\glssshorttok}
7401 \glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
7402 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7403 \newabbreviationstyle{short-long-desc}%
7404 {%
7405 \renewcommand*{\CustomAbbreviationFields}{%
7406 name={\glxtrshortlongdescname},
7407 sort={\glxtrshortlongdescsort},
7408 first={\protect\glsfirstabbrvfont{\the\glssshorttok}}%
7409 \protect\glxtrfullsep{\the\glslabeltok}%
7410 \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7411 firstplural={\protect\glsfirstabbrvfont{\the\glssshortpltok}}%
7412 \protect\glxtrfullsep{\the\glslabeltok}%
7413 \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7414 text={\protect\glsabbrvfont{\the\glssshorttok}},%
7415 plural={\protect\glsabbrvfont{\the\glssshortpltok}}}%
7416 }%

```

Unset the regular attribute if it has been set.

```

7417 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7418   \glshasattribute{\the\glslabeltok}{regular}%
7419   {%
7420     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7421     }%
7422   }%
7423 }%
7424 }%
7425 {%
7426   \GlsXtrUseAbbrStyleFmts{short-long}%
7427 }

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7428 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7429 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`\glsextrabrvfootnote` `\glsextrabrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

7430 \newcommand*{\glsextrabrvfootnote}[2]{\footnote{#2}}

```

`\glsextrfootnotename`

```

7431 \newcommand*{\glsextrfootnotename}{%
7432   \protect\glsabbrvfont{\the\glsshorttok}%
7433 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

7434 \newabbreviationstyle{footnote}%
7435 {%
7436   \renewcommand*{\CustomAbbreviationFields}{%
7437     name={\glsextrfootnotename},
7438     sort={\the\glsshorttok},
7439     description={\the\glslongtok},%
7440     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7441       \protect\glsextrabrvfootnote{\the\glslabeltok}%
7442       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7443     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7444       \protect\glsextrabrvfootnote{\the\glslabeltok}%
7445       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

```

7446 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

7447 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7448   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7449   \glsattribute{\the\glslabeltok}{regular}%
7450   {%
7451     \glssetattribute{\the\glslabeltok}{regular}{false}%
7452   }%
7453   {}%
7454 }%
7455 }%
7456 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7457 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7458 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7459 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7460 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7461 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7462 \renewcommand*{\glsxtrfullformat}[2]{%
7463   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7464   \ifglsxtrininsertinside\else##2\fi
7465   \protect\glsxtrabbrvfootnote{##1}%
7466   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7467 }%
7468 \renewcommand*{\glsxtrfullplformat}[2]{%
7469   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7470   \ifglsxtrininsertinside\else##2\fi
7471   \protect\glsxtrabbrvfootnote{##1}%
7472   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7473 }%
7474 \renewcommand*{\Glsxtrfullformat}[2]{%
7475   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7476   \ifglsxtrininsertinside\else##2\fi
7477   \protect\glsxtrabbrvfootnote{##1}%
7478   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7479 }%
7480 \renewcommand*{\Glsxtrfullplformat}[2]{%
7481   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7482   \ifglsxtrininsertinside\else##2\fi
7483   \protect\glsxtrabbrvfootnote{##1}%
7484   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7485 }%

```

The first use full form and the inline full form use the short (long) style.

```

7486 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7487   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%

```

```

7488     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7489     \glxtrparen{\glsfirstlongfootnotefont{\glaccesslong{##1}}}%
7490 }%
7491 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7492     \glsfirstabbrvfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7493     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7494     \glxtrparen{\glsfirstlongfootnotefont{\glaccesslongpl{##1}}}%
7495 }%
7496 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7497     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7498     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7499     \glxtrparen{\glsfirstlongfootnotefont{\glaccesslong{##1}}}%
7500 }%
7501 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7502     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7503     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7504     \glxtrparen{\glsfirstlongfootnotefont{\glaccesslongpl{##1}}}%
7505 }%
7506 }

```

short-footnote

```

7507 \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.

```

7508 \newabbreviationstyle{postfootnote}%
7509 {%
7510     \renewcommand*{\CustomAbbreviationFields}{%
7511         name={\glxtrfootnotename},
7512         sort={\the\glsshorttok},
7513         description={\the\glslongtok},%
7514         first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7515         firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7516         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.

```

7517 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7518     \csdef{glxtrpostlink\glscategorylabel}{%
7519         \glxtrifwasfirstuse
7520         {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7521         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7522         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7523     }%

```

```

7524     {}%
7525   }%
7526   \glshasattribute{\the\glslabeltok}{regular}%
7527   {%
7528     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7529   }%
7530   {}%
7531 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7532 \renewcommand*{\glxtrsetupfulldefs}{%
7533   \let\glxtrifwasfirstuse\@secondoftwo
7534 }%
7535 }%
7536 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7537 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7538 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
7539 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
7540 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongfootnotefont{##1}}%
7541 \renewcommand*{\glsslongfont}[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

7542 \renewcommand*{\glxtrfullformat}[2]{%
7543   \glssfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7544   \ifglxtrininsertinside\else##2\fi
7545 }%
7546 \renewcommand*{\glxtrfullplformat}[2]{%
7547   \glssfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7548   \ifglxtrininsertinside\else##2\fi
7549 }%
7550 \renewcommand*{\Glsxtrfullformat}[2]{%
7551   \glssfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7552   \ifglxtrininsertinside\else##2\fi
7553 }%
7554 \renewcommand*{\Glsxtrfullplformat}[2]{%
7555   \glssfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7556   \ifglxtrininsertinside\else##2\fi
7557 }%

```

The first use full form and the inline full form use the short (long) style.

```

7558 \renewcommand*{\glxtrininlinefullformat}[2]{%
7559   \glssfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7560   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7561   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
7562 }%
7563 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7564   \glssfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7565   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7566 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7567 }%
7568 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7569 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
7570 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7571 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7572 }%
7573 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7574 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
7575 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7576 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7577 }%
7578 }

```

rt-postfootnote

```
7579 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```

7580 \newcommand*{\glsxtrshortnolongname}{%
7581 \protect\glsabbrvfont{\the\glsshorttok}%
7582 }

```

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.

```

7583 \newabbreviationstyle{short}%
7584 {%
7585 \renewcommand*{\CustomAbbreviationFields}{%
7586 name={\glsxtrshortnolongname},
7587 sort={\the\glsshorttok},
7588 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7589 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7590 text={\protect\glsabbrvfont{\the\glsshorttok}},
7591 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7592 description={\the\glslongtok}}%
7593 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7594 \glssetattribute{\the\glslabeltok}{regular}{true}}%
7595 }%
7596 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7597 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7598 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7599 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7600 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7601 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7602 \renewcommand*{\glxstrinlinefullformat}[2]{%
7603   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7604   \ifglxstrinsertinside##2\fi}%
7605   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
7606   \glxstrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7607 }%
7608 \renewcommand*{\glxstrinlinefullplformat}[2]{%
7609   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7610   \ifglxstrinsertinside##2\fi}%
7611   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
7612   \glxstrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7613 }%
7614 \renewcommand*{\Glsxstrinlinefullformat}[2]{%
7615   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7616   \ifglxstrinsertinside##2\fi}%
7617   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
7618   \glxstrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7619 }%
7620 \renewcommand*{\Glsxstrinlinefullplformat}[2]{%
7621   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7622   \ifglxstrinsertinside##2\fi}%
7623   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
7624   \glxstrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7625 }%

```

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.

```

7626 \renewcommand*{\glxstrfullformat}[2]{%
7627   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxstrinsertinside##2\fi}%
7628   \ifglxstrinsertinside\else##2\fi
7629 }%
7630 \renewcommand*{\glxstrfullplformat}[2]{%
7631   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxstrinsertinside##2\fi}%
7632   \ifglxstrinsertinside\else##2\fi
7633 }%
7634 \renewcommand*{\Glsxstrfullformat}[2]{%
7635   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxstrinsertinside##2\fi}%
7636   \ifglxstrinsertinside\else##2\fi
7637 }%
7638 \renewcommand*{\Glsxstrfullplformat}[2]{%
7639   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxstrinsertinside##2\fi}%
7640   \ifglxstrinsertinside\else##2\fi
7641 }%
7642 }

```

Set this as the default style for acronyms:

```

7643 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7644 \letabbreviationstyle{short-nolong}{short}

```

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```
7645 \newabbreviationstyle{short-nolong-noreg}%
7646 {%
7647   \GlsXtrUseAbbrStyleSetup{short-nolong}%

   Unset the regular attribute if it has been set.
7648   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7649     \glshasattribute{\the\glslabeltok}{regular}%
7650     {%
7651       \glissetattribute{\the\glslabeltok}{regular}{false}%
7652     }%
7653   }%
7654 }%
7655 }%
7656 {%
7657   \GlsXtrUseAbbrStyleFmts{short-nolong}%
7658 }
```

trshortdescname

```
7659 \newcommand*{\glxtrshortdescname}{%
7660   \protect\glsabbrvfont{\the\glsshorttok}%
7661 }
```

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.

```
7662 \newabbreviationstyle{short-desc}%
7663 {%
7664   \renewcommand*{\CustomAbbreviationFields}{%
7665     name={\glxtrshortdescname},
7666     sort={\the\glsshorttok},
7667     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7668     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7669     text={\protect\glsabbrvfont{\the\glsshorttok}},
7670     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7671     description={\the\glslongtok}}%
7672   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7673     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7674 }%
7675 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7676 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7677 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7678 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7679 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7680 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
7681 \renewcommand*{\glxtrinlinefullformat}[2]{%
7682   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%

```

```

7683     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7684     \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7685 }%
7686 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7687     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7688     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7689     \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7690 }%
7691 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7692     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7693     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7694     \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7695 }%
7696 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7697     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7698     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7699     \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7700 }%

```

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.

```

7701 \renewcommand*{\glxtrfullformat}[2]{%
7702     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7703     \ifglxtrinsertinside\else##2\fi
7704 }%
7705 \renewcommand*{\glxtrfullplformat}[2]{%
7706     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7707     \ifglxtrinsertinside\else##2\fi
7708 }%
7709 \renewcommand*{\Glsxtrfullformat}[2]{%
7710     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7711     \ifglxtrinsertinside\else##2\fi
7712 }%
7713 \renewcommand*{\Glsxtrfullplformat}[2]{%
7714     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7715     \ifglxtrinsertinside\else##2\fi
7716 }%
7717 }

```

ort-nolong-desc

```
7718 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7719 \newabbreviationstyle{short-nolong-desc-noreg}%
7720 {%
7721     \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7722 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7723     \glshasattribute{\the\glslabeltok}{regular}%

```

```

7724    {%
7725        \glssetattribute{\the\glslabeltok}{regular}{false}%
7726    }%
7727    {}%
7728 }%
7729 }%
7730 {%
7731     \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7732 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7733 \newabbreviationstyle{nolong-short}%
7734 {%
7735     \GlsXtrUseAbbrStyleSetup{short-nolong}%
7736 }%
7737 {%
7738     \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7739 \renewcommand*{\glxtrinlinefullformat}[2]{%
7740     \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7741     \ifglxtrininsertinside##2\fi}%
7742 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7743 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7744 }%
7745 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7746     \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7747     \ifglxtrininsertinside##2\fi}%
7748 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7749 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7750 }%
7751 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7752     \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7753     \ifglxtrininsertinside##2\fi}%
7754 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7755 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7756 }%
7757 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7758     \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7759     \ifglxtrininsertinside##2\fi}%
7760 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7761 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7762 }%
7763 }

```

ong-short-noreg Like nolong-short but doesn't set the regular attribute.

```

7764 \newabbreviationstyle{nolong-short-noreg}%
7765 {%
7766     \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7767 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7768   \glshasattribute{\the\glslabeltok}{regular}%
7769   {%
7770     \glissetattribute{\the\glslabeltok}{regular}{false}%
7771   }%
7772   {}%
7773 }%
7774 }%
7775 {%
7776   \GlsXtrUseAbbrStyleFmts{nolong-short}%
7777 }

```

noshortdescname

```

7778 \newcommand*{\glsxtrlongnoshortdescname}{%
7779   \protect\glslongfont{\the\glslongtok}%
7780 }

```

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.

```

7781 \newabbreviationstyle{long-desc}%
7782 {%
7783   \renewcommand*{\CustomAbbreviationFields}{%
7784     name={\glsxtrlongnoshortdescname},
7785     sort={\the\glslongtok},
7786     first={\protect\glslongfont{\the\glslongtok}},
7787     firstplural={\protect\glslongfont{\the\glslongpltok}},
7788     text={\glslongfont{\the\glslongtok}},
7789     plural={\glslongfont{\the\glslongpltok}}%
7790   }%
7791   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7792     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7793 }%
7794 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7795 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7796 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
7797 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7798 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7799 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7800 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
7801   \glslongfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
7802   \ifglsxtrininsertinside \else##2\fi
7803 }%
7804 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%

```

```

7805 \glslongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7806 \ifglxtrinsertinside \else##2\fi
7807 }%
7808 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7809 \glslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7810 \ifglxtrinsertinside \else##2\fi
7811 }%
7812 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7813 \glslongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7814 \ifglxtrinsertinside \else##2\fi
7815 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7816 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7817 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7818 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7819 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7820 }%
7821 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7822 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7823 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7824 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7825 }%
7826 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7827 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7828 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7829 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7830 }%
7831 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7832 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7833 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7834 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7835 }%

```

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.

```

7836 \renewcommand*{\glsxtrfullformat}[2]{%
7837 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7838 \ifglxtrinsertinside\else##2\fi
7839 }%
7840 \renewcommand*{\glsxtrfullplformat}[2]{%
7841 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7842 \ifglxtrinsertinside\else##2\fi
7843 }%
7844 \renewcommand*{\Glsxtrfullformat}[2]{%
7845 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7846 \ifglxtrinsertinside\else##2\fi
7847 }%
7848 \renewcommand*{\Glsxtrfullplformat}[2]{%
7849 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%

```

```

7850     \ifglxtrinsertinside\else##2\fi
7851   }%
7852 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```
7853 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```

7854 \newabbreviationstyle{long-noshort-desc-noreg}%
7855 {%
7856   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
7857   Unset the regular attribute if it has been set.
7857   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7858     \glshasattribute{\the\glslabeltok}{regular}%
7859     {%
7860       \glsssetattribute{\the\glslabeltok}{regular}{false}%
7861     }%
7862     {}%
7863   }%
7864 }%
7865 {%
7866   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7867 }

```

longnoshortname

```

7868 \newcommand*{\glxtrlongnoshortname}{%
7869   \protect\glsabbrvfont{\the\glsshorttok}%
7870 }

```

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.

```

7871 \newabbreviationstyle{long}%
7872 {%
7873   \renewcommand*{\CustomAbbreviationFields}{%
7874     name={\glxtrlongnoshortname},
7875     sort={\the\glsshorttok},
7876     first={\protect\glsfirstlongfont{\the\glslongtok}},
7877     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7878     text={\glsfont{\the\glslongtok}},
7879     plural={\glsfont{\the\glslongpltok}},%
7880     description={\the\glslongtok}%
7881   }%
7882   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7883     \glsssetattribute{\the\glslabeltok}{regular}{true}}%
7884 }%
7885 {%
7886   \GlsXtrUseAbbrStyleFmts{long-desc}%
7887 }

```

`long-noshort` Provide a synonym that matches similar styles.

```
7888 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```
7889 \newabbreviationstyle{long-noshort-noreg}%
```

```
7890 {%
```

```
7891   \GlsXtrUseAbbrStyleSetup{long-noshort}%
```

Unset the regular attribute if it has been set.

```
7892   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
7893     \glshasattribute{\the\glslabeltok}{regular}%
```

```
7894     {%
```

```
7895       \glissetattribute{\the\glslabeltok}{regular}{false}%
```

```
7896     }%
```

```
7897   }%
```

```
7898 }%
```

```
7899 }%
```

```
7900 {%
```

```
7901   \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

```
7902 }
```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
7903 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7904 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
7905 \newcommand*{\glxsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`irstabbrvscfont` Added for consistent naming.

```
7906 \newcommand*{\glsfirstabbrvscfont}{\glxsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
7907 \newcommand*{\glsxtrscsuffix}{\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
7908 \newabbreviationstyle{long-short-sc}%
```

```
7909 {%
```

```
7910   \renewcommand*{\CustomAbbreviationFields}{%
```

```
7911     name={\glsxtrlongshortname},
```

```
7912     sort={\the\glsshorttok},
```

```
7913     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
```

```

7914 \protect\glxtrfullsep{\the\glslabeltok}%
7915 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7916 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7917 \protect\glxtrfullsep{\the\glslabeltok}%
7918 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7919 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}},%
7920 description={\the\glslongtok}}%
7921 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7922 \glsasattribute{\the\glslabeltok}{regular}}%
7923 {%
7924 \glsssetAttribute{\the\glslabeltok}{regular}{false}}%
7925 }%
7926 {}%
7927 }%
7928 }%
7929 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7930 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7931 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7932 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

7933 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7934 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7935 \renewcommand*{\glxtrfullformat}[2]{%
7936 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7937 \ifglxtrininsertinside\else##2\fi
7938 \glxtrfullsep{##1}%
7939 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7940 }%
7941 \renewcommand*{\glxtrfullplformat}[2]{%
7942 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7943 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7944 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7945 }%
7946 \renewcommand*{\Glsxtrfullformat}[2]{%
7947 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7948 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7949 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7950 }%
7951 \renewcommand*{\Glsxtrfullplformat}[2]{%
7952 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7953 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7954 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7955 }%
7956 }

```

g-short-sc-desc

```

7957 \newabbreviationstyle{long-short-sc-desc}%
7958 {%
7959   \renewcommand*{\CustomAbbreviationFields}{%
7960     name={\glxtrlongshortdescname},
7961     sort={\glxtrlongshortdescsort},%
7962     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7963       \protect\glxtrfullsep{\the\glslabeltok}%
7964       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7965     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7966       \protect\glxtrfullsep{\the\glslabeltok}%
7967       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7968     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7969     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7970   }%

```

Unset the regular attribute if it has been set.

```

7971 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7972   \glshasattribute{\the\glslabeltok}{regular}%
7973   {%
7974     \glissetattribute{\the\glslabeltok}{regular}{false}%
7975   }%
7976   {}%
7977 }%
7978 }%
7979 {%

```

As long-short-sc style:

```

7980 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7981 }

```

Now the short (long) version

```

7982 \newabbreviationstyle{short-sc-long}%
7983 {%
7984   \renewcommand*{\CustomAbbreviationFields}{%
7985     name={\glxtrshortlongname},
7986     sort={\the\glsshorttok},
7987     description={\the\glslongtok},%
7988     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7989       \protect\glxtrfullsep{\the\glslabeltok}%
7990       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7991     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7992       \protect\glxtrfullsep{\the\glslabeltok}%
7993       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7994     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7995 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7996   \glshasattribute{\the\glslabeltok}{regular}%
7997   {%
7998     \glissetattribute{\the\glslabeltok}{regular}{false}%
7999   }%

```

```

8000    {}%
8001  }%
8002 }%
8003 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8004  \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8005  \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8006  \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvscfont{##1}}%
8007  \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8008  \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8009  \renewcommand*{\glxtrfullformat}[2]{%
8010    \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8011    \ifglxtrininsertinside\else##2\fi
8012    \glxtrfullsep{##1}%
8013    \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
8014  }%
8015  \renewcommand*{\glxtrfullplformat}[2]{%
8016    \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8017    \ifglxtrininsertinside\else##2\fi
8018    \glxtrfullsep{##1}%
8019    \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8020  }%
8021  \renewcommand*{\Glsxtrfullformat}[2]{%
8022    \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8023    \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8024    \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
8025  }%
8026  \renewcommand*{\Glsxtrfullplformat}[2]{%
8027    \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8028    \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8029    \glxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8030  }%
8031 }

```

As before but user provides description

```

8032 \newabbreviationstyle{short-sc-long-desc}%
8033 {%
8034  \renewcommand*{\CustomAbbreviationFields}{%
8035    name={\glxtrshortlongdescname},
8036    sort={\glxtrshortlongdescsort},
8037    first={\protect\glssfirstabbrvscfont{\the\glssshorttok}}%
8038    \protect\glxtrfullsep{\the\glsslabeltok}}%
8039    \glxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongtok}}},%
8040    firstplural={\protect\glssfirstabbrvscfont{\the\glssshortpltok}}%
8041    \protect\glxtrfullsep{\the\glsslabeltok}}%
8042    \glxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongpltok}}},%
8043    text={\protect\glssabbrvscfont{\the\glssshorttok}},%

```

```

8044 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8045 }%

```

Unset the regular attribute if it has been set.

```

8046 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8047   \glsattribute{\the\glslabeltok}{regular}%
8048   {%
8049     \glssetattribute{\the\glslabeltok}{regular}{false}%
8050   }%
8051   {}%
8052 }%
8053 }%
8054 {%

```

As short-sc-long style:

```

8055 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8056 }

```

short-sc

```

8057 \newabbreviationstyle{short-sc}%
8058 {%
8059   \renewcommand*{\CustomAbbreviationFields}{%
8060     name={\glsxtrshortnolongname},
8061     sort={\the\glsshorttok},
8062     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8063     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8064     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8065     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8066     description={\the\glslongtok}}%
8067   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8068     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8069 }%
8070 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8071 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8072 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8073 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8074 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8075 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8076 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8077   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8078   \ifglsxtrininsertinside##2\fi}%
8079   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8080   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8081 }%
8082 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8083   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8084   \ifglsxtrininsertinside##2\fi}%

```

```

8085 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8086 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8087 }%

8088 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8089 \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
8090 \ifglxtrinsertinside##2\fi}%
8091 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8092 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8093 }%

8094 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8095 \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
8096 \ifglxtrinsertinside##2\fi}%
8097 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8098 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8099 }%

```

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.

```

8100 \renewcommand*{\glxtrfullformat}[2]{%
8101 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8102 \ifglxtrinsertinside\else##2\fi
8103 }%

8104 \renewcommand*{\glxtrfullplformat}[2]{%
8105 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8106 \ifglxtrinsertinside\else##2\fi
8107 }%

8108 \renewcommand*{\Glsxtrfullformat}[2]{%
8109 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8110 \ifglxtrinsertinside\else##2\fi
8111 }%

8112 \renewcommand*{\Glsxtrfullplformat}[2]{%
8113 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8114 \ifglxtrinsertinside\else##2\fi
8115 }%
8116 }

```

short-sc-nolong

```

8117 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8118 \newabbreviationstyle{short-sc-desc}%
8119 {%
8120 \renewcommand*{\CustomAbbreviationFields}{%
8121 name={\glxtrshortdescname},
8122 sort={\the\glssshorttok},
8123 first={\protect\glsfirstabbrvscfont{\the\glssshorttok}},
8124 firstplural={\protect\glsfirstabbrvscfont{\the\glssshortpltok}},
8125 text={\protect\glssabbrvscfont{\the\glssshorttok}},

```

```

8126 plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8127 description={\the\glslongtok}}%
8128 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8129 \glssetattribute{\the\glslabeltok}{regular}{true}}%
8130 }%
8131 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8132 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8133 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8134 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8135 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8136 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8137 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8138 \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8139 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8140 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8141 }%
8142 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8143 \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8144 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8145 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8146 }%
8147 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8148 \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8149 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8150 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8151 }%
8152 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8153 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8154 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8155 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8156 }%

```

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.

```

8157 \renewcommand*{\glsxtrfullformat}[2]{%
8158 \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8159 \ifglsxtrininsertinside\else##2\fi
8160 }%
8161 \renewcommand*{\glsxtrfullplformat}[2]{%
8162 \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8163 \ifglsxtrininsertinside\else##2\fi
8164 }%
8165 \renewcommand*{\Glsxtrfullformat}[2]{%
8166 \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8167 \ifglsxtrininsertinside\else##2\fi
8168 }%
8169 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

8170 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8171 \ifglxtrinsertinside\else##2\fi
8172 }%
8173 }

```

-sc-nolong-desc

```

8174 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

8175 \newabbreviationstyle{nolong-short-sc}%
8176 {%
8177 \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8178 }%
8179 {%
8180 \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8181 \renewcommand*{\glxtrinlinefullformat}[2]{%
8182 \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8183 \ifglxtrinsertinside##2\fi}%
8184 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8185 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8186 }%
8187 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8188 \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8189 \ifglxtrinsertinside##2\fi}%
8190 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8191 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8192 }%
8193 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8194 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8195 \ifglxtrinsertinside##2\fi}%
8196 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8197 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8198 }%
8199 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8200 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8201 \ifglxtrinsertinside##2\fi}%
8202 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8203 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8204 }%
8205 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

8206 \newabbreviationstyle{long-noshort-sc}%
8207 {%
8208 \renewcommand*{\CustomAbbreviationFields}{%
8209 name={\glxtrlongnoshortname},

```

```

8210     sort={\the\glsshorttok},
8211     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8212     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8213     text={\protect\glslongdefaultfont{\the\glslongtok}},
8214     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8215     description={\the\glslongtok}%
8216 }%
8217 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8218   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8219 }%
8220 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8221 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8222 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8223 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8224 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8225 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8226 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8227   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8228   \ifglsxtrininsertinside \else##2\fi
8229 }%
8230 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8231   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8232   \ifglsxtrininsertinside \else##2\fi
8233 }%
8234 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8235   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8236   \ifglsxtrininsertinside \else##2\fi
8237 }%
8238 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8239   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8240   \ifglsxtrininsertinside \else##2\fi
8241 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8242 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8243   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8244   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8245   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8246 }%
8247 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8248   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8249   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8250   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8251 }%
8252 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8253   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8254   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

8255 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8256 }%
8257 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8258 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8259 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8260 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8261 }%

```

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.

```

8262 \renewcommand*{\glxtrfullformat}[2]{%
8263 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8264 \ifglxtrinsertinside\else##2\fi
8265 }%
8266 \renewcommand*{\glxtrfullplformat}[2]{%
8267 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8268 \ifglxtrinsertinside\else##2\fi
8269 }%
8270 \renewcommand*{\Glsxtrfullformat}[2]{%
8271 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8272 \ifglxtrinsertinside\else##2\fi
8273 }%
8274 \renewcommand*{\Glsxtrfullplformat}[2]{%
8275 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8276 \ifglxtrinsertinside\else##2\fi
8277 }%
8278 }

```

long-sc Backward compatibility:

```

8279 \@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`.

```

8280 \newabbreviationstyle{long-noshort-sc-desc}%
8281 {%
8282 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8283 }%
8284 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8285 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8286 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8287 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8288 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8289 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8290 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8291 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8292 \ifglxtrinsertinside \else##2\fi

```

```

8293 }%
8294 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8295   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8296   \ifglxtrinsertinside \else##2\fi
8297 }%
8298 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8299   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8300   \ifglxtrinsertinside \else##2\fi
8301 }%
8302 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8303   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8304   \ifglxtrinsertinside \else##2\fi
8305 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8306 \renewcommand*{\glxtrinlinefullformat}[2]{%
8307   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8308   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8309   \glxtrparen{\protect\glslfirstabbrvscfont{\glssaccessshort{##1}}}%
8310 }%
8311 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8312   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8313   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8314   \glxtrparen{\protect\glslfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8315 }%
8316 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8317   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8318   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8319   \glxtrparen{\protect\glslfirstabbrvscfont{\glssaccessshort{##1}}}%
8320 }%
8321 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8322   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8323   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8324   \glxtrparen{\protect\glslfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8325 }%

```

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.

```

8326 \renewcommand*{\glxtrfullformat}[2]{%
8327   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8328   \ifglxtrinsertinside\else##2\fi
8329 }%
8330 \renewcommand*{\glxtrfullplformat}[2]{%
8331   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8332   \ifglxtrinsertinside\else##2\fi
8333 }%
8334 \renewcommand*{\Glsxtrfullformat}[2]{%
8335   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8336   \ifglxtrinsertinside\else##2\fi
8337 }%

```

```

8338 \renewcommand*{\Glsxtrfullplformat}[2]{%
8339   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8340   \ifglxtrinsertinside\else##2\fi
8341 }%
8342 }

```

long-desc-sc Backward compatibility:

```

8343 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

8344 \newabbreviationstyle{short-sc-footnote}%
8345 {%
8346   \renewcommand*{\CustomAbbreviationFields}{%
8347     name={\glxtrfootnotename},
8348     sort={\the\glsshorttok},
8349     description={\the\glslongtok},%
8350     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8351       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8352       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8353     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8354       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8355       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8356     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.

```

8357 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8358   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8359   \glsattribute{\the\glslabeltok}{regular}%
8360   {%
8361     \glssetattribute{\the\glslabeltok}{regular}{false}%
8362   }%
8363   {}%
8364 }%
8365 }%
8366 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8367 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8368 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8369 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8370 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8371 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8372 \renewcommand*{\glxtrfullformat}[2]{%
8373   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8374   \ifglxtrinsertinside\else##2\fi
8375   \protect\glxtrabbrvfootnote{##1}%
8376   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%

```

```

8377 }%
8378 \renewcommand*{\glxtrfullplformat}[2]{%
8379   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8380   \ifglxtrininsertinside\else##2\fi
8381   \protect\glxtrabbrvfootnote{##1}%
8382   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8383 }%
8384 \renewcommand*{\Glsxtrfullformat}[2]{%
8385   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8386   \ifglxtrininsertinside\else##2\fi
8387   \protect\glxtrabbrvfootnote{##1}%
8388   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8389 }%
8390 \renewcommand*{\Glsxtrfullplformat}[2]{%
8391   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8392   \ifglxtrininsertinside\else##2\fi
8393   \protect\glxtrabbrvfootnote{##1}%
8394   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8395 }%

```

The first use full form and the inline full form use the short (long) style.

```

8396 \renewcommand*{\glxtrininlinefullformat}[2]{%
8397   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8398   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8399   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8400 }%
8401 \renewcommand*{\glxtrininlinefullplformat}[2]{%
8402   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8403   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8404   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8405 }%
8406 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
8407   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8408   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8409   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8410 }%
8411 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
8412   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8413   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8414   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8415 }%
8416 }

```

footnote-sc Backward compatibility:

```

8417 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8418 \newabbreviationstyle{short-sc-postfootnote}%
8419 {%
8420   \renewcommand*{\CustomAbbreviationFields}{%

```

```

8421   name={\glxtrfootnotename},
8422   sort={\the\glsshorttok},
8423   description={\the\glslongtok},%
8424   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8425   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8426   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.

```

8427   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8428     \csdef{glsxtrpostlink\glscategorylabel}{%
8429       \glxtrifwasfirstuse
8430       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8431         \glxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
8432         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8433     }%
8434   }%
8435 }%
8436 \glshasattribute{\the\glslabeltok}{regular}%
8437 {%
8438   \glssetattribute{\the\glslabeltok}{regular}{false}%
8439   }%
8440   }%
8441 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8442   \renewcommand*{\glxtrsetupfulldefs}{%
8443     \let\glxtrifwasfirstuse\@secondoftwo
8444   }%
8445 }%
8446 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8447   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8448   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8449   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8450   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8451   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8452   \renewcommand*{\glxtrfullformat}[2]{%
8453     \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8454     \ifglxtrininsertinside\else##2\fi
8455   }%
8456   \renewcommand*{\glxtrfullplformat}[2]{%
8457     \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8458     \ifglxtrininsertinside\else##2\fi

```

```

8459 }%
8460 \renewcommand*{\Glsxtrfullformat}[2]{%
8461   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8462   \ifglxtrinsertinside\else##2\fi
8463 }%
8464 \renewcommand*{\Glsxtrfullplformat}[2]{%
8465   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8466   \ifglxtrinsertinside\else##2\fi
8467 }%

```

The first use full form and the inline full form use the short (long) style.

```

8468 \renewcommand*{\glxtrinlinefullformat}[2]{%
8469   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8470   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8471   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8472 }%
8473 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8474   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8475   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8476   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8477 }%
8478 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8479   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8480   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8481   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8482 }%
8483 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8484   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8485   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8486   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8487 }%
8488 }

```

postfootnote-sc Backward compatibility:

```

8489 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the `relsize` package, which must be loaded by the user. These styles all use:

`\glxtrsmfont` Maintained for backward compatibility.

```

8490 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

```

`\glsabbrvsmfont` Added for consistent naming.

```

8491 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

```

`sxtrfirstsmfont` Maintained for backward compatibility.

```

8492 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

```

irstabbrvsmfont Added for consistent naming.

```
8493 \newcommand*{\glsfirstabbrvsmfont}{\glsxtrfirstsmfont}
```

and for the default short form suffix:

\glsxtrsmsuffix

```
8494 \newcommand*{\glsxtrsmsuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-sm

```
8495 \newabbreviationstyle{long-short-sm}%
8496 {%
8497   \renewcommand*{\CustomAbbreviationFields}{%
8498     name={\glsxtrlongshortname},
8499     sort={\the\glsshorttok},
8500     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8501       \protect\glsxtrfullsep{\the\glslabeltok}%
8502       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8503     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8504       \protect\glsxtrfullsep{\the\glslabeltok}%
8505       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8506     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8507     description={\the\glslongtok}}%
8508   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8509     \glsattribute{\the\glslabeltok}{regular}%
8510     {%
8511       \glssetattribute{\the\glslabeltok}{regular}{false}%
8512     }%
8513   }%
8514 }%
8515 }%
8516 {%
8517   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8518   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8519   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%

```

Use the default long fonts.

```
8520 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8521 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8522 \renewcommand*{\glsxtrfullformat}[2]{%
8523   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8524   \ifglsxtrininsertinside\else##2\fi
8525   \glsxtrfullsep{##1}%
8526   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8527 }%
8528 \renewcommand*{\glsxtrfullplformat}[2]{%
8529   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8530   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8531   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%

```

```

8532 }%
8533 \renewcommand*{\Glsxtrfullformat}[2]{%
8534   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8535   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8536   \glsxtrparen{\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8537 }%
8538 \renewcommand*{\Glsxtrfullplformat}[2]{%
8539   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8540   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8541   \glsxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8542 }%
8543 }

```

g-short-sm-desc

```

8544 \newabbreviationstyle{long-short-sm-desc}%
8545 {%
8546   \renewcommand*{\CustomAbbreviationFields}{%
8547     name={\glsxtrlongshortdescname},
8548     sort={\glsxtrlongshortdescsort},%
8549     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8550       \protect\glsxtrfullsep{\the\glslabeltok}%
8551       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8552     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8553       \protect\glsxtrfullsep{\the\glslabeltok}%
8554       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8555     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8556     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
8557 }%

```

Unset the regular attribute if it has been set.

```

8558 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8559   \glshasattribute{\the\glslabeltok}{regular}%
8560   {%
8561     \glissetattribute{\the\glslabeltok}{regular}{false}%
8562   }%
8563   {}%
8564 }%
8565 }%
8566 {%

```

As long-short-sm style:

```

8567   \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8568 }

```

short-sm-long Now the short (long) version

```

8569 \newabbreviationstyle{short-sm-long}%
8570 {%
8571   \renewcommand*{\CustomAbbreviationFields}{%
8572     name={\glsxtrshortlongname},
8573     sort={\the\glsshorttok},

```

```

8574   description={\the\glslongtok},%
8575   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8576     \protect\glsxtrfullsep{\the\glslabeltok}%
8577     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8578   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8579     \protect\glsxtrfullsep{\the\glslabeltok}%
8580     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8581   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8582   \renewcommand*\GlsXtrPostNewAbbreviation{%
8583     \glshasattribute{\the\glslabeltok}{regular}%
8584     {%
8585       \glssetattribute{\the\glslabeltok}{regular}{false}%
8586     }%
8587   }%
8588 }%
8589 }%
8590 {%
8591   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8592   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8593   \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
8594   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8595   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8596   \renewcommand*\glsxtrfullformat[2]{%
8597     \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8598     \ifglsxtrininsertinside\else##2\fi
8599     \glsxtrfullsep{##1}%
8600     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8601   }%
8602   \renewcommand*\glsxtrfullplformat[2]{%
8603     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8604     \ifglsxtrininsertinside\else##2\fi
8605     \glsxtrfullsep{##1}%
8606     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8607   }%
8608   \renewcommand*\Glsxtrfullformat[2]{%
8609     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8610     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8611     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8612   }%
8613   \renewcommand*\Glsxtrfullplformat[2]{%
8614     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8615     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8616     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8617   }%
8618 }

```

rt-sm-long-desc As before but user provides description

```
8619 \newabbreviationstyle{short-sm-long-desc}%
8620 {%
8621   \renewcommand*{\CustomAbbreviationFields}{%
8622     name={\glxtrshortlongdescname},
8623     sort={\glxtrshortlongdescsort},
8624     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8625       \protect\glxtrfullsep{\the\glslabeltok}%
8626       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8627     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8628       \protect\glxtrfullsep{\the\glslabeltok}%
8629       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8630     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8631     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8632   }%
```

Unset the regular attribute if it has been set.

```
8633   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8634     \glshasattribute{\the\glslabeltok}{regular}%
8635     {%
8636       \glissetattribute{\the\glslabeltok}{regular}{false}%
8637     }%
8638   }%
8639 }%
8640 }%
8641 {%
```

As short-sm-long style:

```
8642   \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8643 }
```

short-sm

```
8644 \newabbreviationstyle{short-sm}%
8645 {%
8646   \renewcommand*{\CustomAbbreviationFields}{%
8647     name={\glxtrshortnolongname},
8648     sort={\the\glsshorttok},
8649     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8650     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8651     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8652     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8653     description={\the\glslongtok}}%
8654   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8655     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8656 }%
8657 {%
8658   \renewcommand*{\glsabbrvfont[1]{\glsabbrvsmfont{##1}}}%
8659   \renewcommand*{\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}}%
8660   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrrmsuffix}%
8661   \renewcommand*{\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}}%

```

```
8662 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
8663 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8664   \protect\glsfirstabbrvsmfont{\glsaccesssshort{##1}}%
8665   \ifglsxtrinsertinside##2\fi}%
8666   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8667   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8668 }%
8669 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8670   \protect\glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}%
8671   \ifglsxtrinsertinside##2\fi}%
8672   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8673   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8674 }%

8675 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8676   \protect\glsfirstabbrvsmfont{\Glsaccesssshort{##1}}%
8677   \ifglsxtrinsertinside##2\fi}%
8678   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8679   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8680 }%
8681 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8682   \protect\glsfirstabbrvsmfont{\Glsaccesssshortpl{##1}}%
8683   \ifglsxtrinsertinside##2\fi}%
8684   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8685   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8686 }%
```

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.

```
8687 \renewcommand*{\glsxtrfullformat}[2]{%
8688   \glsfirstabbrvsmfont{\glsaccesssshort{##1}}\ifglsxtrinsertinside##2\fi}%
8689   \ifglsxtrinsertinside\else##2\fi
8690 }%
8691 \renewcommand*{\glsxtrfullplformat}[2]{%
8692   \glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
8693   \ifglsxtrinsertinside\else##2\fi
8694 }%
8695 \renewcommand*{\Glsxtrfullformat}[2]{%
8696   \glsfirstabbrvsmfont{\glsaccesssshort{##1}}\ifglsxtrinsertinside##2\fi}%
8697   \ifglsxtrinsertinside\else##2\fi
8698 }%
8699 \renewcommand*{\Glsxtrfullplformat}[2]{%
8700   \glsfirstabbrvsmfont{\glsaccesssshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
8701   \ifglsxtrinsertinside\else##2\fi
8702 }%
8703 }
```

short-sm-nolong

```
8704 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8705 \newabbreviationstyle{short-sm-desc}%
8706 {%
8707   \renewcommand*{\CustomAbbreviationFields}{%
8708     name={\glstrshortdesname},
8709     sort={\the\glsshorttok},
8710     first={\protect\glfirstabbrvsmfont{\the\glsshorttok}},
8711     firstplural={\protect\glfirstabbrvsmfont{\the\glsshortpltok}},
8712     text={\protect\glabbrvsmfont{\the\glsshorttok}},
8713     plural={\protect\glabbrvsmfont{\the\glsshortpltok}},
8714     description={\the\glslongtok}}%
8715   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8716     \glsetattribute{\the\glslabeltok}{regular}{true}}%
8717 }%
8718 {%
8719   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8720   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvsmfont{##1}}%
8721   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
8722   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8723   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8724 \renewcommand*{\glxtrinlinefullformat}[2]{%
8725   \glfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8726   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8727   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8728 }%
8729 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8730   \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8731   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8732   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8733 }%
8734 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8735   \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8736   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8737   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8738 }%
8739 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8740   \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8741   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8742   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8743 }%
```

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.

```
8744 \renewcommand*{\glxtrfullformat}[2]{%
8745   \glfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8746   \ifglxtrininsertinside\else##2\fi
```

```

8747 }%
8748 \renewcommand*{\glxtrfullplformat}[2]{%
8749   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8750   \ifglxtrininsertinside\else##2\fi
8751 }%
8752 \renewcommand*{\Glsxtrfullformat}[2]{%
8753   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8754   \ifglxtrininsertinside\else##2\fi
8755 }%
8756 \renewcommand*{\Glsxtrfullplformat}[2]{%
8757   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8758   \ifglxtrininsertinside\else##2\fi
8759 }%
8760 }

```

-sm-nolong-desc

```

8761 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8762 \newabbreviationstyle{nolong-short-sm}%
8763 {%
8764   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8765 }%
8766 {%
8767   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8768 \renewcommand*{\glxtrinlinefullformat}[2]{%
8769   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8770     \ifglxtrininsertinside##2\fi}%
8771   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8772   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8773 }%
8774 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8775   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8776     \ifglxtrininsertinside##2\fi}%
8777   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8778   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8779 }%
8780 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8781   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8782     \ifglxtrininsertinside##2\fi}%
8783   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8784   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8785 }%
8786 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8787   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8788     \ifglxtrininsertinside##2\fi}%
8789   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8790   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%

```

```

8791 }%
8792 }

```

`long-noshort-sm` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8793 \newabbreviationstyle{long-noshort-sm}%
8794 {%
8795   \renewcommand*{\CustomAbbreviationFields}{%
8796     name={\glxtrlongnoshortname},
8797     sort={\the\glsshorttok},
8798     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8799     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8800     text={\protect\glslongdefaultfont{\the\glslongtok}},
8801     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8802     description={\the\glslongtok}%
8803   }%
8804   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8805     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8806 }%
8807 {%
8808   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8809   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8810   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8811   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8812   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8813 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8814   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8815   \ifglxtrininsertinside \else##2\fi
8816 }%
8817 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8818   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8819   \ifglxtrininsertinside \else##2\fi
8820 }%
8821 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8822   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8823   \ifglxtrininsertinside \else##2\fi
8824 }%
8825 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8826   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8827   \ifglxtrininsertinside \else##2\fi
8828 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8829 \renewcommand*{\glxtrinlinefullformat}[2]{%
8830   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8831   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8832   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8833 }%

```

```

8834 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8835   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8836   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8837   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8838 }%
8839 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8840   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8841   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8842   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8843 }%
8844 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8845   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8846   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8847   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8848 }%

```

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.

```

8849 \renewcommand*{\glxtrfullformat}[2]{%
8850   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8851   \ifglxtrinsertinside\else##2\fi
8852 }%
8853 \renewcommand*{\glxtrfullplformat}[2]{%
8854   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8855   \ifglxtrinsertinside\else##2\fi
8856 }%
8857 \renewcommand*{\Glsxtrfullformat}[2]{%
8858   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8859   \ifglxtrinsertinside\else##2\fi
8860 }%
8861 \renewcommand*{\Glsxtrfullplformat}[2]{%
8862   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8863   \ifglxtrinsertinside\else##2\fi
8864 }%
8865 }

```

`long-sm` Backward compatibility:

```
8866 \@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`.

```

8867 \newabbreviationstyle{long-noshort-sm-desc}%
8868 {%
8869   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8870 }%
8871 {%
8872   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8873   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8874   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

```

8875 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8876 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8877 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8878   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
8879   \ifglsxtrinsertinside \else##2\fi
8880 }%
8881 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8882   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
8883   \ifglsxtrinsertinside \else##2\fi
8884 }%
8885 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8886   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
8887   \ifglsxtrinsertinside \else##2\fi
8888 }%
8889 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8890   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
8891   \ifglsxtrinsertinside \else##2\fi
8892 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8893 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8894   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8895   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8896   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8897 }%
8898 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8899   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8900   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8901   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8902 }%
8903 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8904   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8905   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8906   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8907 }%
8908 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8909   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8910   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8911   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8912 }%

```

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.

```

8913 \renewcommand*{\glsxtrfullformat}[2]{%
8914   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8915   \ifglsxtrinsertinside\else##2\fi
8916 }%
8917 \renewcommand*{\glsxtrfullplformat}[2]{%
8918   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%

```

```

8919 \ifglxtrinsertinside\else##2\fi
8920 }%
8921 \renewcommand*{\Glsxtrfullformat}[2]{%
8922 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8923 \ifglxtrinsertinside\else##2\fi
8924 }%
8925 \renewcommand*{\Glsxtrfullplformat}[2]{%
8926 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8927 \ifglxtrinsertinside\else##2\fi
8928 }%
8929 }

```

long-desc-sm Backward compatibility:

```

8930 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8931 \newabbreviationstyle{short-sm-footnote}%
8932 {%
8933 \renewcommand*{\CustomAbbreviationFields}{%
8934 name={\glxtrfootnotename},
8935 sort={\the\glsshorttok},
8936 description={\the\glslongtok},%
8937 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8938 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8939 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8940 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8941 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8942 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8943 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.

```

8944 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8945 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8946 \glsattribute{\the\glslabeltok}{regular}%
8947 {%
8948 \glssetattribute{\the\glslabeltok}{regular}{false}%
8949 }%
8950 {}%
8951 }%
8952 }%
8953 {%
8954 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8955 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8956 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8957 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8958 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8959 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

8960 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8961 \ifglxtrinsertinside\else##2\fi
8962 \protect\glxtrabbrvfootnote{##1}%
8963 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8964 }%
8965 \renewcommand*{\glxtrfullplformat}[2]{%
8966 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8967 \ifglxtrinsertinside\else##2\fi
8968 \protect\glxtrabbrvfootnote{##1}%
8969 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8970 }%
8971 \renewcommand*{\Glsxtrfullformat}[2]{%
8972 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8973 \ifglxtrinsertinside\else##2\fi
8974 \protect\glxtrabbrvfootnote{##1}%
8975 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8976 }%
8977 \renewcommand*{\Glsxtrfullplformat}[2]{%
8978 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8979 \ifglxtrinsertinside\else##2\fi
8980 \protect\glxtrabbrvfootnote{##1}%
8981 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8982 }%

```

The first use full form and the inline full form use the short (long) style.

```

8983 \renewcommand*{\glxtrininlinefullformat}[2]{%
8984 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8985 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8986 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8987 }%
8988 \renewcommand*{\glxtrininlinefullplformat}[2]{%
8989 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8990 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8991 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8992 }%
8993 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
8994 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8995 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8996 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8997 }%
8998 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
8999 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9000 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9001 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9002 }%
9003 }

```

footnote-sm Backward compatibility:

```

9004 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```
9005 \newabbreviationstyle{short-sm-postfootnote}%
9006 {%
9007   \renewcommand*{\CustomAbbreviationFields}{%
9008     name={\glstrfootnotename},
9009     sort={\the\glsshorttok},
9010     description={\the\glslongtok},%
9011     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9012     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9013     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
9014 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9015   \csdef{glstrpostlink\glscategorylabel}{%
9016     \glstrifwasfirstuse
9017     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
9018       \glstrdopostpunc{\protect\glstrabbrvfootnote{\glslabel}%
9019       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
9020     }%
9021   }%
9022 }%
9023 \glshasattribute{\the\glslabeltok}{regular}%
9024 {%
9025   \glssetattribute{\the\glslabeltok}{regular}{false}%
9026 }%
9027 {}%
9028 }%
```

The footnote needs to be suppressed in the inline form, so \glstrfull must set the first use switch off.

```
9029 \renewcommand*{\glstrsetupfulldefs}{%
9030   \let\glstrifwasfirstuse\@secondoftwo
9031 }%
9032 }%
9033 {%
9034   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9035   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9036   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
9037   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9038   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
9039 \renewcommand*{\glstrfullformat}[2]{%
9040   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9041   \ifglstrinsertinside\else##2\fi
9042 }%
```

```

9043 \renewcommand*\glstrfullplformat}[2]{%
9044   \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9045   \ifglstrinsertinside\else##2\fi
9046 }%
9047 \renewcommand*\Glsxtrfullformat}[2]{%
9048   \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9049   \ifglstrinsertinside\else##2\fi
9050 }%
9051 \renewcommand*\Glsxtrfullplformat}[2]{%
9052   \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9053   \ifglstrinsertinside\else##2\fi
9054 }%

```

The first use full form and the inline full form use the short (long) style.

```

9055 \renewcommand*\glstrinlinefullformat}[2]{%
9056   \glfirstabbrvsmfont{\glaccessshort{##1}\ifglstrinsertinside##2\fi}%
9057   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9058   \glstrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
9059 }%
9060 \renewcommand*\glstrinlinefullplformat}[2]{%
9061   \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9062   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9063   \glstrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
9064 }%
9065 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9066   \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9067   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9068   \glstrparen{\glfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9069 }%
9070 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9071   \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9072   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9073   \glstrparen{\glfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9074 }%
9075 }

```

postfootnote-sm Backward compatibility:

```

9076 \@glstr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9077 \newcommand*\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glfirstabbrvemfont`

```

9078 \newcommand*\glfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxxtremsuffix`

```
9079 \newcommand*{\glxxtremsuffix}{\glxtrabbrvpluralsuffix}
```

`firstlongemfont` Only used by the “long-em” styles.

```
9080 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
9081 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9082 \newabbreviationstyle{long-short-em}%
9083 {%
9084   \renewcommand*{\CustomAbbreviationFields}{%
9085     name={\glxtrlongshortname},
9086     sort={\the\glsshorttok},
9087     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
9088       \protect\glxtrfullsep{\the\glslabeltok}%
9089       \glxtrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9090     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
9091       \protect\glxtrfullsep{\the\glslabeltok}%
9092       \glxtrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9093     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9094     description={\the\glslongtok}}%
9095   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9096     \glshasattribute{\the\glslabeltok}{regular}}%
9097   {%
9098     \glissetattribute{\the\glslabeltok}{regular}{false}%
9099   }%
9100   {}%
9101 }%
9102 }%
9103 {%
9104   \renewcommand*{\glabbrvfont}[1]{\glsabbrvemfont{##1}}%
9105   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9106   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%

```

Use the default long fonts.

```
9107 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9108 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9109 \renewcommand*{\glxtrfullformat}[2]{%
9110   \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9111   \ifglxtrininsertinside\else##2\fi
9112   \glxtrfullsep{##1}%
9113   \glxtrparen{\glfirstabbrvemfont{\glaccessshort{##1}}}%
9114 }%
9115 \renewcommand*{\glxtrfullplformat}[2]{%
9116   \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

9117 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9118 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9119 }%
9120 \renewcommand*{\Glsxtrfullformat}[2]{%
9121 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9122 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9123 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9124 }%
9125 \renewcommand*{\Glsxtrfullplformat}[2]{%
9126 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9127 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9128 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9129 }%
9130 }

```

g-short-em-desc

```

9131 \newabbreviationstyle{long-short-em-desc}%
9132 {%
9133 \renewcommand*{\CustomAbbreviationFields}{%
9134 name={\glxtrlongshortdescname},
9135 sort={\glxtrlongshortdescsort},%
9136 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9137 \protect\glxtrfullsep{\the\glslabeltok}%
9138 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9139 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9140 \protect\glxtrfullsep{\the\glslabeltok}%
9141 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9142 text={\protect\glssabbrvemfont{\the\glsshorttok}},%
9143 plural={\protect\glssabbrvemfont{\the\glsshortpltok}}}%
9144 }%

```

Unset the regular attribute if it has been set.

```

9145 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9146 \glshasattribute{\the\glslabeltok}{regular}%
9147 {%
9148 \glissetattribute{\the\glslabeltok}{regular}{false}%
9149 }%
9150 {}%
9151 }%
9152 }%
9153 {%

```

As long-short-em style:

```

9154 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9155 }

```

ong-em-short-em

```

9156 \newabbreviationstyle{long-em-short-em}%
9157 {%

```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```

9158 \renewcommand*{\CustomAbbreviationFields}{%
9159   name={\glsxtrlongshortname},
9160   sort={\the\glsshorttok},
9161   first={\protect\glsfirstlongemfont{\the\glslongtok}%
9162     \protect\glsxtrfullsep{\the\glslabeltok}%
9163     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9164   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9165     \protect\glsxtrfullsep{\the\glslabeltok}%
9166     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9167   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9168   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9169 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9170   \glsattribute{\the\glslabeltok}{regular}%
9171   {%
9172     \glssetattribute{\the\glslabeltok}{regular}{false}%
9173   }%
9174   {}%
9175 }%
9176}%
9177{%
9178 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9179 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9180 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9181 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9182 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9183 \renewcommand*{\glsxtrfullformat}[2]{%
9184   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9185   \ifglsxtrininsertinside\else##2\fi
9186   \glsxtrfullsep{##1}%
9187   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9188 }%
9189 \renewcommand*{\glsxtrfullplformat}[2]{%
9190   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9191   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9192   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9193 }%
9194 \renewcommand*{\Glsxtrfullformat}[2]{%
9195   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9196   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9197   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9198 }%
9199 \renewcommand*{\Glsxtrfullplformat}[2]{%
9200   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9201   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

9202 \glstrparen{\glfirstabbrvemfont{\glaccessshortpl{##1}}}%
9203 }%
9204 }

```

m-short-em-desc

```

9205 \newabbreviationstyle{long-em-short-em-desc}%
9206 {%
9207 \renewcommand*{\CustomAbbreviationFields}{%
9208   name={\glxtrlongshortdescname},
9209   sort={\glxtrlongshortdescsort},%
9210   first={\protect\glfirstlongemfont{\the\glslongtok}%
9211     \protect\glxtrfullsep{\the\glslabeltok}%
9212     \glstrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9213   firstplural={\protect\glfirstlongemfont{\the\glslongpltok}%
9214     \protect\glxtrfullsep{\the\glslabeltok}%
9215     \glstrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9216   text={\protect\glabbrvemfont{\the\glsshorttok}},%
9217   plural={\protect\glabbrvemfont{\the\glsshortpltok}}}%
9218 }%

```

Unset the regular attribute if it has been set.

```

9219 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9220 \glshasattribute{\the\glslabeltok}{regular}%
9221 {%
9222 \glissetattribute{\the\glslabeltok}{regular}{false}%
9223 }%
9224 {}%
9225 }%
9226 }%
9227 {%
9228 \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9229 }

```

short-em-long Now the short (long) version

```

9230 \newabbreviationstyle{short-em-long}%
9231 {%
9232 \renewcommand*{\CustomAbbreviationFields}{%
9233   name={\glxtrshortlongname},
9234   sort={\the\glsshorttok},
9235   description={\the\glslongtok},%
9236   first={\protect\glfirstabbrvemfont{\the\glsshorttok}%
9237     \protect\glxtrfullsep{\the\glslabeltok}%
9238     \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongtok}}},%
9239   firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}%
9240     \protect\glxtrfullsep{\the\glslabeltok}%
9241     \glstrparen{\protect\glfirstlongdefaultfont{\the\glslongpltok}}},%
9242   plural={\protect\glabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9243 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

9244 \glshasattribute{\the\glslabeltok}{regular}%
9245 {%
9246 \glissetattribute{\the\glslabeltok}{regular}{false}%
9247 }%
9248 {}%
9249 }%
9250 }%
9251 {%

```

Mostly as short-long style:

```

9252 \renewcommand*\abbrvpluralsuffix{\protect\glxxtremsuffix}%
9253 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9254 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9255 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9256 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9257 \renewcommand*\glsxtrfullformat[2]{%
9258 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9259 \ifglsxtrininsertinside\else##2\fi
9260 \glsxtrfullsep{##1}%
9261 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9262 }%
9263 \renewcommand*\glsxtrfullplformat[2]{%
9264 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9265 \ifglsxtrininsertinside\else##2\fi
9266 \glsxtrfullsep{##1}%
9267 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9268 }%
9269 \renewcommand*\Glsxtrfullformat[2]{%
9270 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9271 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9272 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9273 }%
9274 \renewcommand*\Glsxtrfullplformat[2]{%
9275 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9276 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9277 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9278 }%
9279 }

```

rt-em-long-desc As before but user provides description

```

9280 \newabbreviationstyle{short-em-long-desc}%
9281 {%
9282 \renewcommand*\CustomAbbreviationFields{%
9283 name={\glsxtrshortlongdescname},
9284 sort={\glsxtrshortlongdescsort},
9285 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9286 \protect\glsxtrfullsep{\the\glslabeltok}%
9287 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%

```

```

9288     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9289     \protect\glstrfullsep{\the\glslabeltok}%
9290     \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9291     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9292     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9293 }%

```

Unset the regular attribute if it has been set.

```

9294 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9295   \glshasattribute{\the\glslabeltok}{regular}%
9296   {%
9297     \glssetattribute{\the\glslabeltok}{regular}{false}%
9298   }%
9299   {}%
9300 }%
9301 }%
9302 {%
9303   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9304 }

```

short-em-long-em

```

9305 \newabbreviationstyle{short-em-long-em}%
9306 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9307 \renewcommand*{\CustomAbbreviationFields}{%
9308   name={\glstrshortlongname},
9309   sort={\the\glsshorttok},
9310   description={\protect\glslongemfont{\the\glslongtok}},%
9311   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9312     \protect\glstrfullsep{\the\glslabeltok}%
9313     \glstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9314   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9315     \protect\glstrfullsep{\the\glslabeltok}%
9316     \glstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9317   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9318 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9319   \glshasattribute{\the\glslabeltok}{regular}%
9320   {%
9321     \glssetattribute{\the\glslabeltok}{regular}{false}%
9322   }%
9323   {}%
9324 }%
9325 }%
9326 {%
9327   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9328   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9329   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%

```

```

9330 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9331 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9332 \renewcommand*{\glsxtrfullformat}[2]{%
9333   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9334   \ifglsxtrininsertinside\else##2\fi
9335   \glsxtrfullsep{##1}%
9336   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9337 }%
9338 \renewcommand*{\glsxtrfullplformat}[2]{%
9339   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9340   \ifglsxtrininsertinside\else##2\fi
9341   \glsxtrfullsep{##1}%
9342   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9343 }%
9344 \renewcommand*{\Glsxtrfullformat}[2]{%
9345   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9346   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9347   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9348 }%
9349 \renewcommand*{\Glsxtrfullplformat}[2]{%
9350   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9351   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9352   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9353 }%
9354 }

```

em-long-em-desc

```

9355 \newabbreviationstyle{short-em-long-em-desc}%
9356 {%
9357   \renewcommand*{\CustomAbbreviationFields}{%
9358     name={\glsxtrshortlongdescname},%
9359     sort={\glsxtrshortlongdescsort},%
9360     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9361     \protect\glsxtrfullsep{\the\glslabeltok}%
9362     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9363     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9364     \protect\glsxtrfullsep{\the\glslabeltok}%
9365     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9366     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9367     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9368 }%

```

Unset the regular attribute if it has been set.

```

9369 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9370   \glshasattribute{\the\glslabeltok}{regular}%
9371   {%
9372     \glissetattribute{\the\glslabeltok}{regular}{false}%
9373   }%

```

```

9374     {}%
9375   }%
9376 }%
9377 {%
9378   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9379 }

```

short-em

```

9380 \newabbreviationstyle{short-em}%
9381 {%
9382   \renewcommand*{\CustomAbbreviationFields}{%
9383     name={\glstrshortnolongname},
9384     sort={\the\glsshorttok},
9385     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9386     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9387     text={\protect\glabbrvemfont{\the\glsshorttok}},
9388     plural={\protect\glabbrvemfont{\the\glsshortpltok}},
9389     description={\the\glslongtok}}%
9390   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9391     \glssetAttribute{\the\glslabeltok}{regular}{true}}%
9392 }%
9393 {%
9394   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9395   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9396   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9397   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9398   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9399   \renewcommand*{\glxtrinlinefullformat}[2]{%
9400     \protect\glfirstabbrvemfont{\glsaccessshort{##1}}%
9401     \ifglxtrininsertinside##2\fi}%
9402   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9403   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9404 }%
9405   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9406     \protect\glfirstabbrvemfont{\glsaccessshortpl{##1}}%
9407     \ifglxtrininsertinside##2\fi}%
9408   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9409   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9410 }%

9411   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9412     \protect\glfirstabbrvemfont{\Glsaccessshort{##1}}%
9413     \ifglxtrininsertinside##2\fi}%
9414   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9415   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9416 }%
9417   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9418     \protect\glfirstabbrvemfont{\Glsaccessshortpl{##1}}%

```

```

9419     \ifglxtrinsertinside##2\fi}%
9420     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9421     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9422 }%

```

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.

```

9423 \renewcommand*{\glxtrfullformat}[2]{%
9424   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9425   \ifglxtrinsertinside\else##2\fi
9426 }%
9427 \renewcommand*{\glxtrfullplformat}[2]{%
9428   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9429   \ifglxtrinsertinside\else##2\fi
9430 }%
9431 \renewcommand*{\Glsxtrfullformat}[2]{%
9432   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9433   \ifglxtrinsertinside\else##2\fi
9434 }%
9435 \renewcommand*{\Glsxtrfullplformat}[2]{%
9436   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9437   \ifglxtrinsertinside\else##2\fi
9438 }%
9439 }

```

short-em-nolong

```

9440 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9441 \newabbreviationstyle{short-em-desc}%
9442 {%
9443   \renewcommand*{\CustomAbbreviationFields}{%
9444     name={\glxtrshortdescname},
9445     sort={\the\glssshorttok},
9446     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9447     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9448     text={\protect\glsabbrvemfont{\the\glssshorttok}},
9449     plural={\protect\glsabbrvemfont{\the\glssshortpltok}},
9450     description={\the\glslongtok}}%
9451   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9452     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9453 }%
9454 {%
9455   \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9456   \renewcommand*{\glsabbrvf}[1]{\glsabbrvemfont{##1}}%
9457   \renewcommand*{\glsfirstabbrvf}[1]{\glsfirstabbrvemfont{##1}}%
9458   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9459   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9460 \renewcommand*{\glxtrinlinefullformat}[2]{%
9461   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9462   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9463   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9464 }%
9465 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9466   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9467   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9468   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9469 }%
9470 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9471   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9472   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9473   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9474 }%
9475 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9476   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9477   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9478   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9479 }%

```

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.

```

9480 \renewcommand*{\glxtrfullformat}[2]{%
9481   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9482   \ifglxtrininsertinside\else##2\fi
9483 }%
9484 \renewcommand*{\glxtrfullplformat}[2]{%
9485   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9486   \ifglxtrininsertinside\else##2\fi
9487 }%
9488 \renewcommand*{\Glsxtrfullformat}[2]{%
9489   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9490   \ifglxtrininsertinside\else##2\fi
9491 }%
9492 \renewcommand*{\Glsxtrfullplformat}[2]{%
9493   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9494   \ifglxtrininsertinside\else##2\fi
9495 }%
9496 }

```

-em-nolong-desc

```

9497 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9498 \newabbreviationstyle{nolong-short-em}%
9499 {%
9500   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9501 }%

```

```

9502 {%
9503   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%
    The inline full form displays the long form followed by the short form in parentheses.
9504   \renewcommand*{\glsxtrinlinefullformat}[2]{%
9505     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}}%
9506     \ifglsxtrininsertinside##2\fi}%
9507   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9508   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9509 }%
9510   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9511     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}%
9512     \ifglsxtrininsertinside##2\fi}%
9513   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9514   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9515 }%
9516   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9517     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}}%
9518     \ifglsxtrininsertinside##2\fi}%
9519   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9520   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9521 }%
9522   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9523     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}%
9524     \ifglsxtrininsertinside##2\fi}%
9525   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9526   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9527 }%
9528 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9529 \newabbreviationstyle{long-noshort-em}%
9530 {%
9531   \renewcommand*{\CustomAbbreviationFields}{%
9532     name={\glxtrlongnoshortname},
9533     sort={\the\glsshorttok},
9534     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9535     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9536     text={\protect\glslongdefaultfont{\the\glslongtok}},
9537     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9538     description={\the\glslongtok}%
9539   }%
9540   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9541     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9542 }%
9543 {%
9544   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9545   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9546   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9547   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%

```

```
9548 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9549 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9550   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9551   \ifglsxtrininsertinside \else##2\fi
9552 }%
9553 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9554   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9555   \ifglsxtrininsertinside \else##2\fi
9556 }%
9557 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9558   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9559   \ifglsxtrininsertinside \else##2\fi
9560 }%
9561 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9562   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9563   \ifglsxtrininsertinside \else##2\fi
9564 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9565 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9566   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9567   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9568   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9569 }%
9570 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9571   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9572   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9573   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9574 }%
9575 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9576   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9577   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9578   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9579 }%
9580 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9581   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9582   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9583   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9584 }%
```

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.

```
9585 \renewcommand*{\glsxtrfullformat}[2]{%
9586   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9587   \ifglsxtrininsertinside\else##2\fi
9588 }%
9589 \renewcommand*{\glsxtrfullplformat}[2]{%
9590   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9591   \ifglsxtrininsertinside\else##2\fi
```

```

9592 }%
9593 \renewcommand*{\Glsxtrfullformat}[2]{%
9594   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9595   \ifglxtrinsertinside\else##2\fi
9596 }%
9597 \renewcommand*{\Glsxtrfullplformat}[2]{%
9598   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9599   \ifglxtrinsertinside\else##2\fi
9600 }%
9601 }

```

long-em Backward compatibility:

```

9602 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9603 \newabbreviationstyle{long-em-noshort-em}%
9604 {%
9605   \renewcommand*{\CustomAbbreviationFields}{%
9606     name={\glxtrlongnoshortname},
9607     sort={\the\glsshorttok},
9608     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9609     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9610     text={\protect\glslongemfont{\the\glslongtok}},
9611     plural={\protect\glslongemfont{\the\glslongpltok}},%
9612     description={\protect\glslongemfont{\the\glslongtok}}%
9613   }%
9614   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9615     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9616 }%
9617 {%
9618   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9619   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9620   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9621   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9622   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9623 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9624   \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9625   \ifglxtrinsertinside \else##2\fi
9626 }%
9627 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9628   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9629   \ifglxtrinsertinside \else##2\fi
9630 }%
9631 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9632   \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9633   \ifglxtrinsertinside \else##2\fi
9634 }%
9635 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

```

```

9636 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9637 \ifglxtrinsertinside \else##2\fi
9638 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9639 \renewcommand*{\glxtrinlinefullformat}[2]{%
9640 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9641 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9642 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9643 }%
9644 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9645 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9646 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9647 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9648 }%
9649 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9650 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9651 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9652 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9653 }%
9654 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9655 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9656 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9657 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9658 }%

```

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.

```

9659 \renewcommand*{\glxtrfullformat}[2]{%
9660 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9661 \ifglxtrinsertinside\else##2\fi
9662 }%
9663 \renewcommand*{\glxtrfullplformat}[2]{%
9664 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9665 \ifglxtrinsertinside\else##2\fi
9666 }%
9667 \renewcommand*{\Glsxtrfullformat}[2]{%
9668 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9669 \ifglxtrinsertinside\else##2\fi
9670 }%
9671 \renewcommand*{\Glsxtrfullplformat}[2]{%
9672 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9673 \ifglxtrinsertinside\else##2\fi
9674 }%
9675 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

9676 \newabbreviationstyle{long-em-noshort-em-noreg}%
9677 {%
9678 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%

```

Unset the regular attribute if it has been set.

```

9679 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9680   \glshasattribute{\the\glslabeltok}{regular}}%
9681   {%
9682     \glissetattribute{\the\glslabeltok}{regular}{false}}%
9683   }%
9684   {}%
9685 }%
9686 }%
9687 {%
9688   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}}%
9689 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

```

9690 \newabbreviationstyle{long-noshort-em-desc}%
9691 {%
9692   \GlsXtrUseAbbrStyleSetup{long-noshort-em-desc}}%
9693 }%
9694 {%
9695   \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9696   \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9697   \renewcommand*{\glssfirstabrvfont}[1]{\glssfirstabrvemfont{##1}}%
9698   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9699   \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9700 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9701   \glsslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9702   \ifglsxtrininsertinside \else##2\fi
9703 }%
9704 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9705   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9706   \ifglsxtrininsertinside \else##2\fi
9707 }%
9708 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9709   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9710   \ifglsxtrininsertinside \else##2\fi
9711 }%
9712 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9713   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9714   \ifglsxtrininsertinside \else##2\fi
9715 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9716 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9717   \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9718   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9719   \glsxtrparen{\protect\glssfirstabrvemfont{\glssaccessshort{##1}}}%
9720 }%

```

```

9721 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9722   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9723   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9724   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9725 }%
9726 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9727   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9728   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9729   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9730 }%
9731 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9732   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9733   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9734   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9735 }%

```

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.

```

9736 \renewcommand*{\glxtrfullformat}[2]{%
9737   \glsfirslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9738   \ifglxtrinsertinside\else##2\fi
9739 }%
9740 \renewcommand*{\glxtrfullplformat}[2]{%
9741   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9742   \ifglxtrinsertinside\else##2\fi
9743 }%
9744 \renewcommand*{\Glsxtrfullformat}[2]{%
9745   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9746   \ifglxtrinsertinside\else##2\fi
9747 }%
9748 \renewcommand*{\Glsxtrfullplformat}[2]{%
9749   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9750   \ifglxtrinsertinside\else##2\fi
9751 }%
9752 }

```

long-desc-em Backward compatibility:

```
9753 \@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.

```

9754 \newabbreviationstyle{long-em-noshort-em-desc}%
9755 {%
9756   \renewcommand*{\CustomAbbreviationFields}{%
9757     name={\glxtrlongnoshortdescname},
9758     sort={\the\glslongtok},
9759     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9760     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9761     text={\glslongemfont{\the\glslongtok}},

```

```

9762 plural={\glslongemfont{\the\glslongpltok}}%
9763 }%
9764 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9765   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9766 }%
9767 {%
9768   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9769   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9770   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9771   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9772   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9773 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9774   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9775   \ifglsxtrininsertinside \else##2\fi
9776 }%
9777 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9778   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9779   \ifglsxtrininsertinside \else##2\fi
9780 }%
9781 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9782   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9783   \ifglsxtrininsertinside \else##2\fi
9784 }%
9785 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9786   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9787   \ifglsxtrininsertinside \else##2\fi
9788 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9789 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9790   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9791   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9792   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9793 }%
9794 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9795   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9796   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9797   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9798 }%
9799 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9800   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9801   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9802   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9803 }%
9804 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9805   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9806   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9807   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%

```

9808 }%

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.

```
9809 \renewcommand*{\glxtrfullformat}[2]{%
9810   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9811   \ifglxtrinsertinside\else##2\fi
9812 }%
9813 \renewcommand*{\glxtrfullplformat}[2]{%
9814   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9815   \ifglxtrinsertinside\else##2\fi
9816 }%
9817 \renewcommand*{\Glsxtrfullformat}[2]{%
9818   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9819   \ifglxtrinsertinside\else##2\fi
9820 }%
9821 \renewcommand*{\Glsxtrfullplformat}[2]{%
9822   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9823   \ifglxtrinsertinside\else##2\fi
9824 }%
9825 }
```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```
9826 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9827 {%
9828   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
```

Unset the regular attribute if it has been set.

```
9829 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9830   \glshasattribute{\the\glslabeltok}{regular}%
9831   {%
9832     \glssetattribute{\the\glslabeltok}{regular}{false}%
9833   }%
9834   {}%
9835 }%
9836 }%
9837 {%
9838   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9839 }
```

ort-em-footnote

```
9840 \newabbreviationstyle{short-em-footnote}%
9841 {%
9842   \renewcommand*{\CustomAbbreviationFields}{%
9843     name={\glxtrfootnotename},
9844     sort={\the\glsshorttok},
9845     description={\the\glslongtok},%
9846     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9847       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9848       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
```

```

9849   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9850   \protect\glsextrabbrvfootnote{\the\glslabeltok}%
9851   {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9852   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.

```

9853   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9854   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9855   \glsattribute{\the\glslabeltok}{regular}%
9856   {%
9857   \glssetattribute{\the\glslabeltok}{regular}{false}%
9858   }%
9859   {}%
9860   }%
9861 }%
9862 {%
9863   \renewcommand*{\abbrvpluralsuffix}{\protect\glsextrsuffix}%
9864   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9865   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9866   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9867   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9868   \renewcommand*{\glsextrfullformat}[2]{%
9869   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsextrinsertinside##2\fi}%
9870   \ifglsextrinsertinside\else##2\fi
9871   \protect\glsextrabbrvfootnote{##1}%
9872   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9873   }%
9874   \renewcommand*{\glsextrfullplformat}[2]{%
9875   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsextrinsertinside##2\fi}%
9876   \ifglsextrinsertinside\else##2\fi
9877   \protect\glsextrabbrvfootnote{##1}%
9878   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9879   }%
9880   \renewcommand*{\GlsXtrfullformat}[2]{%
9881   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsextrinsertinside##2\fi}%
9882   \ifglsextrinsertinside\else##2\fi
9883   \protect\glsextrabbrvfootnote{##1}%
9884   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9885   }%
9886   \renewcommand*{\GlsXtrfullplformat}[2]{%
9887   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsextrinsertinside##2\fi}%
9888   \ifglsextrinsertinside\else##2\fi
9889   \protect\glsextrabbrvfootnote{##1}%
9890   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9891   }%

```

The first use full form and the inline full form use the short (long) style.

```

9892 \renewcommand*{\glxtrinlinefullformat}[2]{%
9893   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9894   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9895   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9896 }%
9897 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9898   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9899   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9900   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9901 }%
9902 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9903   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9904   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9905   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9906 }%
9907 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9908   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9909   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9910   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9911 }%
9912 }

```

footnote-em Backward compatibility:

```

9913 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9914 \newabbreviationstyle{short-em-postfootnote}%
9915 {%
9916   \renewcommand*{\CustomAbbreviationFields}{%
9917     name={\glxtrfootnotename},
9918     sort={\the\glssshorttok},
9919     description={\the\glslongtok},%
9920     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},%
9921     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},%
9922     plural={\protect\glssabbrvemfont{\the\glssshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

9923 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9924   \csdef{glxtrpostlink\glscategorylabel}{%
9925     \glxtrifwasfirstuse
9926     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9927     \glxtrdopostpunc{\protect\glxtrabbrvf footnote{\glslabel}%
9928     {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
9929   }%
9930 }%
9931 }%

```

```

9932 \glshasattribute{\the\glslabeltok}{regular}%
9933 {%
9934 \glissetattribute{\the\glslabeltok}{regular}{false}%
9935 }%
9936 {}%
9937 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9938 \renewcommand*\glxtrsetupfulldefs{%
9939 \let\glxtrifwasfirstuse\@secondoftwo
9940 }%
9941 }%
9942 {%
9943 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
9944 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9945 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9946 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
9947 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9948 \renewcommand*\glxtrfullformat[2]{%
9949 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9950 \ifglxtrininsertinside\else##2\fi
9951 }%
9952 \renewcommand*\glxtrfullplformat[2]{%
9953 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9954 \ifglxtrininsertinside\else##2\fi
9955 }%
9956 \renewcommand*\Glsxtrfullformat[2]{%
9957 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9958 \ifglxtrininsertinside\else##2\fi
9959 }%
9960 \renewcommand*\Glsxtrfullplformat[2]{%
9961 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9962 \ifglxtrininsertinside\else##2\fi
9963 }%

```

The first use full form and the inline full form use the short (long) style.

```

9964 \renewcommand*\glxtrininlinefullformat[2]{%
9965 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9966 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9967 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9968 }%
9969 \renewcommand*\glxtrininlinefullplformat[2]{%
9970 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9971 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9972 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9973 }%
9974 \renewcommand*\Glsxtrininlinefullformat[2]{%
9975 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```

```

9976     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9977     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9978 }%
9979 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9980     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9981     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9982     \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9983 }%
9984 }

```

postfootnote-em Backward compatibility:

```

9985 \@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.

```

9986 \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.

```

9987 \ifdef\glscurrentfieldvalue
9988 {
9989     \newcommand*{\glxtruserparen}[2]{%
9990         \glxtrfullsep{#2}%
9991         \glxtrparen
9992             {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9993     }
9994 }
9995 {
9996     \newcommand*{\glxtruserparen}[2]{%
9997         \glxtrfullsep{#2}%
9998         \glxtrparen
9999             {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
10000     }
10001 }

```

Font used for short form:

lsabbrvuserfont

```

10002 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

10003 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
10004 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10005 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10006 \newcommand*{\lsxtrusersuffix}{\lsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10007 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10008 \newabbreviationstyle{long-short-user}%
10009 {%
10010   \renewcommand*{\CustomAbbreviationFields}{%
10011     name={\lsxtrlongshortname},
10012     sort={\the\glsshorttok},
10013     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10014       \protect\lsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10015     {\the\glslabeltok}},%
10016     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10017       \protect\lsxtruserparen
10018       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10019     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10020     description={\protect\glsuserdescription{\the\glslongtok}%
10021       {\the\glslabeltok}}}%
10021 }
```

Unset the regular attribute if it has been set.

```
10022 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10023   \glshasattribute{\the\glslabeltok}{regular}%
10024   {%
10025     \glssetattribute{\the\glslabeltok}{regular}{false}%
10026   }%
10027 }%
10028 }%
10029 }%
10030 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10031 \renewcommand*{\abbrvpluralsuffix}{\lsxtrusersuffix}%
10032 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
```

```

10033 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10034 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10035 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10036 \renewcommand*{\glsxtrfullformat}[2]{%
10037   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10038   \ifglsxtrinsertinside\else##2\fi
10039   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10040 }%
10041 \renewcommand*{\glsxtrfullplformat}[2]{%
10042   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10043   \ifglsxtrinsertinside\else##2\fi
10044   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10045 }%
10046 \renewcommand*{\Glsxtrfullformat}[2]{%
10047   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10048   \ifglsxtrinsertinside\else##2\fi
10049   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10050 }%
10051 \renewcommand*{\Glsxtrfullplformat}[2]{%
10052   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10053   \ifglsxtrinsertinside\else##2\fi
10054   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10055 }%
10056 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10057 \newabbreviationstyle{long-postshort-user}%
10058 {%
10059   \renewcommand*{\CustomAbbreviationFields}{%
10060     name={\glsxtrlongshortname},
10061     sort={\the\glsshorttok},
10062     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10063     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10064     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10065     description={\protect\glsuserdescription{\the\glslongtok}%
10066       {\the\glslabeltok}}}%
10067   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10068     \csdef{glsxtrpostlink\glscategorylabel}{%
10069       \glsxtrifwasfirstuse
10070       {%
10071         \glsxtruserparen
10072           {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10073           {\glslabel}%
10074       }%
10075       {}}%
10076   }%
10077   \glshasattribute{\the\glslabeltok}{regular}%

```

```

10078   {%
10079     \glssetattribute{\the\glslabeltok}{regular}{false}%
10080   }%
10081   {}%
10082 }%
10083 }%
10084 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10085 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10086 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10087 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10088 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10089 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10090 \renewcommand*{\glsxtrfullformat}[2]{%
10091   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10092   \ifglsxtrininsertinside\else##2\fi
10093 }%
10094 \renewcommand*{\glsxtrfullplformat}[2]{%
10095   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10096   \ifglsxtrininsertinside\else##2\fi
10097 }%
10098 \renewcommand*{\Glsxtrfullformat}[2]{%
10099   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10100   \ifglsxtrininsertinside\else##2\fi
10101 }%
10102 \renewcommand*{\Glsxtrfullplformat}[2]{%
10103   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10104   \ifglsxtrininsertinside\else##2\fi
10105 }%

```

In-line format:

```

10106 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10107   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10108   \ifglsxtrininsertinside\else##2\fi
10109   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10110 }%
10111 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10112   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10113   \ifglsxtrininsertinside\else##2\fi
10114   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10115 }%
10116 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10117   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10118   \ifglsxtrininsertinside\else##2\fi
10119   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10120 }%
10121 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

```

```

10122 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10123 \ifglxtrinsertinside\else##2\fi
10124 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10125 }%
10126 }

```

ortuserdescname

```

10127 \newcommand*{\glxtrlongshortuserdescname}{%
10128 \protect\glslonguserfont{\the\glslongtok}%
10129 \protect\glxtruserparen
10130 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10131 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10132 \newabbreviationstyle{long-postshort-user-desc}%
10133 {%
10134 \renewcommand*{\CustomAbbreviationFields}{%
10135 name={\glxtrlongshortuserdescname},
10136 sort={\the\glslongtok},
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 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10148 {\glslabel}%
10149 }%
10150 }%
10151 }%
10152 \glshasattribute{\the\glslabeltok}{regular}%
10153 {%
10154 \glssetattribute{\the\glslabeltok}{regular}{false}%
10155 }%
10156 {}%
10157 }%
10158 }%
10159 {%
10160 \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10161 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10162 \newabbreviationstyle{short-postlong-user}%
10163 {%

```

```

10164 \renewcommand*{\CustomAbbreviationFields}{%
10165     name={\glxtrshortlongname},
10166     sort={\the\glsshorttok},
10167     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10168     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10169     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10170     description={\protect\glsuserdescription{\the\glslongtok}%
10171         {\the\glslabeltok}}}%
10172 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10173     \csdef{glxtrpostlink\glscategorylabel}{%
10174         \glxtrifwasfirstuse
10175         {%
10176             \glxtruserparen
10177             {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10178             {\glslabel}}%
10179         }%
10180     }%
10181 }%
10182 \glsattribute{\the\glslabeltok}{regular}%
10183 {%
10184     \glssetattribute{\the\glslabeltok}{regular}{false}%
10185 }%
10186 {}%
10187 }%
10188 }%
10189 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10190 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10191 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10192 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10193 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10194 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10195 \renewcommand*{\glxtrfullformat}[2]{%
10196     \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10197     \ifglxtrininsertinside\else##2\fi
10198 }%
10199 \renewcommand*{\glxtrfullplformat}[2]{%
10200     \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10201     \ifglxtrininsertinside\else##2\fi
10202 }%
10203 \renewcommand*{\Glsxtrfullformat}[2]{%
10204     \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10205     \ifglxtrininsertinside\else##2\fi
10206 }%
10207 \renewcommand*{\Glsxtrfullplformat}[2]{%
10208     \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

10209 \ifglxtrinsertinside\else##2\fi
10210 }%
      In-line format:
10211 \renewcommand*{\glxtrinlinefullformat}[2]{%
10212   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10213   \ifglxtrinsertinside\else##2\fi
10214   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10215 }%
10216 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10217   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10218   \ifglxtrinsertinside\else##2\fi
10219   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10220 }%
10221 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10222   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10223   \ifglxtrinsertinside\else##2\fi
10224   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10225 }%
10226 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10227   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10228   \ifglxtrinsertinside\else##2\fi
10229   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10230 }%
10231 }

```

onguserdescname

```

10232 \newcommand*{\glxtrshortlonguserdescname}{%
10233   \protect\glssabbrvuserfont{\the\glssshorttok}%
10234   \protect\glxtruserparen
10235     {\protect\glslonguserfont{\the\glslongpltok}}}%
10236   {\the\glslabelltok}%
10237 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10238 \newabbreviationstyle{short-postlong-user-desc}%
10239 {%
10240   \renewcommand*{\CustomAbbreviationFields}{%
10241     name={\glxtrshortlonguserdescname},
10242     sort={\the\glssshorttok},
10243     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10244     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10245     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10246     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10247 }%
10248 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10249   \csdef{glxtrpostlink\glscategorylabel}{%
10250     \glxtrifwasfirstuse
10251     {%

```

```

10252      \glxtruserparen
10253      {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10254      {\glslabel}%
10255      }%
10256      {}%
10257      }%
10258      \glshasattribute{\the\glslabeltok}{regular}%
10259      {%
10260      \glissetattribute{\the\glslabeltok}{regular}{false}%
10261      }%
10262      {}%
10263      }%
10264      }%
10265      {%
10266      \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10267      }

```

short-user-desc

```

10268 \newabbreviationstyle{long-short-user-desc}%
10269 {%
10270   \renewcommand*{\CustomAbbreviationFields}{%
10271     name={\glxtrlongshortuserdescname},
10272     sort={\glxtrlongshortdescsort},%

10273     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10274       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10275     {\the\glslabeltok}},%
10276     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10277       \protect\glxtruserparen
10278       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}}},%
10279     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10280     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10281   }%

```

Unset the regular attribute if it has been set.

```

10282   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10283     \glshasattribute{\the\glslabeltok}{regular}%
10284     {%
10285       \glissetattribute{\the\glslabeltok}{regular}{false}%
10286       }%
10287     {}%
10288   }%
10289   }%
10290   {%
10291   \GlsXtrUseAbbrStyleFmts{long-short-user}%
10292   }

```

short-long-user

```

10293 \newabbreviationstyle{short-long-user}%
10294 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10295 \renewcommand*{\CustomAbbreviationFields}{%
10296   name={\glsxtrshortlongname},
10297   sort={\the\glsshorttok},
10298   description={\protect\glsuserdescription{\the\glslongtok}%
10299     {\the\glslabeltok}},%
10300   first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10301     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10302     {\the\glslabeltok}},%
10303   firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10304     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10305     {\the\glslabeltok}},%

10306   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10307 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10308   \glsattribute{\the\glslabeltok}{regular}%
10309   {%
10310     \glssetattribute{\the\glslabeltok}{regular}{false}%
10311   }%
10312   {}%
10313 }%
10314 }%
10315 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10316 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10317 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10318 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10319 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10320 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10321 \renewcommand*{\glsxtrfullformat}[2]{%
10322   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10323   \ifglsxtrininsertinside\else##2\fi
10324   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10325 }%
10326 \renewcommand*{\glsxtrfullplformat}[2]{%
10327   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10328   \ifglsxtrininsertinside\else##2\fi
10329   \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10330 }%
10331 \renewcommand*{\Glsxtrfullformat}[2]{%
10332   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10333   \ifglsxtrininsertinside\else##2\fi
10334   \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10335 }%

```

```

10336 \renewcommand*{\Glsxtrfullplformat}[2]{%
10337   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10338   \ifglxtrinsertinside\else##2\fi
10339   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10340 }%
10341 }

```

-long-user-desc

```

10342 \newabbreviationstyle{short-long-user-desc}%
10343 {%
10344   \renewcommand*{\CustomAbbreviationFields}{%
10345     name={\glxtrshortlonguserdescname},
10346     sort={\glxtrshortlongdescsort},%

10347     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
10348     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10349     {\the\glslabeltok}},%
10350     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}%
10351     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10352     {\the\glslabeltok}},%
10353     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10354     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10355 }%

```

Unset the regular attribute if it has been set.

```

10356 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10357   \glshasattribute{\the\glslabeltok}{regular}%
10358   {%
10359     \glissetattribute{\the\glslabeltok}{regular}{false}%
10360   }%
10361   {}%
10362 }%
10363 }%
10364 {%
10365   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10366 }

```

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 `\glxtrlong` set `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

`trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```

10367 \newrobustcmd*{\glxtrifhyphenstart}[3]{%
10368   \ifx\glsinsert#1\relax
10369   \expandafter@glxtrifhyphenstart#1\relax\relax

```

```

10370      \@end@glxstrifhyphenstart{#2}{#3}%
10371  \else
10372      \@glxstrifhyphenstart#1\relax\relax\@end@glxstrifhyphenstart{#2}{#3}%
10373  \fi
10374 }

```

trifhyphenstart

```

10375 \def\@glxstrifhyphenstart#1#2\@end@glxstrifhyphenstart#3#4{%
10376   \ifx-#1\relax#3\else #4\fi
10377 }

```

rlonghyphenshort

```
\glxstrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10378 \newcommand*\glxstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10379  {%
```

If *<insert>* starts with a hyphen, redefine `\glxstrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxstrwordsep` if *<insert>* doesn't start with a hyphen.

```

10380   \glxstrifhyphenstart{#4}{\def\glxstrwordsep{-}}{}%
10381   \glsfirslonghyphenfont{#2\ifglxstrinsertinside{#4}\fi}%
10382   \ifglxstrinsertinside\else{#4}\fi
10383   \glxstrfullsep{#1}%
10384   \glxstrparen{\glsfirstabbrvhyphenfont{#3\ifglxstrinsertinside{#4}\fi}%
10385     \ifglxstrinsertinside\else{#4}\fi}%
10386 }%
10387 }

```

abbrvhyphenfont

```
10388 \newcommand*\glsabbrvhyphenfont{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10389 \newcommand*\glsfirstabbrvhyphenfont{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10390 \newcommand*\glslonghyphenfont{\glslongdefaultfont}%
```

tlonghyphenfont

```
10391 \newcommand*\glsfirslonghyphenfont{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10392 \newcommand*{\glxtrhyphensuffix}{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the markwords attribute.

```
10393 \newabbreviationstyle{long-hyphen-short-hyphen}%
10394 {%
10395   \renewcommand*{\CustomAbbreviationFields}{%
10396     name={\glxtrlongshortname},
10397     sort={\the\glsshorttok},
10398     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10399       \protect\glxtrfullsep{\the\glslabeltok}%
10400       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10401     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10402       \protect\glxtrfullsep{\the\glslabeltok}%
10403       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10404     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
10405     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10406   }
```

Unset the regular attribute if it has been set.

```
10406   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10407     \glshasattribute{\the\glslabeltok}{regular}%
10408     {%
10409       \glissetattribute{\the\glslabeltok}{regular}{false}%
10410     }%
10411   }%
10412 }%
10413 }%
10414 {%
10415   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10416   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10417   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10418   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10419   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
10420 }
```

The first use full form and the inline full form are the same for this style.

```
10420   \renewcommand*{\glxtrfullformat}[2]{%
10421     \glxtrlonghyphenshort{##1}{\glssaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
10422   }%
10423   \renewcommand*{\glxtrfullplformat}[2]{%
10424     \glxtrlonghyphenshort{##1}{\glssaccesslongpl{##1}}%
10425     {\glssaccessshortpl{##1}}{##2}%
10426   }%
10427   \renewcommand*{\Glsxtrfullformat}[2]{%
10428     \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
10429   }%
10430   \renewcommand*{\Glsxtrfullplformat}[2]{%
10431     \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10432     {\glssaccessshortpl{##1}}{##2}%
10433   }%
10434 }
```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10435 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10436 {%
10437   \renewcommand*{\CustomAbbreviationFields}{%
10438     name={\glxstrlongshortdescname},
10439     sort={\glxstrlongshortdescsort},
10440     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10441       \protect\glxstrfullsep{\the\glslabeltok}%
10442       \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10443     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10444       \protect\glxstrfullsep{\the\glslabeltok}%
10445       \glxstrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10446     text={\protect\glssabbrvhyphenfont{\the\glsshorttok}},%
10447     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}}%
10448   }%

```

Unset the regular attribute if it has been set.

```

10449   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10450     \glshasattribute{\the\glslabeltok}{regular}%
10451     {%
10452       \glissetattribute{\the\glslabeltok}{regular}{false}%
10453     }%
10454   }%
10455 }%
10456 }%
10457 {%
10458   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10459 }

```

onghyphennoshort

```
\glxstrlonghyphennoshort{\label}{\long}{\insert}
```

```
10460 \newcommand*{\glxstrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
10461 {%
```

If *<insert>* starts with a hyphen, redefine `\glxstrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxstrwordsep` if *<insert>* doesn't start with a hyphen.

```

10462   \glxstrifhyphenstart{#3}{\def\glxstrwordsep{-}}{%
10463     \glsfirstlonghyphenfont{#2\ifglxstrinsertinside{#3}\fi}%
10464     \ifglxstrinsertinside\else{#3}\fi
10465   }%
10466 }

```

short-desc-noreg

This version doesn't show the short form (except explicitly with `\glxstrshort`). Since `\glxstrshort` 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.

```

10467 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10468 {%
10469   \renewcommand*{\CustomAbbreviationFields}{%
10470     name={\glxtrlongnoshortdescname},
10471     sort={\expandonce\glxtrorglong},
10472     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10473     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10474     plural={\protect\glslonghyphenfont{\the\glslongpltok}}}%
10475   }%

```

Unset the regular attribute if it has been set.

```

10476 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10477   \glshasattribute{\the\glslabeltok}{regular}%
10478   {%
10479     \glissetattribute{\the\glslabeltok}{regular}{false}%
10480   }%
10481   {}%
10482 }%
10483 }%
10484 {%
10485   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10486 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10487 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
10488 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10489 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10490 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10491 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10492   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10493 }%
10494 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10495   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10496 }%
10497 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10498   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10499 }%
10500 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10501   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10502 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10503 \renewcommand*{\glxtrinlinenfullformat}[2]{%
10504   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10505   \glxtrfullsep{##1}%
10506   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10507 }%

```

```

10508 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10509   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10510   \glxtrfullsep{##1}%
10511   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10512 }%
10513 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10514   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10515   \glxtrfullsep{##1}%
10516   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10517 }%
10518 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10519   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10520   \glxtrfullsep{##1}%
10521   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10522 }%

```

The first use full form only displays the long form.

```

10523 \renewcommand*{\glxtrfullformat}[2]{%
10524   \glxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10525 }%
10526 \renewcommand*{\glxtrfullplformat}[2]{%
10527   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10528 }%
10529 \renewcommand*{\Glsxtrfullformat}[2]{%
10530   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10531 }%
10532 \renewcommand*{\Glsxtrfullplformat}[2]{%
10533   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10534 }%
10535 }

```

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.

```

10536 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10537 {%
10538   \renewcommand*{\CustomAbbreviationFields}{%
10539     name={\glxtrlongnoshortname},
10540     sort={\the\glsshorttok},
10541     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10542     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10543     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10544     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10545     description={\the\glslongtok}%
10546   }%

```

Unset the regular attribute if it has been set.

```

10547 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10548   \glshasattribute{\the\glslabeltok}{regular}%
10549   {%

```

```

10550     \glsssetAttribute{\the\glslabelltok}{regular}{false}%
10551   }%
10552   {}%
10553 }%
10554 }%
10555 {%
10556   \GlsXtrUseAbbrStyleFmts{long-desc}%
10557 }

```

glxstrlonghyphen `\glxstrlonghyphen{<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.

```

10558 \newcommand*{\glxstrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10559   {%
10560     \glxstrifhyphenstart{#3}{\def\glxstrwordsep{-}}{}%
10561     \glxstrfirstlonghyphenfont{#1}%
10562   }%
10563 }

```

posthyphenshort `\glxstrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glxstrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```

10564 \newcommand*{\glxstrposthyphenshort}[2]{%
10565   {%
10566     \glxstrifhyphenstart{#2}{\def\glxstrwordsep{-}}{}%
10567     \ifglxstrinsertinside{\glxstrfirstlonghyphenfont{#2}}\else{#2}\fi
10568     \glxstrfullsep{#1}%
10569     \glxstrparen
10570     {\glxstrfirstabbrvhyphenfont{\glxstryshort{#1}}\ifglxstrinsertinside{#2}\fi}%
10571     \ifglxstrinsertinside\else{#2}\fi
10572   }%
10573 }%
10574 }

```

hyphensequent `\glxstrposthyphensequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10575 \newcommand*{\glxtrposthyphensubsequent}[2]{%
10576   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
10577   \ifglxtrininsertinside \else{#2}\fi
10578 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10579 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10580 {%
10581   \renewcommand*{\CustomAbbreviationFields}{%
10582     name={\glsxtrlongshortname},
10583     sort={\the\glsshorttok},
10584     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10585     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10586     plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
10587     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10588   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10589     \csdef{glxtrpostlink\glscategorylabel}{%
10590       \glsxtrifwasfirstuse
10591       {%
10592         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10593       }%
10594       {%

```

Put the insertion into the post-link:

```

10595       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10596     }%
10597   }%
10598   \glshasattribute{\the\glslabeltok}{regular}%
10599   {%
10600     \glissetattribute{\the\glslabeltok}{regular}{false}%
10601   }%
10602   {}%
10603 }%
10604 }%
10605 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10606 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
10607 \renewcommand*{\glssabrvfont}[1]{\glssabrvhyphenfont{##1}}%
10608 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10609 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10610 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10611 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10612   \glssabrvfont{\glssaccessshort{##1}}%
10613 }%
10614 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10615   \glssabrvfont{\glssaccessshortpl{##1}}%

```

```

10616 }%
10617 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10618   \glsabbrvfont{\Glsaccessshort{##1}}}%
10619 }%
10620 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10621   \glsabbrvfont{\Glsaccessshortpl{##1}}}%
10622 }%

  First use full form:

10623 \renewcommand*{\glsxtrfullformat}[2]{%
10624   \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10625 }%
10626 \renewcommand*{\glsxtrfullplformat}[2]{%
10627   \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10628 }%
10629 \renewcommand*{\Glsxtrfullformat}[2]{%
10630   \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10631 }%
10632 \renewcommand*{\Glsxtrfullplformat}[2]{%
10633   \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10634 }%

```

In-line format.

```

10635 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10636   \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10637   \ifglsxtrininsertinside{##2}\fi}%
10638   \ifglsxtrininsertinside \else{##2}\fi
10639 }%
10640 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10641   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10642   \ifglsxtrininsertinside{##2}\fi}%
10643   \ifglsxtrininsertinside \else{##2}\fi
10644 }%
10645 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10646   \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10647   \ifglsxtrininsertinside{##2}\fi}%
10648   \ifglsxtrininsertinside \else{##2}\fi
10649 }%
10650 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10651   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10652   \ifglsxtrininsertinside{##2}\fi}%
10653   \ifglsxtrininsertinside \else{##2}\fi
10654 }%
10655 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10656 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10657 {%
10658   \renewcommand*{\CustomAbbreviationFields}{%
10659     name={\glsxtrlongshortdescname},

```

```

10660     sort={\glxtrlongshortdescsort},%
10661     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10662     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10663     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10664     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10665 }%
10666 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10667   \csdef{glxtrpostlink\glscategorylabel}{%
10668     \glxtrifwasfirstuse
10669     {%
10670       \glxtrposthyphenshort{\glslabel}{\glsinsert}}%
10671     }%
10672     {%

```

Put the insertion into the post-link:

```

10673       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
10674     }%
10675   }%
10676   \glshasattribute{\the\glslabeltok}{regular}%
10677   {%
10678     \glissetattribute{\the\glslabeltok}{regular}{false}%
10679   }%
10680   {}%
10681 }%
10682 }%
10683 {%
10684   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10685 }

```

rshorthyphenlong

```
\glxtrshorthyphenlong{<label>}{<short>}{<long>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10686 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10687 {%
```

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.)

```

10688   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{%
10689   \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10690   \ifglxtrininsertinside\else{#4}\fi
10691   \glxtrfullsep{#1}%
10692   \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10693   \ifglxtrininsertinside\else{#4}\fi}%

```

```

10694 }%
10695 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```

10696 \newabbreviationstyle{short-hyphen-long-hyphen}%
10697 {%
10698   \renewcommand*{\CustomAbbreviationFields}{%
10699     name={\glxtrshortlongname},
10700     sort={\the\glsshorttok},
10701     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10702       \protect\glxtrfullsep{\the\glslabeltok}%
10703       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10704     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10705       \protect\glxtrfullsep{\the\glslabeltok}%
10706       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10707     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10708     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10709   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10710     \glshasattribute{\the\glslabeltok}{regular}%
10711     {%
10712       \glissetattribute{\the\glslabeltok}{regular}{false}%
10713     }%
10714   }%
10715 }%
10716 }%
10717 {%
10718   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10719   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10720   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10721   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10722   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10723   \renewcommand*{\glxtrfullformat}[2]{%
10724     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10725   }%
10726   \renewcommand*{\glxtrfullplformat}[2]{%
10727     \glxtrshorthyphenlong{##1}%
10728     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10729   }%
10730   \renewcommand*{\Glsxtrfullformat}[2]{%
10731     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10732   }%
10733   \renewcommand*{\Glsxtrfullplformat}[2]{%
10734     \glxtrshorthyphenlong{##1}%
10735     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10736   }%
10737 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10738 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10739 {%
10740   \renewcommand*{\CustomAbbreviationFields}{%
10741     name={\glxtrshortlongdescname},
10742     sort={\glxtrshortlongdescsort},
10743     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10744       \protect\glxtrfullsep{\the\glslabeltok}%
10745       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10746     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10747       \protect\glxtrfullsep{\the\glslabeltok}%
10748       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10749     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10750     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10751   }%

```

Unset the regular attribute if it has been set.

```

10752   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10753     \glshasattribute{\the\glslabeltok}{regular}%
10754     {%
10755       \glissetattribute{\the\glslabeltok}{regular}{false}%
10756     }%
10757   }%
10758 }%
10759 }%
10760 {%
10761   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10762 }

```

lsxtrshorthyphen `\glxtrshorthyphen{<short>}{<label>}{<insert>}`

Used by short-hyphen-postlong-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10763 \newcommand*{\glxtrshorthyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10764   {%
10765     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10766       \glsfirstabbrvhyphenfont{#1}%
10767     }%
10768 }

```

trposthyphenlong `\glxtrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the `<short>` part. This always uses the singular long form.

```

10769 \newcommand*{\glxtrposthyphenlong}[2]{%
10770   {%
10771     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{%
10772       \ifglxtrininsertinside{\glsfirstabrvhyphenfont{#2}}\else{#2}\fi
10773       \glxtrfullsep{#1}%
10774       \glxtrparen
10775       {\glsfirstlonghyphenfont{\glsentrylong{#1}\ifglxtrininsertinside{#2}\fi}%
10776       \ifglxtrininsertinside\else{#2}\fi
10777     }%
10778   }%
10779 }

```

`postlong-hyphen` Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10780 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10781 {%
10782   \renewcommand*{\CustomAbbreviationFields}{%
10783     name={\glxtrshortlongname},
10784     sort={\the\glsshorttok},
10785     first={\protect\glsfirstabrvhyphenfont{\the\glsshorttok}},%
10786     firstplural={\protect\glsfirstabrvhyphenfont{\the\glsshortpltok}},%
10787     plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
10788     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10789   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10790     \csdef{glxtrpostlink\glscategorylabel}{%
10791       \glxtrifwasfirstuse
10792       {%
10793         \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10794       }%
10795     }%

```

Put the insertion into the post-link:

```

10796       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10797     }%
10798   }%
10799   \glshasattribute{\the\glslabeltok}{regular}%
10800   {%
10801     \glissetattribute{\the\glslabeltok}{regular}{false}%
10802   }%
10803   {}%
10804 }%
10805 }%
10806 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10807 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10808 \renewcommand*{\glssabrvfont}[1]{\glssabrvhyphenfont{##1}}%
10809 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvhyphenfont{##1}}%

```

```

10810 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10811 \renewcommand*\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10812 \renewcommand*\glsxtrsubsequentfmt}[2]{%
10813   \glsabbrvfont{\glsaccessshort{##1}}%
10814 }%
10815 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
10816   \glsabbrvfont{\glsaccessshortpl{##1}}%
10817 }%
10818 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
10819   \glsabbrvfont{\Glsaccessshort{##1}}%
10820 }%
10821 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
10822   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10823 }%

```

First use full form:

```

10824 \renewcommand*\glsxtrfullformat}[2]{%
10825   \glsxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10826 }%
10827 \renewcommand*\glsxtrfullplformat}[2]{%
10828   \glsxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10829 }%
10830 \renewcommand*\Glsxtrfullformat}[2]{%
10831   \glsxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10832 }%
10833 \renewcommand*\Glsxtrfullplformat}[2]{%
10834   \glsxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10835 }%

```

In-line format. Commands like `\glsxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10836 \renewcommand*\glsxtrinlinefullformat}[2]{%
10837   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10838   \ifglsxtrininsertinside{##2}\fi}%
10839   \ifglsxtrininsertinside \else{##2}\fi
10840 }%
10841 \renewcommand*\glsxtrinlinefullplformat}[2]{%
10842   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10843   \ifglsxtrininsertinside{##2}\fi}%
10844   \ifglsxtrininsertinside \else{##2}\fi
10845 }%
10846 \renewcommand*\Glsxtrinlinefullformat}[2]{%
10847   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10848   \ifglsxtrininsertinside{##2}\fi}%
10849   \ifglsxtrininsertinside \else{##2}\fi
10850 }%
10851 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
10852   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10853   \ifglsxtrininsertinside{##2}\fi}%

```

```

10854     \ifglxtrinsertinside \else{##2}\fi
10855   }%
10856 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

10857 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10858 {%
10859   \renewcommand*{\CustomAbbreviationFields}{%
10860     name={\glxtrshortlongdescname},
10861     sort={\glxtrshortlongdescsort},%
10862     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10863     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10864     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10865     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10866   }%
10867   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10868     \csdef{glxtrpostlink\glscategorylabel}{%
10869       \glxtrifwasfirstuse
10870       {%
10871         \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10872       }%
10873       {%

```

Put the insertion into the post-link:

```

10874         \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10875       }%
10876     }%
10877     \glshasattribute{\the\glslabeltok}{regular}%
10878     {%
10879       \glssetattribute{\the\glslabeltok}{regular}{false}%
10880     }%
10881     {}%
10882   }%
10883 }%
10884 {%
10885   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10886 }

```

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

```

10887 \newcommand*{\glssabbrvonlyfont}{\glssabbrvdefaultfont}%

```

stabbrvonlyfont

```

10888 \newcommand*{\glsfirstabbrvonlyfont}{\glssabbrvonlyfont}%

```

glslongonlyfont

```

10889 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

```

rstlongonlyfont

```
10890 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10891 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
10892 \newcommand*{\glsxtronlyname}{%
```

```
10893   \protect\glsabbrvonlyfont{\the\glsshorttok}}%
```

```
10894 }
```

only-short-only

```
10895 \newabbreviationstyle{long-only-short-only}%
```

```
10896 {%
```

```
10897   \renewcommand*{\CustomAbbreviationFields}{%
```

```
10898     name={\glsxtronlyname},
```

```
10899     sort={\the\glsshorttok},
```

```
10900     first={\protect\glfirstlongonlyfont{\the\glslongtok}},%
```

```
10901     firstplural={\protect\glfirstlongonlyfont{\the\glslongpltok}},%
```

```
10902     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
```

```
10903     description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10904   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
10905     \glshasattribute{\the\glslabeltok}{regular}}%
```

```
10906   {%
```

```
10907     \glissetattribute{\the\glslabeltok}{regular}{false}}%
```

```
10908   }%
```

```
10909   {}%
```

```
10910 }%
```

```
10911 }%
```

```
10912 {%
```

```
10913   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
```

```
10914   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
```

```
10915   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
```

```
10916   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
```

```
10917   \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
10918   \renewcommand*{\glxtrfullformat}[2]{%
```

```
10919     \glfirstlongonlyfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}}%
```

```
10920   \ifglxtrininsertinside\else##2\fi
```

```
10921 }%
```

```
10922   \renewcommand*{\glxtrfullplformat}[2]{%
```

```
10923     \glfirstlongonlyfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}}%
```

```
10924   \ifglxtrininsertinside\else##2\fi
```

```
10925 }%
```

```
10926   \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

10927 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10928 \ifglxtrinsertinside\else##2\fi
10929 }%
10930 \renewcommand*{\Glsxtrfullplformat}[2]{%
10931 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10932 \ifglxtrinsertinside\else##2\fi
10933 }%

```

The inline full form does show the short form.

```

10934 \renewcommand*{\glxtrinlinefullformat}[2]{%
10935 \glsfirstlongonlyfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10936 \ifglxtrinsertinside\else##2\fi
10937 \glxtrfullsep{##1}%
10938 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshort{##1}}}%
10939 }%
10940 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10941 \glsfirstlongonlyfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10942 \ifglxtrinsertinside\else##2\fi
10943 \glxtrfullsep{##1}%
10944 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
10945 }%
10946 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10947 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10948 \ifglxtrinsertinside\else##2\fi
10949 \glxtrfullsep{##1}%
10950 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
10951 }%
10952 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10953 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10954 \ifglxtrinsertinside\else##2\fi
10955 \glxtrfullsep{##1}%
10956 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10957 }%
10958 }

```

xttronlydescsort

```

10959 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xttronlydescname

```

10960 \newcommand*{\glxtronlydescname}{%
10961 \protect\glslongfont{\the\glslongtok}%
10962 }

```

short-only-desc

```

10963 \newabbreviationstyle{long-only-short-only-desc}%
10964 {%
10965 \renewcommand*{\CustomAbbreviationFields}{%
10966 name={\glxtronlydescname},
10967 sort={\glxtronlydescsort},%

```

```

10968   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10969   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10970   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
10971   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
10972 }%

```

Unset the regular attribute if it has been set.

```

10973 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10974   \glshasattribute{\the\glslabeltok}{regular}}%
10975   {%
10976     \glssetattribute{\the\glslabeltok}{regular}{false}}%
10977   }%
10978   {}%
10979 }%
10980}%
10981{%
10982 \GlsXtrUseAbbrStyleFmts{long-only-short-only}}%
10983}

```

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:

```

10984 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```
10985 \renewcommand*{\markright}[1]{%
10986   \glxtrmarkhook
10987   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10988   \glxtrrestoremarkhook
10989 }
```

\markboth Save original definition:

```
10990 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10991 \renewcommand*{\markboth}[2]{%
10992   \glxtrmarkhook
10993   \@glxtr@org@markboth
10994   {\@glxtrinmark#1\@glxtrnotinmark}%
10995   {\@glxtrinmark#2\@glxtrnotinmark}%
10996   \glxtrrestoremarkhook
10997 }
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
10998 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
10999 \renewcommand*{\@starttoc}[1]{%
11000   \glxtrmarkhook
11001   \@glxtrinmark
11002   \@glxtr@org@@starttoc{#1}%
11003   \@glxtrnotinmark
11004   \glxtrrestoremarkhook
11005 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

glxtrRevertMarks

```
11006 \newcommand*{\glxtrRevertMarks}{%
11007   \let\markright\@glxtr@org@markright
11008   \let\markboth\@glxtr@org@markboth
11009   \let\@starttoc\@glxtr@org@@starttoc
11010 }
```

glxtrRevertTocMarks Just restores \@starttoc.

```
11011 \newcommand*{\glxtrRevertTocMarks}{%
11012   \let\@starttoc\@glxtr@org@@starttoc
11013 }
```

\glxtrifinmark

```
11014 \newcommand*{\glxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
11015 \newrobustcmd*{\@glsxtrinmark}{%
11016   \let\glsxtrifinmark\@firstoftwo
11017 }
```

glsxtrnotinmark

```
11018 \newrobustcmd*{\@glsxtrnotinmark}{%
11019   \let\glsxtrifinmark\@secondoftwo
11020 }
```

eorpdforheading

```
11021 \ifdef\texorpdfstring
11022 {
11023   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11024 }
11025 {
11026   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
11027 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11028 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
11029   \let\@glsxtr@org@MakeUppercase\MakeUppercase
11030   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11031   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11032   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11033   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11034   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11035   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11036   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11037   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11038   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11039   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11040   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11041   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11042   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11043   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11044   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11045   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11046   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11047   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11048   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11049   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11050   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11051   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11052   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```

11053 \let\glsxtrifinmark\@firstoftwo
11054 \let\MakeUppercase\MakeTextUppercase
11055 \let\glsxtrtitleorpdforheading\@thirdofthree
11056 \let\glsxtrtitleshort\glsxtrheadshort
11057 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11058 \let\Glsxtrtitleshort\Glsxtrheadshort
11059 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11060 \let\glsxtrtitlename\glsxtrheadname
11061 \let\Glsxtrtitlename\Glsxtrheadname
11062 \let\glsxtrtitletext\glsxtrheadtext
11063 \let\Glsxtrtitletext\Glsxtrheadtext
11064 \let\glsxtrtitleplural\glsxtrheadplural
11065 \let\Glsxtrtitleplural\Glsxtrheadplural
11066 \let\glsxtrtitlefirst\glsxtrheadfirst
11067 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11068 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11069 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11070 \let\glsxtrtitlelong\glsxtrheadlong
11071 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11072 \let\Glsxtrtitlelong\Glsxtrheadlong
11073 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11074 \let\glsxtrtitlefull\glsxtrheadfull
11075 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11076 \let\Glsxtrtitlefull\Glsxtrheadfull
11077 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11078 }

```

`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.)

```

11079 \newcommand*{\glsxtrrestoremarkhook}{%
11080 \let\glsxtrifinmark\@secondoftwo
11081 \let\MakeUppercase\@glsxtr@org@MakeUppercase
11082 \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
11083 \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
11084 \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
11085 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
11086 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
11087 \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
11088 \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
11089 \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
11090 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
11091 \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
11092 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
11093 \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
11094 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
11095 \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural

```

```

11096 \let\Glsxtrtitlefirstplural\@glxstr@org@Glsxtrtitlefirstplural
11097 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
11098 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
11099 \let\Glsxtrtitlelong\@glxstr@org@Glsxtrtitlelong
11100 \let\Glsxtrtitlelongpl\@glxstr@org@Glsxtrtitlelongpl
11101 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
11102 \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
11103 \let\Glsxtrtitlefull\@glxstr@org@Glsxtrtitlefull
11104 \let\Glsxtrtitlefullpl\@glxstr@org@Glsxtrtitlefullpl
11105 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glxstrheadshort Command used to display short form in the page header.

```

11106 \newcommand*{\glxstrheadshort}[1]{%
11107 \protect\NoCaseChange
11108 {%
11109 \gl@ifattribute{#1}{headuc}{true}%
11110 {%
11111 \GLSxtrshort[noindex,hyper=false]{#1}[]%
11112 }%
11113 {%
11114 \glxstrshort[noindex,hyper=false]{#1}[]%
11115 }%
11116 }%
11117 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11118 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11119 \glxstrshort[noindex,hyper=false]{#1}[]%
11120 }

```

glxstrheadshortpl 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.

```

11121 \newcommand*{\glxstrheadshortpl}[1]{%
11122 \protect\NoCaseChange
11123 {%
11124 \gl@ifattribute{#1}{headuc}{true}%
11125 {%
11126 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11127 }%
11128 {%
11129 \glxstrshortpl[noindex,hyper=false]{#1}[]%
11130 }%
11131 }%
11132 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11133 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11134   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11135 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11136 \newcommand*{\Glsxtrheadshort}[1]{%
11137   \protect\NoCaseChange
11138   {%
11139     \glsifattribute{#1}{headuc}{true}%
11140     {%
11141       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11142     }%
11143     {%
11144       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11145     }%
11146   }%
11147 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11148 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11149   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11150 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11151 \newcommand*{\Glsxtrheadshortpl}[1]{%
11152   \protect\NoCaseChange
11153   {%
11154     \glsifattribute{#1}{headuc}{true}%
11155     {%
11156       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11157     }%
11158     {%
11159       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11160     }%
11161   }%
11162 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11163 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11164   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11165 }
```

`\glxtrheadname` As above but for the name value.

```

11166 \newcommand*{\glxtrheadname}[1]{%
11167   \protect\NoCaseChange
11168   {%
11169     \gl@ifattribute{#1}{headuc}{true}%
11170     {%
11171       \GLSname[noindex,hyper=false]{#1}[]%
11172     }%
11173     {%
11174       \glSname[noindex,hyper=false]{#1}[]%
11175     }%
11176   }%
11177 }

```

glxtrtitlename Command to display name value in section title and table of contents.

```

11178 \newrobustcmd*{\glxtrtitlename}[1]{%
11179   \glSname[noindex,hyper=false]{#1}[]%
11180 }

```

\Glsxtrheadname First letter converted to upper case

```

11181 \newcommand*{\Glsxtrheadname}[1]{%
11182   \protect\NoCaseChange
11183   {%
11184     \gl@ifattribute{#1}{headuc}{true}%
11185     {%
11186       \GLSname[noindex,hyper=false]{#1}[]%
11187     }%
11188     {%
11189       \Glsname[noindex,hyper=false]{#1}[]%
11190     }%
11191   }%
11192 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11193 %\changes{1.21}{2017-11-03}{new}
11194 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11195   \Glsname[noindex,hyper=false]{#1}[]%
11196 }

```

\glxtrheadtext As above but for the text value.

```

11197 \newcommand*{\glxtrheadtext}[1]{%
11198   \protect\NoCaseChange
11199   {%
11200     \gl@ifattribute{#1}{headuc}{true}%
11201     {%
11202       \GLStext[noindex,hyper=false]{#1}[]%
11203     }%
11204     {%
11205       \glstext[noindex,hyper=false]{#1}[]%

```

```

11206 }%
11207 }%
11208 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

11209 \newrobustcmd*{\glsxtrtitletext}[1]{%
11210   \glstext[noindex,hyper=false]{#1}[]%
11211 }

```

\Glsxtrheadtext First letter converted to upper case

```

11212 \newcommand*{\Glsxtrheadtext}[1]{%
11213   \protect\NoCaseChange
11214   {%
11215     \glsifattribute{#1}{headuc}{true}%
11216     {%
11217       \GLStext[noindex,hyper=false]{#1}[]%
11218     }%
11219     {%
11220       \Glstext[noindex,hyper=false]{#1}[]%
11221     }%
11222   }%
11223 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

11224 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11225   \Glstext[noindex,hyper=false]{#1}[]%
11226 }

```

lsxtrheadplural As above but for the plural value.

```

11227 \newcommand*{\lsxtrheadplural}[1]{%
11228   \protect\NoCaseChange
11229   {%
11230     \glsifattribute{#1}{headuc}{true}%
11231     {%
11232       \GLSplural[noindex,hyper=false]{#1}[]%
11233     }%
11234     {%
11235       \glsplural[noindex,hyper=false]{#1}[]%
11236     }%
11237   }%
11238 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

11239 \newrobustcmd*{\lsxtrtitleplural}[1]{%
11240   \glsplural[noindex,hyper=false]{#1}[]%
11241 }

```

lsxtrheadplural Convert first letter to upper case.

```
11242 \newcommand*{\Glsxtrheadplural}[1]{%
11243   \protect\NoCaseChange
11244   {%
11245     \glsifattribute{#1}{headuc}{true}%
11246     {%
11247       \GLSplural[noindex,hyper=false]{#1}[]%
11248     }%
11249     {%
11250       \Glsplural[noindex,hyper=false]{#1}[]%
11251     }%
11252   }%
11253 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11254 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11255   \Glsplural[noindex,hyper=false]{#1}[]%
11256 }
```

glsxtrheadfirst As above but for the first value.

```
11257 \newcommand*{\glsxtrheadfirst}[1]{%
11258   \protect\NoCaseChange
11259   {%
11260     \glsifattribute{#1}{headuc}{true}%
11261     {%
11262       \GLSfirst[noindex,hyper=false]{#1}[]%
11263     }%
11264     {%
11265       \glsfirst[noindex,hyper=false]{#1}[]%
11266     }%
11267   }%
11268 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
11269 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
11270   \glsfirst[noindex,hyper=false]{#1}[]%
11271 }
```

Glsxtrheadfirst First letter converted to upper case

```
11272 \newcommand*{\Glsxtrheadfirst}[1]{%
11273   \protect\NoCaseChange
11274   {%
11275     \glsifattribute{#1}{headuc}{true}%
11276     {%
11277       \GLSfirst[noindex,hyper=false]{#1}[]%
11278     }%
11279     {%
```

```

11280     \Glsfirst[noindex,hyper=false]{#1}[]%
11281 }%
11282 }%
11283 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11284 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11285   \Glsfirst[noindex,hyper=false]{#1}[]%
11286 }

```

headfirstplural As above but for the firstplural value.

```

11287 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11288   \protect\NoCaseChange
11289   {%
11290     \glsifattribute{#1}{headuc}{true}%
11291     {%
11292       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11293     }%
11294     {%
11295       \glsfirstplural[noindex,hyper=false]{#1}[]%
11296     }%
11297   }%
11298 }

```

itlefirstplural Command to display firstplural value in section title and table of contents.

```

11299 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11300   \glsfirstplural[noindex,hyper=false]{#1}[]%
11301 }

```

headfirstplural First letter converted to upper case

```

11302 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11303   \protect\NoCaseChange
11304   {%
11305     \glsifattribute{#1}{headuc}{true}%
11306     {%
11307       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11308     }%
11309     {%
11310       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11311     }%
11312   }%
11313 }

```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11314 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11315   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11316 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```

11317 \newcommand*{\glxstrheadlong}[1]{%
11318   \protect\NoCaseChange
11319   {%
11320     \gl@ifattribute{#1}{headuc}{true}%
11321     {%
11322       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11323     }%
11324     {%
11325       \glxstrlong[noindex,hyper=false]{#1}[]%
11326     }%
11327   }%
11328 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

11329 \newrobustcmd*{\glxstrtitlelong}[1]{%
11330   \glxstrlong[noindex,hyper=false]{#1}[]%
11331 }
```

`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.

```

11332 \newcommand*{\glxtrheadlongpl}[1]{%
11333   \protect\NoCaseChange
11334   {%
11335     \gl@ifattribute{#1}{headuc}{true}%
11336     {%
11337       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11338     }%
11339     {%
11340       \glxstrlongpl[noindex,hyper=false]{#1}[]%
11341     }%
11342   }%
11343 }
```

`sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

11344 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
11345   \glxstrlongpl[noindex,hyper=false]{#1}[]%
11346 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

11347 \newcommand*{\Glsxtrheadlong}[1]{%
11348   \protect\NoCaseChange
11349   {%
11350     \gl@ifattribute{#1}{headuc}{true}%
11351     {%
11352       \GLSxtrlong[noindex,hyper=false]{#1}[]%

```

```

11353 }%
11354 {%
11355     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11356 }%
11357 }%
11358 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11359 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11360     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11361 }

```

lgsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11362 \newcommand*{\Glsxtrheadlongpl}[1]{%
11363     \protect\NoCaseChange
11364     {%
11365         \glsifattribute{#1}{headuc}{true}%
11366         {%
11367             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11368         }%
11369         {%
11370             \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11371         }%
11372     }%
11373 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11374 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11375     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11376 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11377 \newcommand*{\glsxtrheadfull}[1]{%
11378     \protect\NoCaseChange
11379     {%
11380         \glsifattribute{#1}{headuc}{true}%
11381         {%
11382             \GLSxtrfull[noindex,hyper=false]{#1}[]%
11383         }%
11384         {%
11385             \glsxtrfull[noindex,hyper=false]{#1}[]%
11386         }%
11387     }%
11388 }

```

`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
11389 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11390   \glsxtrfull[noindex,hyper=false]{#1}[]%
11391 }
```

`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.

```
11392 \newcommand*{\glsxtrheadfullpl}[1]{%
11393   \protect\NoCaseChange
11394   {%
11395     \glsifattribute{#1}{headuc}{true}%
11396     {%
11397       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11398     }%
11399     {%
11400       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11401     }%
11402   }%
11403 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
11404 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
11405   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11406 }
```

`\GLsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
11407 \newcommand*{\GLsxtrheadfull}[1]{%
11408   \protect\NoCaseChange
11409   {%
11410     \glsifattribute{#1}{headuc}{true}%
11411     {%
11412       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11413     }%
11414     {%
11415       \GLsxtrfull[noindex,hyper=false]{#1}[]%
11416     }%
11417   }%
11418 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11419 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11420   \GLsxtrfull[noindex,hyper=false]{#1}[]%
11421 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11422 \newcommand*{\Glsxtrheadfullpl}[1]{%
11423   \protect\NoCaseChange
11424   {%
11425     \glsifattribute{#1}{headuc}{true}%
11426     {%
11427       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11428     }%
11429     {%
11430       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11431     }%
11432   }%
11433 }

```

`\sxttrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11434 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11435   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11436 }

```

`\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.

```

11437 \ifdef\texorpdfstring
11438 {
11439   \newcommand*{\glsfmtshort}[1]{%
11440     \texorpdfstring
11441       {\glsxtrtitleshort{#1}}%
11442       {\glsentryshort{#1}}%
11443   }
11444 }
11445 {
11446   \newcommand*{\glsfmtshort}[1]{%
11447     \glsxtrtitleshort{#1}}
11448 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11449 \ifdef\texorpdfstring
11450 {
11451   \newcommand*{\glsfmtshortpl}[1]{%
11452     \texorpdfstring
11453       {\glsxtrtitleshortpl{#1}}%
11454       {\glsentryshortpl{#1}}%
11455   }
11456 }
11457 {
11458   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11459 \glstrtitleshortpl{#1}}
11460 }

```

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).

```

11461 \ifdef\texorpdfstring
11462 {
11463   \newcommand*\Glsfmtshort[1]{%
11464     \texorpdfstring
11465       {\Glsxtrtitleshort{#1}}%
11466       {\glsentryshort{#1}}%
11467   }
11468 }
11469 {
11470   \newcommand*\Glsfmtshort[1]{%
11471     \Glsxtrtitleshort{#1}}
11472 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11473 \ifdef\texorpdfstring
11474 {
11475   \newcommand*\Glsfmtshortpl[1]{%
11476     \texorpdfstring
11477       {\Glsxtrtitleshortpl{#1}}%
11478       {\glsentryshortpl{#1}}%
11479   }
11480 }
11481 {
11482   \newcommand*\Glsfmtshortpl[1]{%
11483     \Glsxtrtitleshortpl{#1}}
11484 }

```

`\glsfmtname` As above but for the name value.

```

11485 \ifdef\texorpdfstring
11486 {
11487   \newcommand*\glsfmtname[1]{%
11488     \texorpdfstring
11489       {\glsxtrtitlename{#1}}%
11490       {\glsentryname{#1}}%
11491   }
11492 }
11493 {
11494   \newcommand*\glsfmtname[1]{%
11495     \glsxtrtitlename{#1}}
11496 }

```

`\Glsfmtname` First letter converted to upper case.

```

11497 \ifdef\texorpdfstring
11498 {
11499   \newcommand*{\Glsfmtname}[1]{%
11500     \texorpdfstring
11501     {\Glsxtrtitlename{#1}}%
11502     {\glsentryname{#1}}%
11503   }
11504 }
11505 {
11506   \newcommand*{\Glsfmtname}[1]{%
11507     \Glsxtrtitlename{#1}}
11508 }

```

`\glsfmttext` As above but for the text value.

```

11509 \ifdef\texorpdfstring
11510 {
11511   \newcommand*{\glsfmttext}[1]{%
11512     \texorpdfstring
11513     {\glsxtrtitletext{#1}}%
11514     {\glsentrytext{#1}}%
11515   }
11516 }
11517 {
11518   \newcommand*{\glsfmttext}[1]{%
11519     \glsxtrtitletext{#1}}
11520 }

```

`\Glsfmttext` First letter converted to upper case.

```

11521 \ifdef\texorpdfstring
11522 {
11523   \newcommand*{\Glsfmttext}[1]{%
11524     \texorpdfstring
11525     {\Glsxtrtitletext{#1}}%
11526     {\glsentrytext{#1}}%
11527   }
11528 }
11529 {
11530   \newcommand*{\Glsfmttext}[1]{%
11531     \Glsxtrtitletext{#1}}
11532 }

```

`\glsfmtplural` As above but for the plural value.

```

11533 \ifdef\texorpdfstring
11534 {
11535   \newcommand*{\glsfmtplural}[1]{%
11536     \texorpdfstring
11537     {\glsxtrtitleplural{#1}}%
11538     {\glsentryplural{#1}}%
11539   }

```

```

11540 }
11541 {
11542   \newcommand*{\glsfmtplural}[1]{%
11543     \glsxtrtitleplural{#1}}
11544 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11545 \ifdef\texorpdfstring
11546 {
11547   \newcommand*{\Glsfmtplural}[1]{%
11548     \texorpdfstring
11549     {\Glsxtrtitleplural{#1}}%
11550     {\glsentryplural{#1}}%
11551   }
11552 }
11553 {
11554   \newcommand*{\Glsfmtplural}[1]{%
11555     \Glsxtrtitleplural{#1}}
11556 }

```

`\glsfmtfirst` As above but for the first value.

```

11557 \ifdef\texorpdfstring
11558 {
11559   \newcommand*{\glsfmtfirst}[1]{%
11560     \texorpdfstring
11561     {\glsxtrtitlefirst{#1}}%
11562     {\glsentryfirst{#1}}%
11563   }
11564 }
11565 {
11566   \newcommand*{\glsfmtfirst}[1]{%
11567     \glsxtrtitlefirst{#1}}
11568 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11569 \ifdef\texorpdfstring
11570 {
11571   \newcommand*{\Glsfmtfirst}[1]{%
11572     \texorpdfstring
11573     {\Glsxtrtitlefirst{#1}}%
11574     {\glsentryfirst{#1}}%
11575   }
11576 }
11577 {
11578   \newcommand*{\Glsfmtfirst}[1]{%
11579     \Glsxtrtitlefirst{#1}}
11580 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11581 \ifdef\texorpdfstring
11582 {
11583   \newcommand*\glsfmtfirstpl}[1]{%
11584     \texorpdfstring
11585     {\glsxtrtitlefirstplural{#1}}%
11586     {\glsentryfirstplural{#1}}%
11587   }
11588 }
11589 {
11590   \newcommand*\glsfmtfirstpl}[1]{%
11591     \glsxtrtitlefirstplural{#1}}
11592 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11593 \ifdef\texorpdfstring
11594 {
11595   \newcommand*\Glsfmtfirstpl}[1]{%
11596     \texorpdfstring
11597     {\Glsxtrtitlefirstplural{#1}}%
11598     {\glsentryfirstplural{#1}}%
11599   }
11600 }
11601 {
11602   \newcommand*\Glsfmtfirstpl}[1]{%
11603     \Glsxtrtitlefirstplural{#1}}
11604 }

```

`\glsfmtlong` As above but for the long value.

```

11605 \ifdef\texorpdfstring
11606 {
11607   \newcommand*\glsfmtlong}[1]{%
11608     \texorpdfstring
11609     {\glsxtrtitlelong{#1}}%
11610     {\glsentrylong{#1}}%
11611   }
11612 }
11613 {
11614   \newcommand*\glsfmtlong}[1]{%
11615     \glsxtrtitlelong{#1}}
11616 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11617 \ifdef\texorpdfstring
11618 {
11619   \newcommand*\Glsfmtlong}[1]{%
11620     \texorpdfstring
11621     {\Glsxtrtitlelong{#1}}%
11622     {\glsentrylong{#1}}%
11623   }

```

```

11624 }
11625 {
11626   \newcommand*{\Glsfmtlong}[1]{%
11627     \Glsxtrtitlelong{#1}}
11628 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11629 \ifdef\texorpdfstring
11630 {
11631   \newcommand*{\glsfmtlongpl}[1]{%
11632     \texorpdfstring
11633     {\Glsxtrtitlelongpl{#1}}%
11634     {\Glsentrylongpl{#1}}%
11635   }
11636 }
11637 {
11638   \newcommand*{\glsfmtlongpl}[1]{%
11639     \Glsxtrtitlelongpl{#1}}
11640 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11641 \ifdef\texorpdfstring
11642 {
11643   \newcommand*{\Glsfmtlongpl}[1]{%
11644     \texorpdfstring
11645     {\Glsxtrtitlelongpl{#1}}%
11646     {\Glsentrylongpl{#1}}%
11647   }
11648 }
11649 {
11650   \newcommand*{\Glsfmtlongpl}[1]{%
11651     \Glsxtrtitlelongpl{#1}}
11652 }

```

`\glsfmtfull` In-line full format.

```

11653 \ifdef\texorpdfstring
11654 {
11655   \newcommand*{\glsfmtfull}[1]{%
11656     \texorpdfstring
11657     {\Glsxtrtitlefull{#1}}%
11658     {\Glsxtrinlinefullformat{#1}{}}%
11659   }
11660 }
11661 {
11662   \newcommand*{\glsfmtfull}[1]{%
11663     \Glsxtrtitlefull{#1}}
11664 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11665 \ifdef\txorpdfstring
11666 {
11667   \newcommand*{\Glsfmtfull}[1]{%
11668     \txorpdfstring
11669     {\Glsxtrtitlefull{#1}}%
11670     {\Glsxtrinlinefullformat{#1}{}}%
11671   }
11672 }
11673 {
11674   \newcommand*{\Glsfmtfull}[1]{%
11675     \Glsxtrtitlefull{#1}}
11676 }

```

`\glsfmtfullpl` In-line full plural format.

```

11677 \ifdef\txorpdfstring
11678 {
11679   \newcommand*{\glsfmtfullpl}[1]{%
11680     \txorpdfstring
11681     {\glsxtrtitlefullpl{#1}}%
11682     {\glsxtrinlinefullplformat{#1}{}}%
11683   }
11684 }
11685 {
11686   \newcommand*{\glsfmtfullpl}[1]{%
11687     \glsxtrtitlefullpl{#1}}
11688 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11689 \ifdef\txorpdfstring
11690 {
11691   \newcommand*{\Glsfmtfullpl}[1]{%
11692     \txorpdfstring
11693     {\Glsxtrtitlefullpl{#1}}%
11694     {\Glsxtrinlinefullplformat{#1}{}}%
11695   }
11696 }
11697 {
11698   \newcommand*{\Glsfmtfullpl}[1]{%
11699     \Glsxtrtitlefullpl{#1}}
11700 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11701 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```

```

11702 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11703 }

```

sariesExtraLang

```

11704 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11705   \ProvidesFile{glossariesxtr-#1.ldf}%
11706 }

```

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.

```

11707 \newcommand{\glxtr@loaddialect}{%
11708   \IfTrackedLanguageFileExists{\this@dialect}%
11709   {glossariesxtr-}% prefix
11710   {.ldf}%
11711   {%
11712     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11713   }%
11714   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11715 \@glxtrdialecthook
11716 }

```

```

11717 \ifpackageloaded{tracklang}
11718 {%
11719   \AnyTrackedLanguages
11720   {%
11721     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11722   }%
11723   {}%
11724 }
11725 {}

```

Load `glossaries-extra-stylemods` if required.

```

11726 \@glxtr@redefstyles

```

and set the style:

```

11727 \@glxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11728 \NeedsTeXFormat{LaTeX2e}
11729 \ProvidesPackage{glossaries-extra-bib2gls}[2018/08/13 v1.35 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11730 \newcommand*{\glshex}{\string\u}

```

`\glscapturedgroup`

```

11731 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```

11732 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11733   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11734 }

```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11735 \newcommand*{\GlsXtrProvideCommand}{\providecommand}

```

`\GlsXtrWrglossaryLocation` For use with indexcounter and bib2gls.

```

11736 \newcommand*{\GlsXtrWrglossaryLocation}[2]{#1}

```

`\GlsXtrIndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with indexcounter and bib2gls.

```

11737 \ifdef\hyperref
11738 {%
11739   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
11740     \GlsXtrIfHasField{indexcounter}{#2}%
11741     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11742     {#1}%
11743   }
11744 }
11745 {
11746   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
11747 }

```

`\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.

```

11748 \newcommand*{\GlsXtrDualField}{dual}

```

sXtrDualBackLink

```
\GlsXtrDualBackLink{<text>}{<label>}
```

Adds a hyperlink to the dual entry.

```
11749 \newcommand*{\GlsXtrDualBackLink}[2]{%
11750   \glstrifhasfield{\GlsXtrDualField}{#2}%
11751   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11752   {#2}%
11753 }
```

TeXEntryAliases

Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
11754 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
11755   article=bibtexentry,
11756   book=bibtexentry,
11757   booklet=bibtexentry,
11758   conference=bibtexentry,
11759   inbook=bibtexentry,
11760   incollection=bibtexentry,
11761   inproceedings=bibtexentry,
11762   manual=bibtexentry,
11763   mastersthesis=bibtexentry,
11764   misc=bibtexentry,
11765   phdthesis=bibtexentry,
11766   proceedings=bibtexentry,
11767   techreport=bibtexentry,
11768   unpublished=bibtexentry
11769 }
```

ideBibTeXFields

Convenient shortcut to define the standard BibTeX fields.

```
11770 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11771   \glsaddstoragekey{address}{\glstrbibaddress}%
11772   \glsaddstoragekey{author}{\glstrbibauthor}%
11773   \glsaddstoragekey{booktitle}{\glstrbibbooktitle}%
11774   \glsaddstoragekey{chapter}{\glstrbibchapter}%
11775   \glsaddstoragekey{edition}{\glstrbibedition}%
11776   \glsaddstoragekey{howpublished}{\glstrbibhowpublished}%
11777   \glsaddstoragekey{institution}{\glstrbibinstitution}%
11778   \glsaddstoragekey{journal}{\glstrbibjournal}%
11779   \glsaddstoragekey{month}{\glstrbibmonth}%
11780   \glsaddstoragekey{note}{\glstrbibnote}%
11781   \glsaddstoragekey{number}{\glstrbibnumber}%
11782   \glsaddstoragekey{organization}{\glstrbiborganization}%
11783   \glsaddstoragekey{pages}{\glstrbibpages}%
11784   \glsaddstoragekey{publisher}{\glstrbibpublisher}%
11785   \glsaddstoragekey{school}{\glstrbibschooll}%
11786   \glsaddstoragekey{series}{\glstrbibseries}%
11787   \glsaddstoragekey{title}{\glstrbibtitle}%

```

```

11788 \glsaddstoragekey{bibtextype}{-}{\glsxtrbibtype}%
11789 \glsaddstoragekey{volume}{-}{\glsxtrbibvolume}%
11790 }

```

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.

\backslash Alpha

```
11791 \providecommand*{\Alpha}{\mathrm{A}}
```

\backslash Beta

```
11792 \providecommand*{\Beta}{\mathrm{B}}
```

\backslash Epsilon

```
11793 \providecommand*{\Epsilon}{\mathrm{E}}
```

\backslash Zeta

```
11794 \providecommand*{\Zeta}{\mathrm{Z}}
```

\backslash Eta

```
11795 \providecommand*{\Eta}{\mathrm{H}}
```

\backslash Iota

```
11796 \providecommand*{\Iota}{\mathrm{I}}
```

\backslash Kappa

```
11797 \providecommand*{\Kappa}{\mathrm{K}}
```

\backslash Mu

```
11798 \providecommand*{\Mu}{\mathrm{M}}
```

\backslash Nu

```
11799 \providecommand*{\Nu}{\mathrm{N}}
```

\backslash Omicron

```
11800 \providecommand*{\Omicron}{\mathrm{O}}
```

\backslash Rho

```
11801 \providecommand*{\Rho}{\mathrm{P}}
```

\backslash Tau

```
11802 \providecommand*{\Tau}{\mathrm{T}}
```

```

\Chi
11803 \providecommand*\Chi{\mathrm{X}}

\Digamma
11804 \providecommand*\Digamma{\mathrm{F}}

\omicron
11805 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11806 \@ifpackageloaded{upgreek}%
11807 {

\Upalpha
11808   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11809   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11810   \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11811   \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11812   \providecommand*\Upeta{\mathrm{H}}

\Upiota
11813   \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11814   \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11815   \providecommand*\Upmu{\mathrm{M}}

\Upnu
11816   \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11817   \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11818   \providecommand*\Uprho{\mathrm{P}}

```

`\Uptau`

```
11819 \providecommand*\Uptau{\mathrm{T}}
```

`\Upchi`

```
11820 \providecommand*\Upchi{\mathrm{X}}
```

`\upomicron`

```
11821 \providecommand*\upomicron{\mathrm{o}}
```

```
11822 }%
```

```
11823 {}% upgreek.sty not loaded
```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-⟨tag⟩.ldf` (where `⟨tag⟩` identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of `⟨tag⟩`. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxtrcontrolrules
; \glxtrspacerules
; \glxtrnonprintablerules
; \glxtrcombiningdiacriticrules
, \glxtrhyphenrules
< \glxtrgeneralpuncrules
< \glxtrdigitrules
< \glxtrfractionrules
< \glxtrGeneralLatinIVrules
< \glxtrMathItalicGreekIrules
}
```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they’ve been made active.

```
11824 \newcommand*\glxtrcontrolrules{%
```

```
11825 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
```

```
11826 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
```

```
11827 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
```

```
11828 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
```

```
11829 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
```

```
11830 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
```

```

11831 0010\string'\string=\glshex 0011
11832 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11833 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11834 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11835 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11836 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11837 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11838 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11839 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11840 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11841 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11842 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11843 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11844 }

```

lsxtrspacerules These are space characters.

```

11845 \newcommand*{\glsxtrspacerules}{%
11846 \string' \string'\string;
11847 \string'\glshex 00A0\string'\string;
11848 \string'\glshex 2000\string'\string;
11849 \string'\glshex 2001\string'\string;
11850 \string'\glshex 2002\string'\string;
11851 \string'\glshex 2003\string'\string;
11852 \string'\glshex 2004\string'\string;
11853 \string'\glshex 2005\string'\string;
11854 \string'\glshex 2006\string'\string;
11855 \string'\glshex 2007\string'\string;
11856 \string'\glshex 2008\string'\string;
11857 \string'\glshex 2009\string'\string;
11858 \string'\glshex 200A\string'\string;
11859 \string'\glshex 3000\string'
11860 }

```

nonprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11861 \newcommand*{\glsxtrnonprintablerules}{%
11862 \string'\glshex FEFF\string'\string;
11863 \string'\glshex 000A\string'\string;
11864 \string'\glshex 0009\string'\string;
11865 \string'\glshex 000C\string'\string;
11866 \string'\glshex 000B\string'
11867 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

11868 \newcommand*{\glsxtrcombiningdiacriticrules}{%
11869 \glsxtrcombiningdiacriticIrules\string;
11870 \glsxtrcombiningdiacriticIIrules\string;
11871 \glsxtrcombiningdiacriticIIIrules\string;
11872 \glsxtrcombiningdiacriticIVrules
11873 }

```

diacriticIrules First set of combining diacritic marks.

```
11874 \newcommand*{\glxtrcombingdiacriticIrules}{%
11875 \glshex 0301\string;% combining acute
11876 \glshex 0300\string;% combining grave
11877 \glshex 0306\string;% combining breve
11878 \glshex 0302\string;% combining circumflex
11879 \glshex 030C\string;% combining caron
11880 \glshex 030A\string;% combining ring
11881 \glshex 030D\string;% combining vertical line above
11882 \glshex 0308\string;% combining diaeresis
11883 \glshex 030B\string;% combining double acute
11884 \glshex 0303\string;% combining tilde
11885 \glshex 0307\string;% combining dot above
11886 \glshex 0304\string;% combining macron
11887 }
```

diacriticIIrules Second set of combining diacritic marks.

```
11888 \newcommand*{\glxtrcombingdiacriticIIrules}{%
11889 \glshex 0337\string;% combining short solidus overlay
11890 \glshex 0327\string;% combining cedilla
11891 \glshex 0328\string;% combining ogonek
11892 \glshex 0323\string;% combining dot below
11893 \glshex 0332\string;% combining low line
11894 \glshex 0305\string;% combining overline
11895 \glshex 0309\string;% combining hook above
11896 \glshex 030E\string;% combining double vertical line above
11897 \glshex 030F\string;% combining double grave accent
11898 \glshex 0310\string;% combining candrabindu
11899 \glshex 0311\string;% combining inverted breve
11900 \glshex 0312\string;% combining turned comma above
11901 \glshex 0313\string;% combining comma above
11902 \glshex 0314\string;% combining reversed comma above
11903 \glshex 0315\string;% combining comma above right
11904 \glshex 0316\string;% combining grave accent below
11905 \glshex 0317\string;% combining acute accent below
11906 }
```

diacriticIIIrules Third set of combining diacritic marks.

```
11907 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
11908 \glshex 0318\string;% combining left tack below
11909 \glshex 0319\string;% combining right tack below
11910 \glshex 031A\string;% combining left angle above
11911 \glshex 031B\string;% combining horn
11912 \glshex 031C\string;% combining left half ring below
11913 \glshex 031D\string;% combining up tack below
11914 \glshex 031E\string;% combining down tack below
11915 \glshex 031F\string;% combining plus sign below
11916 \glshex 0320\string;% combining minus sign below
11917 \glshex 0321\string;% combining palatalized hook below
```

11918 \glshex 0322\string;% combining retroflex hook below
 11919 \glshex 0324\string;% combining diaresis below
 11920 \glshex 0325\string;% combining ring below
 11921 \glshex 0326\string;% combining comma below
 11922 \glshex 0329\string;% combining vertical line below
 11923 \glshex 032A\string;% combining bridge below
 11924 \glshex 032B\string;% combining inverted double arch below
 11925 \glshex 032C\string;% combining caron below
 11926 \glshex 032D\string;% combining circumflex accent below
 11927 \glshex 032E\string;% combining breve below
 11928 \glshex 032F\string;% combining inverted breve below
 11929 \glshex 0330\string;% combining tilde below
 11930 \glshex 0331\string;% combining macron below
 11931 \glshex 0333\string;% combining double low line
 11932 \glshex 0334\string;% combining tilde overlay
 11933 \glshex 0335\string;% combining short stroke overlay
 11934 \glshex 0336\string;% combining long stroke overlay
 11935 \glshex 0338\string;% combining long solidus overlay
 11936 \glshex 0339\string;% combining combining right half ring below
 11937 \glshex 033A\string;% combining inverted bridge below
 11938 \glshex 033B\string;% combining square below
 11939 \glshex 033C\string;% combining seagull below
 11940 \glshex 033D\string;% combining x above
 11941 \glshex 033E\string;% combining vertical tilde
 11942 \glshex 033F\string;% combining double overline
 11943 \glshex 0342\string;% combining Greek perispomeni
 11944 \glshex 0344\string;% combining Greek dialytika tonos
 11945 \glshex 0345\string;% combining Greek ypogegrammeni
 11946 \glshex 0360\string;% combining double tilde
 11947 \glshex 0361\string;% combining double inverted breve
 11948 \glshex 0483\string;% combining Cyrillic titlo
 11949 \glshex 0484\string;% combining Cyrillic palatalization
 11950 \glshex 0485\string;% combining Cyrillic dasia pneumata
 11951 \glshex 0486% combining Cyrillic psili pneumata
 11952 }

iacriticIVrules Fourth set of combining diacritic marks.

11953 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
 11954 \glshex 20D0\string;% combining left harpoon above
 11955 \glshex 20D1\string;% combining right harpoon above
 11956 \glshex 20D2\string;% combining long vertical line overlay
 11957 \glshex 20D3\string;% combining short vertical line overlay
 11958 \glshex 20D4\string;% combining anticlockwise arrow above
 11959 \glshex 20D5\string;% combining clockwise arrow above
 11960 \glshex 20D6\string;% combining left arrow above
 11961 \glshex 20D7\string;% combining right arrow above
 11962 \glshex 20D8\string;% combining ring overlay
 11963 \glshex 20D9\string;% combining clockwise ring overlay
 11964 \glshex 20DA\string;% combining anticlockwise ring overlay

```

11965 \glshex 20DB\string;% combining three dots above
11966 \glshex 20DC\string;% combining four dots above
11967 \glshex 20DD\string;% combining enclosing circle
11968 \glshex 20DE\string;% combining enclosing square
11969 \glshex 20DF\string;% combining enclosing diamond
11970 \glshex 20E0\string;% combining enclosing circle backslash
11971 \glshex 20E1% combining left right arrow above
11972 }

```

sxtrhyphenrules Hyphens.

```

11973 \newcommand*{\glsxtrhyphenrules}{%
11974 \string'\string-\string'\string;% ASCII hyphen
11975 \glshex 00AD\string;% soft hyphen
11976 \glshex 2010\string;% hyphen
11977 \glshex 2011\string;% non-breaking hyphen
11978 \glshex 2012\string;% figure dash
11979 \glshex 2013\string;% en dash
11980 \glshex 2014\string;% em dash
11981 \glshex 2015\string;% horizontal bar
11982 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11983 }

```

eneralpuncrules General punctuation.

```

11984 \newcommand*{\glsxtrgeneralpuncrules}{%
11985 \glsxtrgeneralpuncIrules
11986 \string<\glsxtrcurrencyrules
11987 \string<\glsxtrgeneralpuncIIrules
11988 }

```

neralpuncIrules First set of general punctuation.

```

11989 \newcommand*{\glsxtrgeneralpuncIrules}{%
11990 \string'\glshex 005F\string'% underscore
11991 \string<\glshex 00AF% macron
11992 \string<\string'\glshex 002C\string'% comma
11993 \string<\string'\glshex 003B\string'% semi-colon
11994 \string<\string'\glshex 003A\string'% colon
11995 \string<\string'\glshex 0021\string'% exclamation mark
11996 \string<\glshex 00A1% inverted exclamation mark
11997 \string<\string'\glshex 003F\string'% question mark
11998 \string<\glshex 00BF% inverted question mark
11999 \string<\string'\glshex 002F\string'% solidus
12000 \string<\string'\glshex 002E\string'% full stop
12001 \string<\glshex 00B4% acute accent
12002 \string<\string'\glshex 0060\string'% grave accent
12003 \string<\string'\glshex 005E\string'% circumflex accent
12004 \string<\glshex 00A8% diaeresis
12005 \string<\string'\glshex 007E\string'% tilde
12006 \string<\glshex 00B7% middle dot
12007 \string<\glshex 00B8% cedilla

```

```

12008 \string<\string'\glshex 0027\string'% straight apostrophe
12009 \string<\string'\glshex 0022\string'% straight double quote
12010 \string<\glshex 00AB% left guillemet
12011 \string<\glshex 00BB% right guillemet
12012 \string<\string'\glshex 0028\string'% left parenthesis
12013 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
12014 \string<\string'\glshex 0029\string'% right parenthesis
12015 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
12016 \string<\string'\glshex 005B\string'% left square bracket
12017 \string<\string'\glshex 005D\string'% right square bracket
12018 \string<\string'\glshex 007B\string'% left curly bracket
12019 \string<\string'\glshex 007D\string'% right curly bracket
12020 \string<\glshex 00A7% section sign
12021 \string<\glshex 00B6% pilcrow sign
12022 \string<\glshex 00A9% copyright sign
12023 \string<\glshex 00AE% registered sign
12024 \string<\string'\glshex 0040\string'% at sign
12025 }

```

trcurrencyrules General punctuation.

```

12026 \newcommand*{\glxtrcurrencyrules}{%
12027 \glshex 00A4% currency sign
12028 \string<\glshex 0E3F% Thai currency symbol baht
12029 \string<\glshex 00A2% cent sign
12030 \string<\glshex 20A1% colon sign
12031 \string<\glshex 20A2% cruzeiro sign
12032 \string<\string'\glshex 0024\string'% dollar sign
12033 \string<\glshex 20AB% dong sign
12034 \string<\glshex 20AC% euro sign
12035 \string<\glshex 20A3% French franc sign
12036 \string<\glshex 20A4% lira sign
12037 \string<\glshex 20A5% mill sign
12038 \string<\glshex 20A6% naira sign
12039 \string<\glshex 20A7% peseta sign
12040 \string<\glshex 00A3% pound sign
12041 \string<\glshex 20A8% rupee sign
12042 \string<\glshex 20AA% new sheqel sign
12043 \string<\glshex 20A9% won sign
12044 \string<\glshex 00A5% yen sign
12045 }

```

eralpuncIIrules Second set of general punctuation.

```

12046 \newcommand*{\glxtrgeneralpuncIIrules}{%
12047 \string'\glshex 002A\string'% asterisk
12048 \string<\string'\glshex 005C\string'% backslash
12049 \string<\string'\glshex 0026\string'% ampersand
12050 \string<\string'\glshex 0023\string'% hash sign
12051 \string<\string'\glshex 0025\string'% percent sign
12052 \string<\string'\glshex 002B\string'% plus sign

```

```

12053 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12054 \string<\glshex 00B1% plus-minus sign
12055 \string<\glshex 00F7% division sign
12056 \string<\glshex 00D7% multiplication sign
12057 \string<\string'\glshex 003C\string'% less-than sign
12058 \string<\string'\glshex 003D\string'% equals sign
12059 \string<\string'\glshex 003E\string'% greater-than sign
12060 \string<\glshex 00AC% not sign
12061 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12062 \string<\glshex 00A6% broken bar
12063 \string<\glshex 00B0% degree sign
12064 \string<\glshex 00B5% micron sign
12065 }

```

eralLatinIrules Basic Latin alphabet.

```

12066 \newcommand*{\glxtrGeneralLatinIrules}{%
12067 \glxtrLatinA
12068 \string<b,B%
12069 \string<c,C%
12070 \string<d,D%
12071 \string<\glxtrLatinE
12072 \string<f,F%
12073 \string<g,G%
12074 \string<\glxtrLatinH
12075 \string<\glxtrLatinI
12076 \string<j,J%
12077 \string<\glxtrLatinK
12078 \string<\glxtrLatinL
12079 \string<\glxtrLatinM
12080 \string<\glxtrLatinN
12081 \string<\glxtrLatinO
12082 \string<\glxtrLatinP
12083 \string<q,Q%
12084 \string<r,R%
12085 \string<\glxtrLatinS
12086 \string<\glxtrLatinT
12087 \string<u,U%
12088 \string<v,V%
12089 \string<w,W%
12090 \string<\glxtrLatinX
12091 \string<y,Y%
12092 \string<z,Z
12093 }

```

ralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12094 \newcommand*{\glxtrGeneralLatinIIrules}{%
12095 \glxtrLatinA
12096 \string<b,B%
12097 \string<c,C%

```

```

12098 \string<d,D%
12099 \string<\glxtrLatinEth
12100 \string<\glxtrLatinE
12101 \string<f,F%
12102 \string<g,G%
12103 \string<\glxtrLatinH
12104 \string<\glxtrLatinI
12105 \string<j,J%
12106 \string<\glxtrLatinK
12107 \string<\glxtrLatinL
12108 \string<\glxtrLatinM
12109 \string<\glxtrLatinN
12110 \string<\glxtrLatinO
12111 \string<\glxtrLatinP
12112 \string<q,Q%
12113 \string<r,R%
12114 \string<\glxtrLatinS
12115 \string& SS \string, \glxtrLatinEszettSs
12116 \string<\glxtrLatinT
12117 \string<u,U%
12118 \string<v,V%
12119 \string<w,W%
12120 \string<\glxtrLatinX
12121 \string<y,Y%
12122 \string<z,Z%
12123 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12124 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12125 \glxtrLatinA
12126 \string<b,B%
12127 \string<c,C%
12128 \string<d,D%
12129 \string<\glxtrLatinEth
12130 \string<\glxtrLatinE
12131 \string<f,F%
12132 \string<g,G%
12133 \string<\glxtrLatinH
12134 \string<\glxtrLatinI
12135 \string<j,J%
12136 \string<\glxtrLatinK
12137 \string<\glxtrLatinL
12138 \string<\glxtrLatinM
12139 \string<\glxtrLatinN
12140 \string<\glxtrLatinO
12141 \string<\glxtrLatinP
12142 \string<q,Q%
12143 \string<r,R%
12144 \string<\glxtrLatinS

```

```

12145 \string& SZ, \glxtrLatinEszettSz
12146 \string<\glxtrLatinT
12147 \string<u,U%
12148 \string<v,V%
12149 \string<w,W%
12150 \string<\glxtrLatinX
12151 \string<y,Y%
12152 \string<z,Z%
12153 }

```

GeneralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12154 \newcommand*{\glxtrGeneralLatinIVrules}{%
12155 \glxtrLatinA
12156 \string& AE , \glxtrLatinAELigature
12157 \string<b,B%
12158 \string<c,C%
12159 \string<d,D%
12160 \string<\glxtrLatinEth
12161 \string<\glxtrLatinE
12162 \string<f,F%
12163 \string<g,G%
12164 \string<\glxtrLatinH
12165 \string<\glxtrLatinI
12166 \string<j,J%
12167 \string<\glxtrLatinK
12168 \string<\glxtrLatinL
12169 \string<\glxtrLatinM
12170 \string<\glxtrLatinN
12171 \string<\glxtrLatinO
12172 \string& OE , \glxtrLatinOELigature
12173 \string<\glxtrLatinP
12174 \string<q,Q%
12175 \string<r,R%
12176 \string<\glxtrLatinS
12177 \string& SS , \glxtrLatinEszettSs
12178 \string<\glxtrLatinT
12179 \string& th =\glshex 00DE
12180 \string& TH =\glshex 00FE
12181 \string<u,U%
12182 \string<v,V%
12183 \string<w,W%
12184 \string<\glxtrLatinX
12185 \string<y,Y%
12186 \string<z,Z%
12187 }

```

GeneralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12188 \newcommand*{\glxtrGeneralLatinVrules}{%

```

```

12189 \glxtrLatinA
12190 \string<b,B%
12191 \string<c,C%
12192 \string<d,D%
12193 \string<\glxtrLatinEth
12194 \string<\glxtrLatinE
12195 \string<f,F%
12196 \string<g,G%
12197 \string<\glxtrLatinH
12198 \string<\glxtrLatinI
12199 \string<j,J%
12200 \string<\glxtrLatinK
12201 \string<\glxtrLatinL
12202 \string<\glxtrLatinM
12203 \string<\glxtrLatinN
12204 \string<\glxtrLatinO
12205 \string<\glxtrLatinP
12206 \string<q,Q%
12207 \string<r,R%
12208 \string<\glxtrLatinS
12209 \string& SS , \glxtrLatinEszettSs
12210 \string<\glxtrLatinT
12211 \string& th =\glshex 00DE
12212 \string& TH =\glshex 00FE
12213 \string<u,U%
12214 \string<v,V%
12215 \string<w,W%
12216 \string<\glxtrLatinX
12217 \string<y,Y%
12218 \string<z,Z%
12219 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12220 \newcommand*{\glxtrGeneralLatinVIrules}{%
12221 \glxtrLatinA
12222 \string<b,B%
12223 \string<c,C%
12224 \string<d,D%
12225 \string<\glxtrLatinEth
12226 \string<\glxtrLatinE
12227 \string<f,F%
12228 \string<g,G%
12229 \string<\glxtrLatinH
12230 \string<\glxtrLatinI
12231 \string<j,J%
12232 \string<\glxtrLatinK
12233 \string<\glxtrLatinL
12234 \string<\glxtrLatinM
12235 \string<\glxtrLatinN

```

```

12236 \string<\glxtrLatinO
12237 \string<\glxtrLatinP
12238 \string<q,Q%
12239 \string<r,R%
12240 \string<\glxtrLatinS
12241 \string& SZ , \glxtrLatinEszettSz
12242 \string<\glxtrLatinT
12243 \string& th =\glshex 00DE
12244 \string& TH =\glshex 00FE
12245 \string<u,U%
12246 \string<v,V%
12247 \string<w,W%
12248 \string<\glxtrLatinX
12249 \string<y,Y%
12250 \string<z,Z%
12251 }

```

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).

```

12252 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12253 \glxtrLatinA
12254 \string<\glxtrLatinAELigature
12255 \string<b,B%
12256 \string<c,C%
12257 \string<d,D%
12258 \string<\glxtrLatinEth
12259 \string<\glxtrLatinE
12260 \string<f,F%
12261 \string<\glxtrLatinInsularG
12262 \string<\glxtrLatinH
12263 \string<\glxtrLatinI
12264 \string<j,J%
12265 \string<\glxtrLatinK
12266 \string<\glxtrLatinL
12267 \string<\glxtrLatinM
12268 \string<\glxtrLatinN
12269 \string<\glxtrLatinO
12270 \string<\glxtrLatinOELigature
12271 \string<\glxtrLatinP
12272 \string<q,Q%
12273 \string<r,R%
12274 \string<\glshex 017F=\glxtrLatinS % s and long s
12275 \string<\glxtrLatinT
12276 \string<\glxtrLatinThorn
12277 \string<u,U%
12278 \string<v,V%
12279 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12280 \string<\glxtrLatinX
12281 \string<y,Y%

```

```

12282 \string<z,Z%
12283 }

```

1LatinVIIIrules General Latin alphabet (\mathfrak{A} treated as AE and \mathfrak{C} treated as OE, \mathfrak{P} treated as TH, \mathfrak{B} treated as SS, \mathfrak{eth} treated as D, \mathfrak{O} treated as O, \mathfrak{L} treated as L).

```

12284 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
12285 \glxtrLatinA
12286 \string& AE , \glxtrLatinAELigature
12287 \string<b,B%
12288 \string<c,C%
12289 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12290 \string<\glxtrLatinE
12291 \string<f,F%
12292 \string<g,G%
12293 \string<\glxtrLatinH
12294 \string<\glxtrLatinI
12295 \string<j,J%
12296 \string<\glxtrLatinK
12297 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12298 \string<\glxtrLatinM
12299 \string<\glxtrLatinN
12300 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12301 \string& OE , \glxtrLatinOELigature
12302 \string<\glxtrLatinP
12303 \string<q,Q%
12304 \string<r,R%
12305 \string<\glxtrLatinS
12306 \string& SS , \glxtrLatinEszettSs
12307 \string<\glxtrLatinT
12308 \string& th =\glshex 00DE
12309 \string& TH =\glshex 00FE
12310 \string<u,U%
12311 \string<v,V%
12312 \string<w,W%
12313 \string<\glxtrLatinX
12314 \string<y,Y%
12315 \string<z,Z%
12316 }

```

\glxtrLatinA

```

12317 \newcommand*{\glxtrLatinA}{%
12318 a\string=\glshex 00AA\string=\glshex 2090,A
12319 }

```

\glxtrLatinE

```

12320 \newcommand*{\glxtrLatinE}{%
12321 e\string=\glshex 2091,E
12322 }

```

\glsxtrLatinH

```
12323 \newcommand*{\glsxtrLatinH}{%  
12324   h\string=\glshex 2095,H  
12325 }
```

\glsxtrLatinI

```
12326 \newcommand*{\glsxtrLatinI}{%  
12327   i\string=\glshex 2071,I  
12328 }
```

\glsxtrLatinK

```
12329 \newcommand*{\glsxtrLatinK}{%  
12330   k\string=\glshex 2096,K  
12331 }
```

\glsxtrLatinL

```
12332 \newcommand*{\glsxtrLatinL}{%  
12333   l\string=\glshex 2097,L  
12334 }
```

\glsxtrLatinM

```
12335 \newcommand*{\glsxtrLatinM}{%  
12336   m\string=\glshex 2098,M  
12337 }
```

\glsxtrLatinN

```
12338 \newcommand*{\glsxtrLatinN}{%  
12339   n\string=\glshex 207F\string=\glshex 2099,N  
12340 }
```

\glsxtrLatinO

```
12341 \newcommand*{\glsxtrLatinO}{%  
12342   o\string=\glshex 00BA\string=\glshex 2092,O  
12343 }
```

\glsxtrLatinP

```
12344 \newcommand*{\glsxtrLatinP}{%  
12345   p\string=\glshex 209A,P  
12346 }
```

\glsxtrLatinS

```
12347 \newcommand*{\glsxtrLatinS}{%  
12348   s\string=\glshex 209B,S  
12349 }
```

\glsxtrLatinT

```
12350 \newcommand*{\glsxtrLatinT}{%  
12351   t\string=\glshex 209C,T  
12352 }
```

```

\glxtrLatinX
12353 \newcommand*{\glxtrLatinX}{%
12354   x\string=\glshex 2093,X
12355 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
12356 \newcommand*{\glxtrLatinSchwa}{%
12357   \glshex 0259\string=\glshex 2094,\glshex 018F
12358 }

trLatinEszettSs
12359 \newcommand*{\glxtrLatinEszettSs}{%
12360   \glshex 00DF% eszett
12361   \string=\glshex 017Fs % long S s
12362 }

trLatinEszettSz
12363 \newcommand*{\glxtrLatinEszettSz}{%
12364   \glshex 00DF% eszett
12365   \string= \glshex 017Fz % long S z
12366 }

\glxtrLatinEth
12367 \newcommand*{\glxtrLatinEth}{%
12368   \glshex 00F0,\glshex 00D0% eth
12369 }

lsxtrLatinThorn
12370 \newcommand*{\glxtrLatinThorn}{%
12371   \glshex 00FE,\glshex 00DE% thorn
12372 }

LatinAELigature
12373 \newcommand*{\glxtrLatinAELigature}{%
12374   \glshex 00E6,\glshex 00C6% AE-ligature
12375 }

LatinOELigature
12376 \newcommand*{\glxtrLatinOELigature}{%
12377   \glshex 0153,\glshex 0152% OE-ligature
12378 }

\glxtrLatinAA
12379 \newcommand*{\glxtrLatinAA}{%
12380   \glshex 00E5=a\glshex 030A,% \aa
12381   \glshex 00C5=A\glshex 030A% \AA
12382 }

```

glsxtrLatinWynn

```
12383 \newcommand*{\glsxtrLatinWynn}{%  
12384   \glsheX 01BF,\glsheX 01F7% wynn  
12385 }
```

trLatinInsularG

```
12386 \newcommand*{\glsxtrLatinInsularG}{%  
12387   \glsheX 1D79,\glsheX A77D% insular G  
12388   \string; g, G  
12389 }
```

sxtrLatinOslash

```
12390 \newcommand*{\glsxtrLatinOslash}{%  
12391   \glsheX 00F8,\glsheX 00D8% \o, \O  
12392 }
```

sxtrLatinLslash

```
12393 \newcommand*{\glsxtrLatinLslash}{%  
12394   \glsheX 0142,\glsheX 0141% \l, \L  
12395 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12396 \newcommand*{\glsxtrMathUpGreekIrules}{%  
12397   \glsxtrUpAlpha  
12398   \string<\glsxtrUpBeta  
12399   \string<\glsxtrUpGamma  
12400   \string<\glsxtrUpDelta  
12401   \string<\glsxtrUpEpsilon  
12402   \string<\glsxtrUpDigamma  
12403   \string<\glsxtrUpZeta  
12404   \string<\glsxtrUpEta  
12405   \string<\glsxtrUpTheta  
12406   \string<\glsxtrUpIota  
12407   \string<\glsxtrUpKappa  
12408   \string<\glsxtrUpLambda  
12409   \string<\glsxtrUpMu  
12410   \string<\glsxtrUpNu  
12411   \string<\glsxtrUpXi  
12412   \string<\glsxtrUpOmicron  
12413   \string<\glsxtrUpPi  
12414   \string<\glsxtrUpRho  
12415   \string<\glsxtrUpSigma  
12416   \string<\glsxtrUpTau  
12417   \string<\glsxtrUpUpsilon  
12418   \string<\glsxtrUpPhi  
12419   \string<\glsxtrUpChi  
12420   \string<\glsxtrUpPsi  
12421   \string<\glsxtrUpOmega  
12422 }
```

hUpGreekIIrules Doesn't include digamma.

```
12423 \newcommand*{\glxtrMathUpGreekIIrules}{%
12424   \glxtrUpAlpha
12425   \string<\glxtrUpBeta
12426   \string<\glxtrUpGamma
12427   \string<\glxtrUpDelta
12428   \string<\glxtrUpEpsilon
12429   \string<\glxtrUpZeta
12430   \string<\glxtrUpEta
12431   \string<\glxtrUpTheta
12432   \string<\glxtrUpIota
12433   \string<\glxtrUpKappa
12434   \string<\glxtrUpLambda
12435   \string<\glxtrUpMu
12436   \string<\glxtrUpNu
12437   \string<\glxtrUpXi
12438   \string<\glxtrUpOmicron
12439   \string<\glxtrUpPi
12440   \string<\glxtrUpRho
12441   \string<\glxtrUpSigma
12442   \string<\glxtrUpTau
12443   \string<\glxtrUpUpsilon
12444   \string<\glxtrUpPhi
12445   \string<\glxtrUpChi
12446   \string<\glxtrUpPsi
12447   \string<\glxtrUpOmega
12448 }
```

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.

```
12449 \newcommand*{\glxtrMathItalicGreekIrules}{%
12450   \glxtrMathItalicAlpha
12451   \string<\glxtrMathItalicBeta
12452   \string<\glxtrMathItalicGamma
12453   \string<\glxtrMathItalicDelta
12454   \string<\glxtrMathItalicEpsilon
12455   \string<\glxtrUpDigamma
12456   \string<\glxtrMathItalicZeta
12457   \string<\glxtrMathItalicEta
12458   \string<\glxtrMathItalicTheta
12459   \string<\glxtrMathItalicIota
12460   \string<\glxtrMathItalicKappa
12461   \string<\glxtrMathItalicLambda
12462   \string<\glxtrMathItalicMu
12463   \string<\glxtrMathItalicNu
12464   \string<\glxtrMathItalicXi
12465   \string<\glxtrMathItalicOmicron
12466   \string<\glxtrMathItalicPi
12467   \string<\glxtrMathItalicRho
```

```

12468 \string<\glxtrMathItalicSigma
12469 \string<\glxtrMathItalicTau
12470 \string<\glxtrMathItalicUpsilon
12471 \string<\glxtrMathItalicPhi
12472 \string<\glxtrMathItalicChi
12473 \string<\glxtrMathItalicPsi
12474 \string<\glxtrMathItalicOmega
12475 }

```

licGreekIIrules Doesn't include digamma.

```

12476 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12477 \glxtrMathItalicAlpha
12478 \string<\glxtrMathItalicBeta
12479 \string<\glxtrMathItalicGamma
12480 \string<\glxtrMathItalicDelta
12481 \string<\glxtrMathItalicEpsilon
12482 \string<\glxtrMathItalicZeta
12483 \string<\glxtrMathItalicEta
12484 \string<\glxtrMathItalicTheta
12485 \string<\glxtrMathItalicIota
12486 \string<\glxtrMathItalicKappa
12487 \string<\glxtrMathItalicLambda
12488 \string<\glxtrMathItalicMu
12489 \string<\glxtrMathItalicNu
12490 \string<\glxtrMathItalicXi
12491 \string<\glxtrMathItalicOmicron
12492 \string<\glxtrMathItalicPi
12493 \string<\glxtrMathItalicRho
12494 \string<\glxtrMathItalicSigma
12495 \string<\glxtrMathItalicTau
12496 \string<\glxtrMathItalicUpsilon
12497 \string<\glxtrMathItalicPhi
12498 \string<\glxtrMathItalicChi
12499 \string<\glxtrMathItalicPsi
12500 \string<\glxtrMathItalicOmega
12501 }

```

pperGreekIrules Upper case only (includes upright digamma).

```

12502 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12503 \glshex 1D6E2% upper case alpha (maths italic)
12504 \string<\glshex 1D6E3% upper case beta (maths italic)
12505 \string<\glshex 1D6E4% upper case gamma (maths italic)
12506 \string<\glshex 1D6E5% upper case delta (maths italic)
12507 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12508 \string<\glshex 03DC% upper case digamma
12509 \string<\glshex 1D6E7% upper case zeta (maths italic)
12510 \string<\glshex 1D6E8% upper case eta (maths italic)
12511 \string<\glshex 1D6E9% upper case theta (maths italic)
12512 \string=<\glshex 1D6F3% upper case theta variant (maths italic)

```

```

12513 \string<\glshex 1D6EA% upper case iota (maths italic)
12514 \string<\glshex 1D6EB% upper case kappa (maths italic)
12515 \string<\glshex 1D6EC% upper case lambda (maths italic)
12516 \string<\glshex 1D6ED% upper case mu (maths italic)
12517 \string<\glshex 1D6EE% upper case nu (maths italic)
12518 \string<\glshex 1D6EF% upper case xi (maths italic)
12519 \string<\glshex 1D6F0% upper case omicron (maths italic)
12520 \string<\glshex 1D6F1% upper case pi (maths italic)
12521 \string<\glshex 1D6F2% upper case rho (maths italic)
12522 \string<\glshex 1D6F4% upper case sigma (maths italic)
12523 \string<\glshex 1D6F5% upper case tau (maths italic)
12524 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12525 \string<\glshex 1D6F7% upper case phi (maths italic)
12526 \string<\glshex 1D6F8% upper case chi (maths italic)
12527 \string<\glshex 1D6F9% upper case psi (maths italic)
12528 \string<\glshex 1D6FA% upper case omega (maths italic)
12529 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

12530 \newcommand*{\glsxtrMathItalicUpperGreekIIrules}{%
12531 \glshex 1D6E2% upper case alpha (maths italic)
12532 \string<\glshex 1D6E3% upper case beta (maths italic)
12533 \string<\glshex 1D6E4% upper case gamma (maths italic)
12534 \string<\glshex 1D6E5% upper case delta (maths italic)
12535 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12536 \string<\glshex 1D6E7% upper case zeta (maths italic)
12537 \string<\glshex 1D6E8% upper case eta (maths italic)
12538 \string<\glshex 1D6E9% upper case theta (maths italic)
12539 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12540 \string<\glshex 1D6EA% upper case iota (maths italic)
12541 \string<\glshex 1D6EB% upper case kappa (maths italic)
12542 \string<\glshex 1D6EC% upper case lambda (maths italic)
12543 \string<\glshex 1D6ED% upper case mu (maths italic)
12544 \string<\glshex 1D6EE% upper case nu (maths italic)
12545 \string<\glshex 1D6EF% upper case xi (maths italic)
12546 \string<\glshex 1D6F0% upper case omicron (maths italic)
12547 \string<\glshex 1D6F1% upper case pi (maths italic)
12548 \string<\glshex 1D6F2% upper case rho (maths italic)
12549 \string<\glshex 1D6F4% upper case sigma (maths italic)
12550 \string<\glshex 1D6F5% upper case tau (maths italic)
12551 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12552 \string<\glshex 1D6F7% upper case phi (maths italic)
12553 \string<\glshex 1D6F8% upper case chi (maths italic)
12554 \string<\glshex 1D6F9% upper case psi (maths italic)
12555 \string<\glshex 1D6FA% upper case omega (maths italic)
12556 }

```

lowerGreekIrules Lower case only (includes upright digamma).

```

12557 \newcommand*{\glsxtrMathItalicLowerGreekIrules}{%

```

```

12558 \glshex 1D6FC% lower case alpha (maths italic)
12559 \string<\glshex 1D6FD% lower case beta (maths italic)
12560 \string<\glshex 1D6FE% lower case gamma (maths italic)
12561 \string<\glshex 1D6FF% lower case delta (maths italic)
12562 \string<\glshex 1D700% lower case epsilon (maths italic)
12563 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12564 \string<\glshex 03DD% lower case digamma
12565 \string<\glshex 1D701% lower case zeta (maths italic)
12566 \string<\glshex 1D702% lower case eta (maths italic)
12567 \string<\glshex 1D703% lower case theta (maths italic)
12568 \string=\glshex 1D717% lower case theta variant (maths italic)
12569 \string<\glshex 1D704% lower case iota (maths italic)
12570 \string<\glshex 1D705% lower case kappa (maths italic)
12571 \string=\glshex 1D718% lower case kappa variant (maths italic)
12572 \string<\glshex 1D706% lower case lambda (maths italic)
12573 \string<\glshex 1D707% lower case mu (maths italic)
12574 \string<\glshex 1D708% lower case nu (maths italic)
12575 \string<\glshex 1D709% lower case xi (maths italic)
12576 \string<\glshex 1D70A% lower case omicron (maths italic)
12577 \string<\glshex 1D70B% lower case pi (maths italic)
12578 \string=\glshex 1D71B% lower case pi variant (maths italic)
12579 \string<\glshex 1D70C% lower case rho (maths italic)
12580 \string=\glshex 1D71A% lower case rho variant (maths italic)
12581 \string<\glshex 1D70D% lower case final sigma (maths italic)
12582 \string=\glshex 1D70E% lower case sigma (maths italic)
12583 \string<\glshex 1D70F% lower case tau (maths italic)
12584 \string<\glshex 1D710% lower case upsilon (maths italic)
12585 \string<\glshex 1D711% lower case phi (maths italic)
12586 \string=\glshex 1D719% lower case phi variant (maths italic)
12587 \string<\glshex 1D712% lower case chi (maths italic)
12588 \string<\glshex 1D713% lower case psi (maths italic)
12589 \string<\glshex 1D714% lower case omega (maths italic)
12590 }

```

LowerGreeklIrules Lower case only (doesn't includes upright digamma).

```

12591 \newcommand*{\glxtrMathItalicLowerGreeklIrules}{%
12592 \glshex 1D6FC% lower case alpha (maths italic)
12593 \string<\glshex 1D6FD% lower case beta (maths italic)
12594 \string<\glshex 1D6FE% lower case gamma (maths italic)
12595 \string<\glshex 1D6FF% lower case delta (maths italic)
12596 \string<\glshex 1D700% lower case epsilon (maths italic)
12597 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12598 \string<\glshex 1D701% lower case zeta (maths italic)
12599 \string<\glshex 1D702% lower case eta (maths italic)
12600 \string<\glshex 1D703% lower case theta (maths italic)
12601 \string=\glshex 1D717% lower case theta variant (maths italic)
12602 \string<\glshex 1D704% lower case iota (maths italic)
12603 \string<\glshex 1D705% lower case kappa (maths italic)
12604 \string=\glshex 1D718% lower case kappa variant (maths italic)

```

```

12605 \string<\glshex 1D706% lower case lambda (maths italic)
12606 \string<\glshex 1D707% lower case mu (maths italic)
12607 \string<\glshex 1D708% lower case nu (maths italic)
12608 \string<\glshex 1D709% lower case xi (maths italic)
12609 \string<\glshex 1D70A% lower case omicron (maths italic)
12610 \string<\glshex 1D70B% lower case pi (maths italic)
12611 \string=\glshex 1D71B% lower case pi variant (maths italic)
12612 \string<\glshex 1D70C% lower case rho (maths italic)
12613 \string=\glshex 1D71A% lower case rho variant (maths italic)
12614 \string<\glshex 1D70D% lower case final sigma (maths italic)
12615 \string=\glshex 1D70E% lower case sigma (maths italic)
12616 \string<\glshex 1D70F% lower case tau (maths italic)
12617 \string<\glshex 1D710% lower case upsilon (maths italic)
12618 \string<\glshex 1D711% lower case phi (maths italic)
12619 \string=\glshex 1D719% lower case phi variant (maths italic)
12620 \string<\glshex 1D712% lower case chi (maths italic)
12621 \string<\glshex 1D713% lower case psi (maths italic)
12622 \string<\glshex 1D714% lower case omega (maths italic)
12623 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12624 \newcommand*{\glxtrMathGreekIrules}{%
12625 \glxtrMathItalicAlpha
12626 \string;\glxtrUpAlpha
12627 \string<\glxtrMathItalicBeta
12628 \string;\glxtrUpBeta
12629 \string<\glxtrMathItalicGamma
12630 \string;\glxtrUpGamma
12631 \string<\glxtrMathItalicDelta
12632 \string;\glxtrUpDelta
12633 \string<\glxtrMathItalicEpsilon
12634 \string;\glxtrUpEpsilon
12635 \string<\glxtrUpDigamma
12636 \string<\glxtrMathItalicZeta
12637 \string;\glxtrUpZeta
12638 \string<\glxtrMathItalicEta
12639 \string;\glxtrUpEta
12640 \string<\glxtrMathItalicTheta
12641 \string;\glxtrUpTheta
12642 \string<\glxtrMathItalicIota
12643 \string;\glxtrUpIota
12644 \string<\glxtrMathItalicKappa
12645 \string;\glxtrUpKappa
12646 \string<\glxtrMathItalicLambda
12647 \string;\glxtrUpLambda
12648 \string<\glxtrMathItalicMu
12649 \string;\glxtrUpMu
12650 \string<\glxtrMathItalicNu
12651 \string;\glxtrUpNu

```

```

12652 \string<\glxtrMathItalicXi
12653 \string;\glxtrUpXi
12654 \string<\glxtrMathItalicOmicron
12655 \string;\glxtrUpOmicron
12656 \string<\glxtrMathItalicPi
12657 \string;\glxtrUpPi
12658 \string<\glxtrMathItalicRho
12659 \string;\glxtrUpRho
12660 \string<\glxtrMathItalicSigma
12661 \string;\glxtrUpSigma
12662 \string<\glxtrMathItalicTau
12663 \string;\glxtrUpTau
12664 \string<\glxtrMathItalicUpsilon
12665 \string;\glxtrUpUpsilon
12666 \string<\glxtrMathItalicPhi
12667 \string;\glxtrUpPhi
12668 \string<\glxtrMathItalicChi
12669 \string;\glxtrUpChi
12670 \string<\glxtrMathItalicPsi
12671 \string;\glxtrUpPsi
12672 \string<\glxtrMathItalicOmega
12673 \string;\glxtrUpOmega
12674 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

12675 \newcommand*{\glxtrMathGreekIIrules}{%
12676 \glxtrMathItalicAlpha
12677 \string;\glxtrUpAlpha
12678 \string<\glxtrMathItalicBeta
12679 \string;\glxtrUpBeta
12680 \string<\glxtrMathItalicGamma
12681 \string;\glxtrUpGamma
12682 \string<\glxtrMathItalicDelta
12683 \string;\glxtrUpDelta
12684 \string<\glxtrMathItalicEpsilon
12685 \string;\glxtrUpEpsilon
12686 \string<\glxtrMathItalicZeta
12687 \string;\glxtrUpZeta
12688 \string<\glxtrMathItalicEta
12689 \string;\glxtrUpEta
12690 \string<\glxtrMathItalicTheta
12691 \string;\glxtrUpTheta
12692 \string<\glxtrMathItalicIota
12693 \string;\glxtrUpIota
12694 \string<\glxtrMathItalicKappa
12695 \string;\glxtrUpKappa
12696 \string<\glxtrMathItalicLambda
12697 \string;\glxtrUpLambda
12698 \string<\glxtrMathItalicMu

```

```

12699 \string;\glxtrUpMu
12700 \string<\glxtrMathItalicNu
12701 \string;\glxtrUpNu
12702 \string<\glxtrMathItalicXi
12703 \string;\glxtrUpXi
12704 \string<\glxtrMathItalicOmicron
12705 \string;\glxtrUpOmicron
12706 \string<\glxtrMathItalicPi
12707 \string;\glxtrUpPi
12708 \string<\glxtrMathItalicRho
12709 \string;\glxtrUpRho
12710 \string<\glxtrMathItalicSigma
12711 \string;\glxtrUpSigma
12712 \string<\glxtrMathItalicTau
12713 \string;\glxtrUpTau
12714 \string<\glxtrMathItalicUpsilon
12715 \string;\glxtrUpUpsilon
12716 \string<\glxtrMathItalicPhi
12717 \string;\glxtrUpPhi
12718 \string<\glxtrMathItalicChi
12719 \string;\glxtrUpChi
12720 \string<\glxtrMathItalicPsi
12721 \string;\glxtrUpPsi
12722 \string<\glxtrMathItalicOmega
12723 \string;\glxtrUpOmega
12724 }

```

\glxtrUpAlpha

```

12725 \newcommand*{\glxtrUpAlpha}{%
12726 \glshex 03B1,% lower case alpha
12727 \glshex 0391% upper case alpha
12728 }

```

\glxtrUpBeta

```

12729 \newcommand*{\glxtrUpBeta}{%
12730 \glshex 03B2,% lower case beta
12731 \glshex 0392% upper case beta
12732 }

```

\glxtrUpGamma

```

12733 \newcommand*{\glxtrUpGamma}{%
12734 \glshex 03B3,% lower case gamma
12735 \glshex 0393% upper case gamma
12736 }

```

\glxtrUpDelta

```

12737 \newcommand*{\glxtrUpDelta}{%
12738 \glshex 03B4,% lower case delta
12739 \glshex 0394% upper case delta

```

12740 }

glsxtrUpEpsilon

```
12741 \newcommand*{\glsxtrUpEpsilon}{%
12742   \glsheX 03B5% lower case epsilon
12743   \string=\glsheX 03F5,% lower case epsilon variant
12744   \glsheX 0395% upper case epsilon
12745 }
```

glsxtrUpDigamma

```
12746 \newcommand*{\glsxtrUpDigamma}{%
12747   \glsheX 03DD,% lower case digamma
12748   \glsheX 03DC% upper case digamma
12749 }
```

\glsxtrUpZeta

```
12750 \newcommand*{\glsxtrUpZeta}{%
12751   \glsheX 03B6,% lower case zeta
12752   \glsheX 0396% upper case zeta
12753 }
```

\glsxtrUpEta

```
12754 \newcommand*{\glsxtrUpEta}{%
12755   \glsheX 03B7,% lower case eta
12756   \glsheX 0397% upper case eta
12757 }
```

\glsxtrUpTheta

```
12758 \newcommand*{\glsxtrUpTheta}{%
12759   \glsheX 03B8% lower case theta
12760   \string=\glsheX 03D1,% lower case theta variant
12761   \glsheX 0398% upper case theta
12762 }
```

\glsxtrUpIota

```
12763 \newcommand*{\glsxtrUpIota}{%
12764   \glsheX 03B9,% lower case iota
12765   \glsheX 0399% upper case iota
12766 }
```

\glsxtrUpKappa

```
12767 \newcommand*{\glsxtrUpKappa}{%
12768   \glsheX 03BA% lower case kappa
12769   \string=\glsheX 03F0,% lower case kappa variant
12770   \glsheX 039A% upper case kappa
12771 }
```

\glxtrUpLambda

```
12772 \newcommand*{\glxtrUpLambda}{%
12773   \glshex 03BB,% lower lambda
12774   \glshex 039B% upper case lambda
12775 }
```

\glxtrUpMu

```
12776 \newcommand*{\glxtrUpMu}{%
12777   \glshex 03BC,% lower case mu
12778   \glshex 039C% upper case mu
12779 }
```

\glxtrUpNu

```
12780 \newcommand*{\glxtrUpNu}{%
12781   \glshex 03BD,% lower case nu
12782   \glshex 039D% upper case nu
12783 }
```

\glxtrUpXi

```
12784 \newcommand*{\glxtrUpXi}{%
12785   \glshex 03BE,% lower case xi
12786   \glshex 039E% upper case xi
12787 }
```

\glxtrUpOmicron

```
12788 \newcommand*{\glxtrUpOmicron}{%
12789   \glshex 03BF,% lower case omicron
12790   \glshex 039F% upper case omicron
12791 }
```

\glxtrUpPi

```
12792 \newcommand*{\glxtrUpPi}{%
12793   \glshex 03C0% lower case pi
12794   \string=\glshex 03D6,% lower case pi variant
12795   \glshex 03A0% upper case pi
12796 }
```

\glxtrUpRho

```
12797 \newcommand*{\glxtrUpRho}{%
12798   \glshex 03C1% lower case rho
12799   \string=\glshex 03F1,% lower case rho variant
12800   \glshex 03A1% upper case rho
12801 }
```

\glxtrUpSigma

```
12802 \newcommand*{\glxtrUpSigma}{%
12803   \glshex 03C2% lower case sigma
12804   \string=\glshex 03C3,% lower case sigma
```

```
12805 \glshex 03A3% upper case sigma
12806 }
```

`\glxtrUpTau`

```
12807 \newcommand*{\glxtrUpTau}{%
12808 \glshex 03C4,% lower case tau
12809 \glshex 03A4% upper case tau
12810 }
```

`glxtrUpUpsilon`

```
12811 \newcommand*{\glxtrUpUpsilon}{%
12812 \glshex 03C5,% lower case upsilon
12813 \glshex 03A5% upper case upsilon
12814 }
```

`\glxtrUpPhi`

```
12815 \newcommand*{\glxtrUpPhi}{%
12816 \glshex 03C6% lower case phi
12817 \string=\glshex 03D5,% lower case phi variant
12818 \glshex 03A6% upper case phi
12819 }
```

`\glxtrUpChi`

```
12820 \newcommand*{\glxtrUpChi}{%
12821 \glshex 03C7,% lower case chi
12822 \glshex 03A7% upper case chi
12823 }
```

`\glxtrUpPsi`

```
12824 \newcommand*{\glxtrUpPsi}{%
12825 \glshex 03C8,% lower case psi
12826 \glshex 03A8% upper case psi
12827 }
```

`\glxtrUpOmega`

```
12828 \newcommand*{\glxtrUpOmega}{%
12829 \glshex 03C9,% lower case omega
12830 \glshex 03A9% upper case omega
12831 }
```

`MathItalicAlpha`

```
12832 \newcommand*{\glxtrMathItalicAlpha}{%
12833 \glshex 1D6FC,% lower case alpha (maths italic)
12834 \glshex 1D6E2% upper case alpha (maths italic)
12835 }
```

`rMathItalicBeta`

```
12836 \newcommand*{\glxtrMathItalicBeta}{%
```

```

12837 \glshex 1D6FD,% lower case beta (maths italic)
12838 \glshex 1D6E3% upper case beta (maths italic)
12839 }

```

MathItalicGamma

```

12840 \newcommand*{\glxtrMathItalicGamma}{%
12841 \glshex 1D6FE,% lower case gamma (maths italic)
12842 \glshex 1D6E4% upper case gamma (maths italic)
12843 }

```

MathItalicDelta

```

12844 \newcommand*{\glxtrMathItalicDelta}{%
12845 \glshex 1D6FF,% lower case delta (maths italic)
12846 \glshex 1D6E5% upper case delta (maths italic)
12847 }

```

MathItalicEpsilon

```

12848 \newcommand*{\glxtrMathItalicEpsilon}{%
12849 \glshex 1D700% lower case epsilon (maths italic)
12850 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12851 \glshex 1D6E6% upper case epsilon (maths italic)
12852 }

```

MathItalicZeta

```

12853 \newcommand*{\glxtrMathItalicZeta}{%
12854 \glshex 1D701,% lower case zeta (maths italic)
12855 \glshex 1D6E7% upper case zeta (maths italic)
12856 }

```

MathItalicEta

```

12857 \newcommand*{\glxtrMathItalicEta}{%
12858 \glshex 1D702,% lower case eta (maths italic)
12859 \glshex 1D6E8% upper case eta (maths italic)
12860 }

```

MathItalicTheta

```

12861 \newcommand*{\glxtrMathItalicTheta}{%
12862 \glshex 1D703% lower case theta (maths italic)
12863 \string=\glshex 1D717,% lower case theta variant (maths italic)
12864 \glshex 1D6E9% upper case theta (maths italic)
12865 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12866 }

```

MathItalicIota

```

12867 \newcommand*{\glxtrMathItalicIota}{%
12868 \glshex 1D704,% lower case iota (maths italic)
12869 \glshex 1D6EA% upper case iota (maths italic)
12870 }

```

MathItalicKappa

```
12871 \newcommand*{\glxtrMathItalicKappa}{%
12872   \glshex 1D705% lower case kappa (maths italic)
12873   \string=\glshex 1D718,% lower case kappa variant (maths italic)
12874   \glshex 1D6EB% upper case kappa (maths italic)
12875 }
```

athItalicLambda

```
12876 \newcommand*{\glxtrMathItalicLambda}{%
12877   \glshex 1D706,% lower case lambda (maths italic)
12878   \glshex 1D6EC% upper case lambda (maths italic)
12879 }
```

xtrMathItalicMu

```
12880 \newcommand*{\glxtrMathItalicMu}{%
12881   \glshex 1D707,% lower case mu (maths italic)
12882   \glshex 1D6ED% upper case mu (maths italic)
12883 }
```

xtrMathItalicNu

```
12884 \newcommand*{\glxtrMathItalicNu}{%
12885   \glshex 1D708,% lower case nu (maths italic)
12886   \glshex 1D6EE% upper case nu (maths italic)
12887 }
```

xtrMathItalicXi

```
12888 \newcommand*{\glxtrMathItalicXi}{%
12889   \glshex 1D709,% lower case xi (maths italic)
12890   \glshex 1D6EF% upper case xi (maths italic)
12891 }
```

thItalicOmicron

```
12892 \newcommand*{\glxtrMathItalicOmicron}{%
12893   \glshex 1D70A,% lower case omicron (maths italic)
12894   \glshex 1D6F0% upper case omicron (maths italic)
12895 }
```

xtrMathItalicPi

```
12896 \newcommand*{\glxtrMathItalicPi}{%
12897   \glshex 1D70B% lower case pi (maths italic)
12898   \string=\glshex 1D71B,% lower case pi variant (maths italic)
12899   \glshex 1D6F1% upper case pi (maths italic)
12900 }
```

trMathItalicRho

```
12901 \newcommand*{\glxtrMathItalicRho}{%
12902   \glshex 1D70C% lower case rho (maths italic)
12903   \string=\glshex 1D71A,% lower case rho variant (maths italic)
```

```

12904 \glshex 1D6F2% upper case rho (maths italic)
12905 }

```

MathItalicSigma

```

12906 \newcommand*{\glsxtrMathItalicSigma}{%
12907 \glshex 1D70D% lower case final sigma (maths italic)
12908 \string=\glshex 1D70E,% lower case sigma (maths italic)
12909 \glshex 1D6F4% upper case sigma (maths italic)
12910 }

```

trMathItalicTau

```

12911 \newcommand*{\glsxtrMathItalicTau}{%
12912 \glshex 1D70F,% lower case tau (maths italic)
12913 \glshex 1D6F5% upper case tau (maths italic)
12914 }

```

thItalicUpsilon

```

12915 \newcommand*{\glsxtrMathItalicUpsilon}{%
12916 \glshex 1D710,% lower case upsilon (maths italic)
12917 \glshex 1D6F6% upper case upsilon (maths italic)
12918 }

```

trMathItalicPhi

```

12919 \newcommand*{\glsxtrMathItalicPhi}{%
12920 \glshex 1D711% lower case phi (maths italic)
12921 \string=\glshex 1D719,% lower case phi variant (maths italic)
12922 \glshex 1D6F7% upper case phi (maths italic)
12923 }

```

trMathItalicChi

```

12924 \newcommand*{\glsxtrMathItalicChi}{%
12925 \glshex 1D712,% lower case chi (maths italic)
12926 \glshex 1D6F8% upper case chi (maths italic)
12927 }

```

trMathItalicPsi

```

12928 \newcommand*{\glsxtrMathItalicPsi}{%
12929 \glshex 1D713,% lower case psi (maths italic)
12930 \glshex 1D6F9% upper case psi (maths italic)
12931 }

```

MathItalicOmega

```

12932 \newcommand*{\glsxtrMathItalicOmega}{%
12933 \glshex 1D714,% lower case omega (maths italic)
12934 \glshex 1D6FA% upper case omega (maths italic)
12935 }

```

thItalicPartial

```
12936 \newcommand*{\glxtrMathItalicPartial}{%
12937   \glshex 1D715% partial differential (maths italic)
12938 }
```

MathItalicNabla

```
12939 \newcommand*{\glxtrMathItalicNabla}{%
12940   \glshex 1D6FB% nabla (maths italic)
12941 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12942 \newcommand*{\glxtrdigitrules}{%
12943   0\string=\glshex 2080\string=\glshex 2070
12944   \string<1\string=\glshex 2081\string=\glshex 00B9
12945   \string<2\string=\glshex 2082\string=\glshex 00B2
12946   \string<3\string=\glshex 2083\string=\glshex 00B3
12947   \string<4\string=\glshex 2084\string=\glshex 2074
12948   \string<5\string=\glshex 2085\string=\glshex 2075
12949   \string<6\string=\glshex 2086\string=\glshex 2076
12950   \string<7\string=\glshex 2087\string=\glshex 2077
12951   \string<8\string=\glshex 2088\string=\glshex 2078
12952   \string<9\string=\glshex 2089\string=\glshex 2079
12953 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12954 \newcommand*{\glxtrBasicDigitrules}{%
12955   0\string<1\string<2\string<3\string<4%
12956   \string<5\string<6\string<7\string<8\string<9%
12957 }
```

criptDigitrules Subscript digits.

```
12958 \newcommand*{\glxtrSubScriptDigitrules}{%
12959   \glshex 2080% subscript 0
12960   \string<\glshex 2081% subscript 1
12961   \string<\glshex 2082% subscript 2
12962   \string<\glshex 2083% subscript 3
12963   \string<\glshex 2084% subscript 4
12964   \string<\glshex 2085% subscript 5
12965   \string<\glshex 2086% subscript 6
12966   \string<\glshex 2087% subscript 7
12967   \string<\glshex 2088% subscript 8
12968   \string<\glshex 2089% subscript 9
12969 }
```

criptDigitrules Superscript digits.

```
12970 \newcommand*{\glxtrSuperScriptDigitrules}{%
12971   \glshex 2070% superscript 0
12972   \string<\glshex 00B9% superscript 1
```

```

12973 \string<\glshex 00B2% superscript 2
12974 \string<\glshex 00B3% superscript 3
12975 \string<\glshex 2074% superscript 4
12976 \string<\glshex 2075% superscript 5
12977 \string<\glshex 2076% superscript 6
12978 \string<\glshex 2077% superscript 7
12979 \string<\glshex 2078% superscript 8
12980 \string<\glshex 2079% superscript 9
12981 }

```

trfractionrules Vulgar fractions.

```

12982 \newcommand*{\glxtrfractionrules}{%
12983   \glshex 215F% fraction numerator one (1/)
12984   \string<\glshex 2189% zero thirds (0/3 = 0)
12985   \string<\glshex 2152% one tenth (1/10 = 0.1)
12986   \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12987   \string<\glshex 215B% one eighth (1/8 = 0.125)
12988   \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12989   \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12990   \string<\glshex 2155% one fifth (1/5 = 0.2)
12991   \string<\glshex 00BC% one quarter (1/4 = 0.25)
12992   \string<\glshex 2153% one third (1/3 ~ 0.333)
12993   \string<\glshex 215C% three eighths (3/8 = 0.375)
12994   \string<\glshex 2156% two fifths (2/5 = 0.4)
12995   \string<\glshex 00BD% one half (1/2 = 0.5)
12996   \string<\glshex 2157% three fifths (3/5 = 0.6)
12997   \string<\glshex 215D% five eighths (5/8 = 0.625)
12998   \string<\glshex 2154% two thirds (2/3 ~ 0.667)
12999   \string<\glshex 00BE% three quarters (3/4 = 0.75)
13000   \string<\glshex 2158% four fifths (4/5 = 0.8)
13001   \string<\glshex 215A% five sixths (5/6 ~ 0.833)
13002   \string<\glshex 215E% seven eighths (7/8 = 0.875)
13003 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

13004 \renewcommand{\@glxtrdialecthook}{%
13005   \ifundef\CurrentTrackedScript
13006   {%
13007     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13008     {%
13009       \edef\CurrentTrackedScript{%
13010         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13011     }%
13012   }%
13013 }%
13014 {}%
13015 \ifdef\CurrentTrackedScript
13016 {%
13017   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix

```

```

13018 \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
13019 \let\CurrentTrackedTag\CurrentTrackedScript
13020 \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
13021 {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13022 {}%
13023 \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13024 }%
13025 {}%
13026 }

```

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.

```

13027 \ifdef\glsxtr@loaddialect
13028 {%
13029 \@ifpackageloaded{tracklang}
13030 {%
13031 \AnyTrackedLanguages
13032 {%
13033 \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13034 }%
13035 {}%
13036 }
13037 {}
13038 }
13039 {}

```

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:

```
13040 \NeedsTeXFormat{LaTeX2e}
13041 \ProvidesPackage{glossaries-extra-stylemods}[2018/08/13 v1.35 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-<option>.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glxtr@loadstyles`.

`sxtr@loadstyles`

```
13042 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13043 \DeclareOption{all}{%
13044   \appto\@glxtr@loadstyles{%
13045     \RequirePackage{glossary-inline}%
13046     \RequirePackage{glossary-list}%
13047     \RequirePackage{glossary-tree}%
13048     \RequirePackage{glossary-mcols}%
13049     \RequirePackage{glossary-long}%
13050     \RequirePackage{glossary-longragged}%
13051     \RequirePackage{glossary-longbooktabs}%
13052     \RequirePackage{glossary-super}%
13053     \RequirePackage{glossary-superragged}%
13054     \RequirePackage{glossary-bookindex}%
13055   }
13056 }

13057 \DeclareOption*{%
13058   \IfFileExists{glossary-\CurrentOption.sty}
13059   {\eappto\@glxtr@loadstyles{%
13060     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13061   }%
13062   {%
13063     \PackageError{glossaries-extra-styles}%

```

```

13064     {Unknown option ‘\CurrentOption’}{}%
13065   }%
13066 }

```

Process the package options:

```
13067 \ProcessOptions
```

Load the required packages:

```
13068 \@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`.

```
13069 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

13070 \providecommand{\renewglossarystyle}[2]{%
13071   \ifcsundef{@glsstyle@#1}%
13072   {%
13073     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13074   }%
13075   {%
13076     \csdef{@glsstyle@#1}{#2}%
13077   }%
13078 }

```

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.

```

13079 \ifdef{@glsstyle@listdotted}
13080 {%
13081   \renewglossarystyle{listdotted}{%
13082     \setglossarystyle{list}%
13083     \renewcommand*{\glossentry}[2]{%
13084       \item[]\makebox[\glslistdottedwidth][l]{%
13085         \glsentryitem{##1}%
13086         \glstarget{##1}{\glossentryname{##1}}%
13087         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13088         \glossentrydesc{##1}\glspostdescription}%
13089     \renewcommand*{\subglossentry}[3]{%
13090       \item[]\makebox[\glslistdottedwidth][l]{%
13091         \glssubentryitem{##2}%
13092         \glstarget{##2}{\glossentryname{##2}}%
13093         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13094         \glossentrydesc{##2}\glspostdescription}%
13095   }

```

```
13096 }
13097 {%
```

Assume the style isn't required if it hasn't already been defined.

```
13098 }
```

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.

```
13099 \ifdef{\@glsstyle@list}
13100 {%
```

listprelocation Space before number list for top-level entries.

```
13101 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13102 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13103 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13104 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
13105 \renewglossarystyle{list}{%
13106   \renewenvironment{theglossary}%
13107     {\begin{description}}{\end{description}}%
13108   \renewcommand*\{glossaryheader}{}%
13109   \renewcommand*\{glsgroupheading}[1]{}%
13110   \renewcommand*\{glossentry}[2]{%
13111     \item[\glssentryitem{##1}%
13112       \glstarget{##1}{\glossentryname{##1}}]
13113     \glslistdesc{##1}\glslistprelocation ##2}%
13114   \renewcommand*\{subglossentry}[3]{%
13115     \glssubentryitem{##2}%
13116     \glstarget{##2}{\strut}\space
13117     \glslistdesc{##2}%
13118     \glslistchildprelocation ##3\glslistchildpostlocation}%
13119   \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13120 }
13121 }
13122 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13123 \ifdef{\@glsstyle@altlist}
13124 {%
13125   \renewglossarystyle{altlist}{%
```

```

13126 \setglossarystyle{list}%
13127 \renewcommand*{\glossentry}[2]{%
13128   \item[\glentryitem{##1}%
13129     \glstarget{##1}{\glossentryname{##1}}]%
13130   \mbox{}\par\nobreak\@afterheading
13131   \glslistdesc{##1}\glslistprelocation ##2}%
13132 \renewcommand{\subglossentry}[3]{%
13133   \par
13134   \glssubentryitem{##2}%
13135   \glstarget{##2}{\strut}\glslistdesc{##2}%
13136   \glslistchildprelocation ##3}%
13137 }
13138 }
13139 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13140 \ifdef{\@glsstyle@listgroup}
13141 {%
13142   \renewglossarystyle{listgroup}{%
13143     \setglossarystyle{list}%
13144     \renewcommand*{\glsgroupheading}[1]{%
13145       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13146       \mbox{}\par\nobreak\@afterheading
13147     }%
13148   }
13149 }
13150 {}

```

Similarly for listhypergroup.

```

13151 \ifdef{\@glsstyle@listhypergroup}
13152 {%
13153   \renewglossarystyle{listhypergroup}{%
13154     \setglossarystyle{list}%
13155     \renewcommand*{\glossaryheader}{%
13156       \glslistnavigationitem{\glsnavigation}}%
13157     \renewcommand*{\glsgroupheading}[1]{%
13158       \item[\glslistgroupheaderfmt
13159         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13160       \mbox{}\par\nobreak\@afterheading
13161     }%
13162   }
13163 }
13164 {}

```

Similarly for altlistgroup.

```

13165 \ifdef{\@glsstyle@altlistgroup}
13166 {%
13167   \renewglossarystyle{altlistgroup}{%
13168     \setglossarystyle{altlist}%
13169     \renewcommand*{\glsgroupheading}[1]{%
13170       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```

```

13171      \mbox{}\par\nobreak\@afterheading
13172    }%
13173  }
13174 }
13175 {}

```

Similarly for altlisthypergroup.

```

13176 \ifdef{\@glsstyle@altlisthypergroup}
13177 {%
13178   \renewglossarystyle{altlisthypergroup}{%
13179     \setglossarystyle{altlist}%
13180     \renewcommand*\glossaryheader{%
13181       \glslistnavigationitem{\glsnavigation}}%
13182     \renewcommand*\glsgroupheading[1]{%
13183       \item[\glslistgroupheaderfmt
13184         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13185       \mbox{}\par\nobreak\@afterheading
13186     }%
13187   }
13188 }
13189 {}

```

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`.

```

13190 \ifcsdef{@glsstyle@long}
13191 {%
13192   \renewglossarystyle{long}{%
13193     \renewenvironment{theglossary}%
13194       {\begin{longtable}{lp{\glsdescwidth}}}%
13195       {\end{longtable}}%
13196     \renewcommand*\glossaryheader{}%
13197     \renewcommand*\glsgroupheading[1]{}%
13198     \renewcommand{\glossentry}[2]{%
13199       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13200       \glossentrydesc{##1}\glspostdescription
13201       \glxtrprelocation ##2\tabularnewline
13202     }%
13203     \renewcommand{\subglossentry}[3]{%
13204       &
13205       \glssubentryitem{##2}%
13206       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13207       \glxtrprelocation ##3\tabularnewline
13208     }%
13209     \ifglsnogroupskip
13210       \renewcommand*\glsgroupskip{}%
13211     \else

```

```

13212     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13213     \fi
13214 }
13215 }
13216 {}

```

Three column style:

```

13217 \ifcsdef{@glsstyle@long3col}
13218 {%
13219     \renewglossarystyle{long3col}{%
13220         \renewenvironment{theglossary}%
13221             {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13222             {\end{longtable}}}%
13223     \renewcommand*{\glossaryheader}{}%
13224     \renewcommand*{\glsgroupheading}[1]{}%
13225     \renewcommand{\glossentry}[2]{%
13226         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13227         \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13228     }%
13229     \renewcommand{\subglossentry}[3]{%
13230         &
13231         \glssubentryitem{##2}%
13232         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13233         ##3\tabularnewline
13234     }%

```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```

13235     \ifglsnogroupskip
13236         \renewcommand*{\glsgroupskip}{}%
13237     \else
13238         \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13239     \fi
13240 }
13241 }
13242 {}

```

Four column style:

```

13243 \ifcsdef{@glsstyle@long4col}
13244 {%
13245     \renewglossarystyle{long4col}{%
13246         \renewenvironment{theglossary}%
13247             {\begin{longtable}{llll}}}%
13248             {\end{longtable}}}%
13249     \renewcommand*{\glossaryheader}{}%
13250     \renewcommand*{\glsgroupheading}[1]{}%
13251     \renewcommand{\glossentry}[2]{%
13252         \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13253         \glossentrydesc{##1}\glspostdescription &
13254         \glossentrysymbol{##1} &
13255         ##2\tabularnewline

```

```

13256 }%
13257 \renewcommand{\subglossentry}[3]{%
13258     &
13259     \glssubentryitem{##2}%
13260     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13261     \glossentrysymbol{##2} & ##3\tabularnewline
13262 }%

13263 \ifglsgroupskip
13264     \renewcommand*\{glsgroupskip}{}%
13265 \else
13266     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13267 \fi
13268 }
13269 }
13270 {}

```

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`.

```

13271 \ifcsdef{@glstyle@longragged}
13272 {%
13273     \renewglossarystyle{longragged}{%
13274         \renewenvironment{theglossary}%
13275             {\begin{longtable}[l>\raggedright]p{\glstdescwidth}}}%
13276             {\end{longtable}}%
13277         \renewcommand*\{glossaryheader}{}%
13278         \renewcommand*\{glsgroupheading}[1]{}%
13279         \renewcommand{\glossentry}[2]{%
13280             \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13281             \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13282             \tabularnewline
13283         }%
13284         \renewcommand{\subglossentry}[3]{%
13285             &
13286             \glssubentryitem{##2}%
13287             \glstarget{##2}{\strut}\glossentrydesc{##2}%
13288             \glspostdescription\glxtrprelocation ##3%
13289             \tabularnewline
13290         }%
13291         \ifglsgroupskip
13292             \renewcommand*\{glsgroupskip}{}%
13293         \else
13294             \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13295 \fi
13296 }
13297 }
13298 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13299 \ifcsdef{@glsstyle@longragged3col}
13300 {%
13301 \renewglossarystyle{longragged3col}{%
13302 \renewenvironment{theglossary}%
13303 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13304 >{\raggedright}p{\glspagelistwidth}}}%
13305 {\end{longtable}}}%
13306 \renewcommand*{\glossaryheader}{}%
13307 \renewcommand*{\glsgroupheading}[1]{}%
13308 \renewcommand{\glossentry}[2]{%
13309 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13310 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13311 }%
13312 \renewcommand{\subglossentry}[3]{%
13313 &
13314 \glssubentryitem{##2}%
13315 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13316 ##3\tabularnewline
13317 }%
13318 \ifglsgroupskip
13319 \renewcommand*{\glsgroupskip}{}%
13320 \else
13321 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13322 \fi
13323 }
13324 }
13325 {}

```

Four column style:

```

13326 \ifcsdef{@glsstyle@altlongragged4col}
13327 {%
13328 \renewglossarystyle{altlongragged4col}{%
13329 \renewenvironment{theglossary}%
13330 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13331 >{\raggedright}p{\glspagelistwidth}}}%
13332 {\end{longtable}}}%
13333 \renewcommand*{\glossaryheader}{}%
13334 \renewcommand*{\glsgroupheading}[1]{}%
13335 \renewcommand{\glossentry}[2]{%
13336 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13337 \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13338 ##2\tabularnewline

```

```

13339 }%
13340 \renewcommand{\subglossentry}[3]{%
13341     &
13342     \glssubentryitem{##2}%
13343     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13344     \glossentrysymbol{##2} & ##3\tabularnewline
13345 }%

13346 \ifglsnogroupskip
13347     \renewcommand*\{glsgroupskip}{}%
13348 \else
13349     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13350 \fi
13351 }
13352 }
13353 {}

```

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`.

```

13354 \ifcsdef{@glstyle@super}
13355 {%
13356     \renewglossarystyle{super}{%
13357         \renewenvironment{theglossary}%
13358             {\tablehead{}}\tabletail{}}%
13359         \begin{supertabular}{lp{\glsdescwidth}}%
13360             {\end{supertabular}}%
13361         \renewcommand*\{glossaryheader}{}%
13362         \renewcommand*\{glsgroupheading}[1]{}%
13363         \renewcommand{\glossentry}[2]{%
13364             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13365             \glossentrydesc{##1}\glspostdescription
13366             \glxtrprelocation ##2\tabularnewline
13367         }%
13368         \renewcommand{\subglossentry}[3]{%
13369             &
13370             \glssubentryitem{##2}%
13371             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13372             \glxtrprelocation ##3\tabularnewline
13373         }%
13374         \ifglsnogroupskip
13375             \renewcommand*\{glsgroupskip}{}%
13376         \else
13377             \renewcommand*\{glsgroupskip}{& \tabularnewline}%
13378         \fi
13379     }
13380 }
13381 {}

```

Three column style:

```

13382 \ifcsdef{@glsstyle@super3col}
13383 {%
13384   \renewglossarystyle{super3col}{%
13385     \renewenvironment{theglossary}%
13386       {\tablehead{}\tabletail{}}%
13387       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13388     {\end{supertabular}}%
13389     \renewcommand*{\glossaryheader}{}%
13390     \renewcommand*{\glsgroupheading}[1]{}%
13391     \renewcommand{\glossentry}[2]{%
13392       \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13393       \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
13394     }%
13395     \renewcommand{\subglossentry}[3]{%
13396       &
13397       \glssubentryitem{##2}%
13398       \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13399       ##3\tabularnewline
13400     }%

13401     \ifglsgroupskip
13402       \renewcommand*{\glsgroupskip}{}%
13403     \else
13404       \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13405     \fi
13406   }
13407 }
13408 {}

```

Four column styles:

```

13409 \ifcsdef{@glsstyle@super4col}
13410 {%
13411   \renewglossarystyle{super4col}{%
13412     \renewenvironment{theglossary}%
13413       {\tablehead{}\tabletail{}}%
13414       \begin{supertabular}{llll}}}%
13415     \end{supertabular}}%
13416     \renewcommand*{\glossaryheader}{}%
13417     \renewcommand*{\glsgroupheading}[1]{}%
13418     \renewcommand{\glossentry}[2]{%
13419       \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13420       \glsentrydesc{##1}\glspostdescription &
13421       \glsentrysymbol{##1} & ##2\tabularnewline
13422     }%
13423     \renewcommand{\subglossentry}[3]{%
13424       &
13425       \glssubentryitem{##2}%
13426       \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13427       \glsentrysymbol{##2} & ##3\tabularnewline

```

```

13428 }%
13429 \ifglsgroupskip
13430 \renewcommand*{\glsgroupskip}{}%
13431 \else
13432 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13433 \fi
13434 }
13435 }
13436 {}

```

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`.

```

13437 \ifcsdef{@glstyle@superragged}
13438 {%
13439 \renewglossarystyle{superragged}{%
13440 \renewenvironment{theglossary}%
13441 {\tablehead{}\tabletail}%
13442 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
13443 {\end{supertabular}}%
13444 \renewcommand*{\glossaryheader}{}%
13445 \renewcommand*{\glsgroupheading}[1]{}%
13446 \renewcommand{\glossentry}[2]{%
13447 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13448 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
13449 \tabularnewline
13450 }%
13451 \renewcommand{\subglossentry}[3]{%
13452 &
13453 \glssubentryitem{##2}%
13454 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13455 \glstrprelocation ##3%
13456 \tabularnewline
13457 }%
13458 \ifglsgroupskip
13459 \renewcommand*{\glsgroupskip}{}%
13460 \else
13461 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13462 \fi
13463 }
13464 }
13465 {}

```

Three column style:

```

13466 \ifcsdef{@glstyle@superragged3col}
13467 {%

```

```

13468 \renewglossarystyle{superragged3col}{%
13469   \renewenvironment{theglossary}%
13470     {\tablehead{}\tabletail{}}%
13471     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
13472       >{\raggedright}p{\glspagelistwidth}}}%
13473     {\end{supertabular}}%
13474   \renewcommand*{\glossaryheader}{}%
13475   \renewcommand*{\glsgroupheading}[1]{}%
13476   \renewcommand{\glossentry}[2]{%
13477     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13478     \glossentrydesc{##1}\glspostdescription &
13479     ##2\tabularnewline
13480   }%
13481   \renewcommand{\subglossentry}[3]{%
13482     &
13483     \glssubentryitem{##2}%
13484     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13485     ##3\tabularnewline
13486   }%

13487   \ifglsgroupskip
13488     \renewcommand*{\glsgroupskip}{}%
13489   \else
13490     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13491   \fi
13492 }
13493 }
13494 {}

```

Four columns:

```

13495 \ifcsdef{@glsstyle@altsuperragged4col}
13496 {%
13497   \renewglossarystyle{altsuperragged4col}{%
13498     \renewenvironment{theglossary}%
13499       {\tablehead{}\tabletail{}}%
13500       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
13501         >{\raggedright}p{\glspagelistwidth}}}%
13502       {\end{supertabular}}%
13503     \renewcommand*{\glossaryheader}{}%
13504     \renewcommand{\glossentry}[2]{%
13505       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13506       \glossentrydesc{##1}\glspostdescription &
13507       \glossentrysymbol{##1} & ##2\tabularnewline
13508     }%
13509     \renewcommand{\subglossentry}[3]{%
13510       &
13511       \glssubentryitem{##2}%
13512       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13513       \glossentrysymbol{##2} & ##3\tabularnewline
13514     }%

```

```

13515 \ifglsnogroupskip
13516 \renewcommand*{\glsgroupskip}{}%
13517 \else
13518 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13519 \fi
13520 }
13521 }
13522 {}

```

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`.

```

13523 \ifdef{\@glsstyle@inline}
13524 {%
13525 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13526 \renewcommand*{\glsinlinedescformat}[3]{%
13527 \space#1\glxtrpostdescription}
13528 \renewcommand*{\glsinlinesubdescformat}[3]{%
13529 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13530 }
13531 {}

```

2.8 Tree Styles

Redefine both `\glstreenamefmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamefmt` to make it easier to change both at the same time or only change one without affecting the other.

```

13532 \ifdef\glstreenamefmt
13533 {

```

```

\glstreedefaultnamefmt

```

```

13534 \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}

```

```

\glstreenamefmt

```

```

13535 \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13536 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13537 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13538 }
13539 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13540 \ifdef{\@glsstyle@index}
13541 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13542 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13543 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13544 \renewglossarystyle{index}{%
13545   \renewenvironment{theglossary}%
13546     {\setlength{\parindent}{0pt}%
13547       \setlength{\parskip}{0pt plus 0.3pt}%
13548       \let\item\glstreeitem
13549       \let\subitem\glstreesubitem
13550       \let\subsubitem\glstreesubsubitem
13551     }%
13552   {\par}%
13553   \renewcommand*{\glossaryheader}{}%
13554   \renewcommand*{\glsgroupheading}[1]{}%
13555   \renewcommand*{\glossentry}[2]{}%
13556     \item\glssentryitem{##1}%
13557     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13558     \glstreesymbol{##1}%
13559     \glstreedesc{##1}%
13560     \glstreeprelocation ##2%
13561   }%
13562   \renewcommand{\subglossentry}[3]{}%
13563     \ifcase##1\relax
13564       \item
13565     \or
13566       \subitem
13567       \glssubentryitem{##2}%
13568     \else
13569       \subsubitem
13570     \fi
13571     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13572     \glstreechildsymbol{##2}%
13573     \glstreechilddesc{##2}%
13574     \glstreechildprelocation ##3%
13575   }%
```

```

13576 \renewcommand*{\glsgroupskip}{\ifglsgnigroupskip\else\indexspace\fi}%
13577 }
13578 }
13579 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13580 \ifdef{\@glsstyle@indexgroup}
13581 {%
13582 \renewglossarystyle{indexgroup}{%
13583 \setglossarystyle{index}%
13584 \renewcommand*{\glsgroupheading}[1]{%
13585 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13586 \nopagebreak\indexspace
13587 \nobreak\@afterheading
13588 }%
13589 }
13590 }
13591 {}

```

Similarly for indexhypergroup.

```

13592 \ifdef{\@glsstyle@indexhypergroup}
13593 {%
13594 \renewglossarystyle{indexhypergroup}{%
13595 \setglossarystyle{index}%
13596 \renewcommand*{\glossaryheader}{%
13597 \item\glstreenavigationfmt{\glsnavigation}%
13598 \nobreak\@afterheading\indexspace}%
13599 \renewcommand*{\glsgroupheading}[1]{%
13600 \item\glstreegroupheaderfmt
13601 {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13602 \nopagebreak\indexspace
13603 \nobreak\@afterheading}%
13604 }%
13605 }
13606 {}

```

Adjust tree style to remove hard coded space before number list.

```

13607 \ifdef{\@glsstyle@tree}
13608 {%
13609 %Provide a command for use with the \glostyle{tree} styles that displays
13610 %the pre-description separator, the
13611 %description and post-description hook.
13612 %\begin{macro}{\glstreedesc}
13613 %\changes{1.31}{2018-05-09}{new}
13614 % \begin{macrocode}
13615 \newcommand{\glstreedesc}[1]{%
13616 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13617 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13618 \newcommand{\glstreesymbol}[1]{%
13619   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13620 }%
```

And for the child entries:

`lstreechilddesc`

```
13621 \newcommand{\glstreechilddesc}[1]{%
13622   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13623 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13624 \newcommand{\glstreechildsymbol}[1]{%
13625   \glstreesymbol{#1}%
13626 }%

13627 \renewglossarystyle{tree}{%
13628   \renewenvironment{theglossary}%
13629     {\setlength{\parindent}{0pt}%
13630     \setlength{\parskip}{0pt plus 0.3pt}}%
13631   {%
13632     \renewcommand*{\glossaryheader}{}%
13633     \renewcommand*{\glsgroupheading}[1]{}%
13634     \renewcommand{\glossentry}[2]{%
13635       \hangindent0pt\relax
13636       \parindent0pt\relax
13637       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13638       \glstreesymbol{##1}%
13639       \glstreedesc{##1}%
13640       \glstreeprelocation##2\par
13641     }%
13642     \renewcommand{\subglossentry}[3]{%
13643       \hangindent##1\glstreeindent\relax
13644       \parindent##1\glstreeindent\relax
13645       \ifnum##1=1\relax
13646         \glssubentryitem{##2}%
13647       \fi
13648       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13649       \glstreechildsymbol{##2}%
13650       \glstreechilddesc{##2}%
13651       \glstreechildprelocation ##3\par
13652     }%
13653     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13654   }%
13655 }
13656 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13657 \ifdef{\@glsstyle@treegroup}
```

```

13658 {%
13659   \renewglossarystyle{treegroup}{%
13660     \setglossarystyle{tree}%
13661     \renewcommand{\glsgroupheading}[1]{\par
13662       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13663       \nopagebreak\indexspace\nobreak\@afterheading}%
13664   }
13665 }
13666 {}

```

Similarly for treehypergroup

```

13667 \ifdef{\@glsstyle@treehypergroup}
13668 {%
13669   \renewglossarystyle{treehypergroup}{%
13670     \setglossarystyle{tree}%
13671     \renewcommand*\{\glossaryheader}{%
13672       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13673       \nobreak\@afterheading\indexspace}%
13674     \renewcommand*\{\glsgroupheading}[1]{%
13675       \par\noindent
13676       \glstreegroupheaderfmt
13677         {\glstravhypertarget{##1}{\glsgrouptitle{##1}}}\par
13678       \nopagebreak\indexspace\nobreak\@afterheading}%
13679   }
13680 }
13681 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13682 \ifdef{\@glsstyle@treenoname}
13683 {%
13684 %Provide a command for use with the \glsstyle{treenoname} styles that displays
13685 %the pre-description separator, the
13686 %description and post-description hook.
13687 %\begin{macro}{\glstreenonamedesc}
13688 %\changes{1.31}{2018-05-09}{new}
13689 %   \begin{macrocode}
13690   \newcommand{\glstreenonamedesc}[1]{%
13691     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13692   }%

```

Similarly for the symbol.

treenonamesymbol

```

13693   \newcommand{\glstreenonamesymbol}[1]{%
13694     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13695   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13696   \newcommand{\glstreenonamechilddesc}[1]{%
13697     \glossentrydesc{#1}\glspostdescription
13698   }%

```

```

13699 \renewglossarystyle{treenoname}{%
13700   \renewenvironment{theglossary}%
13701     {\setlength{\parindent}{0pt}%
13702      \setlength{\parskip}{0pt plus 0.3pt}}%
13703     {}%
13704   \renewcommand*{\glossaryheader}{}%
13705   \renewcommand*{\glsgroupheading}[1]{}%
13706   \renewcommand{\glossentry}[2]{%
13707     \hangindent0pt\relax
13708     \parindent0pt\relax
13709     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13710     \glstreenonamesymbol{##1}%
13711     \glstreenonamedesc{##1}%
13712     \glstreeprelocation##2\par
13713   }%
13714   \renewcommand{\subglossentry}[3]{%
13715     \hangindent##1\glstreeindent\relax
13716     \parindent##1\glstreeindent\relax
13717     \ifnum##1=1\relax
13718       \glssubentryitem{##2}%
13719     \fi
13720     \glstarget{##2}{\strut}%
13721     \glstreenonamechilddesc{##2}%
13722     \glstreechildprelocation##3\par
13723   }%
13724   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13725 }
13726 }
13727 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13728 \ifdef{\@glsstyle@treenonamegroup}
13729 {%
13730   \renewglossarystyle{treenonamegroup}{%
13731     \setglossarystyle{treenoname}%
13732     \renewcommand{\glsgroupheading}[1]{\par
13733       \noindent\glstreegroupheaderfmt
13734       {\glsgetgrouptitle{##1}}}%
13735     \nopagebreak\indexspace\nobreak\@afterheading
13736   }%
13737 }
13738 }
13739 {}

```

Similarly for `treenonamehypergroup`

```

13740 \ifdef{\@glsstyle@treenonamehypergroup}
13741 {%
13742   \renewglossarystyle{treenonamehypergroup}{%
13743     \setglossarystyle{treenoname}%
13744     \renewcommand*{\glossaryheader}{%

```

```

13745      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13746      \nobreak\@afterheading\indexspace}%
13747      \renewcommand*{\glsgroupheading}[1]{%
13748      \par\noindent
13749      \glstreegroupheaderfmt
13750      {\glsnavhypertarget{##1}{\glsgroupgetgrouptitle{##1}}}%
13751      \nopagebreak\indexspace\nobreak\@afterheading}%
13752  }
13753 }
13754 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13755 \ifdef{\glstyle@almtree}
13756 {}%

```

Only redefine this style if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

13757 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13758   {%
13759     \let\par\glxtrAltTreePar
13760     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13761     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13762   }%
13763 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13764 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13765 \newcommand{\glxtrAltTreePar}{%
13766   \@@par
13767   \glxtrAltTreeSetHangIndent
13768   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13769 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13770 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13771   \glxtralttreeSymbolDescLocation{#2}{#3}%
13772 }

```

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.

```
13773 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13774 \newcommand*\glxtralttreeInit{%
13775   \settowidth{\glxstrtreetopindent}{\glstreenamfmt{\glsggetwidestname\space}}%
13776   \glxstrAltTreeIndent=\parindent
13777 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13778 \newcommand*\gglsetwidest}[2][0]{%
13779   \csgdef{@glswidestname\romannumeral#1}{#2}%
13780 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13781 \newcommand*\eglssetwidest}[2][0]{%
13782   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13783 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13784 \newcommand*\xglsetwidest}[2][0]{%
13785   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13786 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13787 \newcommand*\glupdatewidest}[2][0]{%
13788   \ifcsundef{@glswidestname\romannumeral#1}%
13789   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13790   {%
13791     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13792     \settowidth{\dimen@ii}{#2}%
13793     \ifdim\dimen@ii>\dimen@
13794       \csdef{@glswidestname\romannumeral#1}{#2}%
13795     \fi
13796   }%
13797 }
```

glupdatewidest As above but global definition.

```
13798 \newcommand*\gglupdatewidest}[2][0]{%
13799   \ifcsundef{@glswidestname\romannumeral#1}%
13800   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13801   {%
13802     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13803     \settowidth{\dimen@ii}{#2}%
13804     \ifdim\dimen@ii>\dimen@
13805       \csgdef{@glswidestname\romannumeral#1}{#2}%
13806     \fi
```

```

13807 }%
13808 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13809 \newcommand*{\eglsupdatewidest}[2][0]{%
13810 \ifcsundef{@glswidestname\romannumeral#1}%
13811 {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13812 {%
13813 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13814 \settowidth{\dimen@ii}{#2}%
13815 \ifdim\dimen@ii>\dimen0
13816 \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13817 \fi
13818 }%
13819 }

```

`glsupdatewidest` As above but global.

```

13820 \newcommand*{\xglsupdatewidest}[2][0]{%
13821 \ifcsundef{@glswidestname\romannumeral#1}%
13822 {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13823 {%
13824 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13825 \settowidth{\dimen@ii}{#2}%
13826 \ifdim\dimen@ii>\dimen0
13827 \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13828 \fi
13829 }%
13830 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13831 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13832 \newcommand*{\glsgetwidestsubname}[1]{%
13833 \ifcsundef{@glswidestname\romannumeral#1}%
13834 {\@glswidestname}%
13835 {\csuse{@glswidestname\romannumeral#1}}%
13836 }

```

`glsFindWidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```

13837 \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.

```

13838 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
13839 \dimen@=0pt\relax

```

```

13840 \gls@tmplen=Opt\relax
13841 \foralllglossaries[#1]{\@gls@type}%
13842 {%
13843   \forglsentries[\@gls@type]{\@glo@label}%
13844   {%
13845     \ifglsused{\@glo@label}%
13846     {%
13847       \ifglshasparent{\@glo@label}%
13848       {}%
13849       {%
13850         \settowidth{\dimen@}%
13851           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13852         \ifdim\dimen@>\gls@tmplen
13853           \gls@tmplen=\dimen@
13854         \eglssetwidest{\glsentryname{\@glo@label}}%
13855         \fi
13856       }%
13857     }%
13858   }%
13859 }%
13860 }%
13861 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13862 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13863   \dimen@=Opt\relax
13864   \gls@tmplen=Opt\relax
13865   \foralllglossaries[#1]{\@gls@type}%
13866   {%
13867     \forglsentries[\@gls@type]{\@glo@label}%
13868     {%
13869       \ifglsused{\@glo@label}%
13870       {%
13871         \settowidth{\dimen@}%
13872           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13873         \ifdim\dimen@>\gls@tmplen
13874           \gls@tmplen=\dimen@
13875         \eglssetwidest{\glsentryname{\@glo@label}}%
13876         \fi
13877       }%
13878     }%
13879   }%
13880 }%
13881 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

13882 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13883   \dimen@=Opt\relax

```

```

13884 \gls@tmplen=0pt\relax
13885 \foralllglossaries[#1]{\@gls@type}%
13886 {%
13887   \forallgsentries[\@gls@type]{\@glo@label}%
13888   {%
13889     \settowidth{\dimen@}%
13890     {\glstreenamfmt{\glstryname{\@glo@label}}}%
13891     \ifdim\dimen@>\gls@tmplen
13892       \gls@tmplen=\dimen@
13893       \eglssetwidest{\glstryname{\@glo@label}}%
13894     \fi
13895   }%
13896 }%
13897 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

13898 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13899   \dimen@=0pt\relax
13900   \dimen@i=0pt\relax
13901   \dimen@ii=0pt\relax
13902   \foralllglossaries[#1]{\@gls@type}%
13903   {%
13904     \forallgsentries[\@gls@type]{\@glo@label}%
13905     {%
13906       \ifglsused{\@glo@label}%
13907       {%
13908         \ifglshasparent{\@glo@label}%
13909         {%
13910           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
13911           \ifglshasparent{\@glo@parent}%
13912           {%
13913             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
13914             \ifglshasparent{\@glo@parent}%
13915             }%
13916           {%
13917             \settowidth{\gls@tmplen}%
13918             {\glstreenamfmt{\glstryname{\@glo@label}}}%
13919             \ifdim\gls@tmplen>\dimen@ii
13920               \dimen@ii=\gls@tmplen
13921               \eglssetwidest[2]{\glstryname{\@glo@label}}%
13922             \fi
13923           }%
13924         }%
13925       }%
13926       \settowidth{\gls@tmplen}%
13927       {\glstreenamfmt{\glstryname{\@glo@label}}}%
13928       \ifdim\gls@tmplen>\dimen@i
13929         \dimen@i=\gls@tmplen

```

```

13930         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13931     \fi
13932 }%
13933 }%
13934 {%
13935     \settowidth{\gls@tmplen}%
13936         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13937     \ifdim\gls@tmplen>\dimen@
13938         \dimen@=\gls@tmplen
13939         \eglssetwidest{\glsentryname{\@glo@label}}%
13940     \fi
13941 }%
13942 }%
13943 {}%
13944 }%
13945 }%
13946 }

```

dWidestLevelTwo This is like \glsFindWidestUsedLevelTwo but doesn't check if the entry has been used.

```

13947 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13948     \dimen@=0pt\relax
13949     \dimen@i=0pt\relax
13950     \dimen@ii=0pt\relax
13951     \foralllglossaries[#1]{\@gls@type}%
13952     {%
13953         \forglsentries[\@gls@type]{\@glo@label}%
13954         {%
13955             \ifglshasparent{\@glo@label}%
13956             {%
13957                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13958                 \ifglshasparent{\@glo@parent}%
13959                 {%
13960                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13961                     \ifglshasparent{\@glo@parent}%
13962                     {}%
13963                     {%
13964                         \settowidth{\gls@tmplen}%
13965                             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13966                         \ifdim\gls@tmplen>\dimen@ii
13967                             \dimen@ii=\gls@tmplen
13968                             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13969                         \fi
13970                     }%
13971                 }%
13972             }%
13973             \settowidth{\gls@tmplen}%
13974                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13975             \ifdim\gls@tmplen>\dimen@i
13976                 \dimen@i=\gls@tmplen

```

```

13977         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13978     \fi
13979 }%
13980 }%
13981 {%
13982     \settowidth{\gls@tmplen}%
13983         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13984     \ifdim\gls@tmplen>\dimen@
13985         \dimen@=\gls@tmplen
13986         \eglssetwidest{\glsentryname{\@glo@label}}%
13987     \fi
13988 }%
13989 }%
13990 }%
13991 }

```

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.

```

13992 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13993     \dimen@=0pt\relax
13994     \gls@tmplen=0pt\relax
13995     #2=0pt\relax
13996     \foralllglossaries[#1]{\@gls@type}%
13997     {%
13998         \forglsentries[\@gls@type]{\@glo@label}%
13999         {%
14000             \ifglsused{\@glo@label}%
14001             {%
14002                 \settowidth{\dimen@}%
14003                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14004                 \ifdim\dimen@>\gls@tmplen
14005                     \gls@tmplen=\dimen@
14006                     \eglssetwidest{\glsentryname{\@glo@label}}%
14007                 \fi
14008                 \settowidth{\dimen@}%
14009                     {\glsentrysymbol{\@glo@label}}%
14010                 \ifdim\dimen@>#2\relax
14011                     #2=\dimen@
14012                 \fi
14013             }%
14014         }%
14015     }%
14016 }%
14017 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

14018 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14019     \dimen@=0pt\relax
14020     \gls@tmplen=0pt\relax

```

```

14021     #2=Opt\relax
14022     \forallglossaries[#1]{\@gls@type}%
14023     {%
14024         \forglsentries[\@gls@type]{\@glo@label}%
14025         {%
14026             \settowidth{\dimen@}%
14027             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14028             \ifdim\dimen@>\gls@tmplen
14029                 \gls@tmplen=\dimen@
14030                 \eglssetwidest{\glsentryname{\@glo@label}}%
14031             \fi
14032             \settowidth{\dimen@}%
14033             {\glsentrysymbol{\@glo@label}}%
14034             \ifdim\dimen@>#2\relax
14035                 #2=\dimen@
14036             \fi
14037         }%
14038     }%
14039 }

```

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.

```

14040 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14041     \dimen@=Opt\relax
14042     \gls@tmplen=Opt\relax
14043     #2=Opt\relax
14044     #3=Opt\relax
14045     \forallglossaries[#1]{\@gls@type}%
14046     {%
14047         \forglsentries[\@gls@type]{\@glo@label}%
14048         {%
14049             \ifglsused{\@glo@label}%
14050             {%
14051                 \settowidth{\dimen@}%
14052                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14053                 \ifdim\dimen@>\gls@tmplen
14054                     \gls@tmplen=\dimen@
14055                     \eglssetwidest{\glsentryname{\@glo@label}}%
14056                 \fi
14057                 \settowidth{\dimen@}%
14058                 {\glsentrysymbol{\@glo@label}}%
14059                 \ifdim\dimen@>#2\relax
14060                     #2=\dimen@
14061                 \fi
14062                 \settowidth{\dimen@}%
14063                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14064                 \ifdim\dimen@>#3\relax

```

```

14065         #3=\dimen@
14066         \fi
14067     }%
14068     {}%
14069 }%
14070 }%
14071 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14072 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14073     \dimen@=0pt\relax
14074     \gls@tmplen=0pt\relax
14075     #2=0pt\relax
14076     #3=0pt\relax
14077     \forallglossaries[#1]{\@gls@type}%
14078     {%
14079         \forglsentries[\@gls@type]{\@glo@label}%
14080         {%
14081             \settowidth{\dimen@}%
14082             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14083             \ifdim\dimen@>\gls@tmplen
14084                 \gls@tmplen=\dimen@
14085                 \eglssetwidest{\glsentryname{\@glo@label}}%
14086             \fi
14087             \settowidth{\dimen@}%
14088             {\glsentrysymbol{\@glo@label}}%
14089             \ifdim\dimen@>#2\relax
14090                 #2=\dimen@
14091             \fi
14092             \settowidth{\dimen@}%
14093             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14094             \ifdim\dimen@>#3\relax
14095                 #3=\dimen@
14096             \fi
14097         }%
14098     }%
14099 }

```

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.

```

14100 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14101     \dimen@=0pt\relax
14102     \gls@tmplen=0pt\relax
14103     #2=0pt\relax
14104     \forallglossaries[#1]{\@gls@type}%
14105     {%
14106         \forglsentries[\@gls@type]{\@glo@label}%
14107         {%

```

```

14108      \ifglsused{\@glo@label}%
14109      {%
14110      \settowidth{\dimen@}%
14111      {\glstreenamfmt{\glentryname{\@glo@label}}}%
14112      \ifdim\dimen@>\gls@tmplen
14113      \gls@tmplen=\dimen@
14114      \eglssetwidest{\glentryname{\@glo@label}}%
14115      \fi
14116      \settowidth{\dimen@}%
14117      {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14118      \ifdim\dimen@>#2\relax
14119      #2=\dimen@
14120      \fi
14121      }%
14122      {}%
14123      }%
14124      }%
14125      }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14126 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14127   \dimen@=0pt\relax
14128   \gls@tmplen=0pt\relax
14129   #2=0pt\relax
14130   \forallglossaries[#1]{\@gls@type}%
14131   {%
14132     \forglsentries[\@gls@type]{\@glo@label}%
14133     {%
14134       \settowidth{\dimen@}%
14135       {\glstreenamfmt{\glentryname{\@glo@label}}}%
14136       \ifdim\dimen@>\gls@tmplen
14137       \gls@tmplen=\dimen@
14138       \eglssetwidest{\glentryname{\@glo@label}}%
14139       \fi
14140       \settowidth{\dimen@}%
14141       {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14142       \ifdim\dimen@>#2\relax
14143       #2=\dimen@
14144       \fi
14145     }%
14146   }%
14147   }

```

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`.

```

14148 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
14149   \glstreeindent=\glxtrtreetopindent\relax
14150 }

```

uteTreeSubIndent

```
\glxtrComputeTreeSubIndent{<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.

```
14151 \newcommand*{\glxtrComputeTreeSubIndent}[3]{%
14152   \ifcsundef{@glswidestname\romannumeral#1}%
14153   {%
14154     \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14155   }%
14156   {%
14157     \settowidth{#3}{\glstreenamefmt{
14158       \csname @glswidestname\romannumeral#1\endcsname\space}}%
14159   }%
14160 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14161 \newcommand*{\glxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14162 \newcommand*{\glxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14163 \renewglossarystyle{alttree}{%
14164   \renewenvironment{theglossary}%
14165   {%
14166     \glxtralttreeInit
14167     \def\@gls@prevlevel{-1}%
14168     \mbox{}\par}%
14169   {\par}%
14170   \renewcommand*{\glossaryheader}{}%
14171   \renewcommand*{\glsgroupheading}[1]{}%
14172   \renewcommand{\glossentry}[2]{%
14173     \ifnum\@gls@prevlevel=0\relax
14174     \else
14175       \glxtrComputeTreeIndent{##1}%
14176     \fi
14177     \parindent\glstreeindent
14178     \glxtrAltTreeSetHangIndent
14179     \makebox[Opt][r]%
14180     {%
14181       \glstreenamebox{\glstreeindent}%
14182     }%
14183     \glsentryitem{##1}%
14184     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14185   }%
14186 }
```

```

14187     \glxtralttreeSymbolDescLocation{##1}{##2}%
14188     \def\@gls@prevlevel{0}%
14189 }
14190 \renewcommand{\subglossentry}[3]{%
14191     \ifnum##1=1\relax
14192         \glssubentryitem{##2}%
14193     \fi
14194     \ifnum\@gls@prevlevel=##1\relax
14195     \else
14196         \glxtrComputeTreeSubIndent{##1}{##2}{\@gls@tmplen}%
14197         \ifnum\@gls@prevlevel<##1\relax
14198             \setlength\glstreeindent\@gls@tmplen
14199             \addtolength\glstreeindent\parindent
14200             \parindent\glstreeindent
14201         \else
14202             \ifnum\@gls@prevlevel=0\relax
14203                 \glxtrComputeTreeIndent{##2}%
14204             \else
14205                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14206             \fi
14207             \addtolength\parindent{-\glstreeindent}%
14208             \setlength\glstreeindent\parindent
14209         \fi
14210     \fi
14211     \glxtrAltTreeSetSubHangIndent{##1}%
14212     \makebox[0pt][r]{\glstreenamebox{\@gls@tmplen}{%
14213         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14214     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14215     \def\@gls@prevlevel{##1}%
14216 }%
14217 \renewcommand*\@gls@groupskip{\ifglsnogroupskip\else\indexspace\fi}%
14218 }
14219 }%
14220 {%
14221 }

```

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.

```

14222 \ifdef{\@glsstyle@alttreegroup}
14223 {%
14224     \renewglossarystyle{alttreegroup}{%
14225         \setglossarystyle{alttree}%
14226         \renewcommand{\glsgroupheading}[1]{\par
14227             \def\@gls@prevlevel{-1}%
14228             \hangindent0pt\relax
14229             \parindent0pt\relax
14230             \glstreegroupheaderfmt{\glsgrouptitle{##1}}%
14231             \nopagebreak\indexspace\nopagebreak
14232         }%
14233     }%

```

```

14234 }%
14235 {%
14236 }

```

Similarly for `alttreehypergroup`.

```

14237 \ifdef{\@glsstyle@alttreehypergroup}
14238 {%
14239   \renewglossarystyle{alttreehypergroup}{%
14240     \setglossarystyle{alttree}%
14241     \renewcommand*{\glossaryheader}{%
14242       \par
14243       \def\@gls@prevlevel{-1}%
14244       \hangindent0pt\relax
14245       \parindent0pt\relax
14246       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14247     }%
14248     \renewcommand*{\glsgroupheading}[1]{%
14249       \par
14250       \def\@gls@prevlevel{-1}%
14251       \hangindent0pt\relax
14252       \parindent0pt\relax
14253       \glstreegroupheaderfmt
14254       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14255       \nopagebreak\indexspace\nopagebreak
14256     }%
14257   }
14258 }%
14259 {%
14260 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14261 \ifdef{\@glsstyle@mcolindexgroup}
14262 {%
14263   \renewglossarystyle{mcolindexgroup}{%
14264     \setglossarystyle{mcolindex}%
14265     \renewcommand*{\glsgroupheading}[1]{%
14266       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14267       \nopagebreak\indexspace\nobreak\@afterheading
14268     }%
14269   }
14270 }%
14271 {%
14272 }

```

Similarly for `mcolindexhypergroup`.

```

14273 \ifdef{\@glsstyle@mcolindexhypergroup}
14274 {%

```

```

14275 \renewglossarystyle{mcolindexhypergroup}{%
14276   \setglossarystyle{mcolindex}%
14277   \renewcommand*{\glossaryheader}{%
14278     \item\glstreenavigationfmt{\glsnavigation}%
14279     \indexspace
14280   }%
14281   \renewcommand*{\glsgroupheading}[1]{%
14282     \item\glstreegroupheaderfmt
14283       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14284     \nopagebreak\indexspace\nobreak\@afterheading
14285   }%
14286 }
14287 }%
14288 {%
14289 }

```

Similarly for mcolindexspannav.

```

14290 \ifdef{\@glsstyle@mcolindexspannav}
14291 {%
14292   \renewglossarystyle{mcolindexspannav}{%
14293     \setglossarystyle{index}%
14294     \renewenvironment{theglossary}%
14295     {%
14296       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14297       \setlength{\parindent}{0pt}%
14298       \setlength{\parskip}{0pt plus 0.3pt}%
14299       \let\item\glstreeitem}%
14300     {\end{multicols}}}%
14301   \renewcommand*{\glsgroupheading}[1]{%
14302     \item\glstreegroupheaderfmt
14303       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14304     \nopagebreak\indexspace\nobreak\@afterheading
14305   }%
14306 }
14307 }%
14308 {%
14309 }

```

Similarly for mcoltreegroup.

```

14310 \ifdef{\@glsstyle@mcoltreegroup}
14311 {%
14312   \renewglossarystyle{mcoltreegroup}{%
14313     \setglossarystyle{mcoltree}%
14314     \renewcommand{\glsgroupheading}[1]{\par
14315       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14316     \nopagebreak\indexspace\nobreak\@afterheading
14317   }%
14318 }
14319 }%
14320 {%

```

14321 }

Similarly for mcoltreehypergroup.

```
14322 \ifdef{\@glsstyle@mcoltreehypergroup}
14323 {%
14324   \renewglossarystyle{mcoltreehypergroup}{%
14325     \setglossarystyle{mcoltree}%
14326     \renewcommand*{\glossaryheader}{%
14327       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14328     }%
14329     \renewcommand*{\glsgroupheading}[1]{%
14330       \par\noindent
14331       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14332       \nopagebreak\indexspace\nobreak\@afterheading
14333     }%
14334   }
14335 }%
14336 {%
14337 }
```

Similarly for mcoltreespannav.

```
14338 \ifdef{\@glsstyle@mcoltreespannav}
14339 {%
14340   \renewglossarystyle{mcoltreespannav}{%
14341     \setglossarystyle{tree}%
14342     \renewenvironment{theglossary}%
14343     {%
14344       \begin{multicols}{\glsmcols}%
14345       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14346       \setlength{\parindent}{0pt}%
14347       \setlength{\parskip}{0pt plus 0.3pt}%
14348     }%
14349     {\end{multicols}}%
14350     \renewcommand*{\glsgroupheading}[1]{%
14351       \par\noindent
14352       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14353       \nopagebreak\indexspace\nobreak\@afterheading
14354     }%
14355   }
14356 }%
14357 {%
14358 }
```

Similarly for mcoltreenonamegroup.

```
14359 \ifdef{\@glsstyle@mcoltreenonamegroup}
14360 {%
14361   \renewglossarystyle{mcoltreenonamegroup}{%
14362     \setglossarystyle{mcoltreenoname}%
14363     \renewcommand{\glsgroupheading}[1]{\par
14364       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14365     \nopagebreak\indexspace\nobreak\@afterheading
14366   }
```

```

14366 }%
14367 }
14368 }%
14369 {%
14370 }

```

Similarly for mcoltreenonamehypergroup.

```

14371 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14372 {%
14373   \renewglossarystyle{mcoltreenonamehypergroup}{%
14374     \setglossarystyle{mcoltreenoname}%
14375     \renewcommand*{\glossaryheader}{%
14376       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14377     \renewcommand*{\glsgroupheading}[1]{%
14378       \par\noindent
14379       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14380       \nopagebreak\indexspace\nobreak\@afterheading}%
14381   }
14382 }%
14383 {%
14384 }

```

Similarly for mcoltreenonamespannav.

```

14385 \ifdef{\@glsstyle@mcoltreenonamespannav}
14386 {%
14387   \renewglossarystyle{mcoltreenonamespannav}{%
14388     \setglossarystyle{treenoname}%
14389     \renewenvironment{theglossary}%
14390     {%
14391       \begin{multicols}{\glsmcols}%
14392       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14393       \setlength{\parindent}{0pt}%
14394       \setlength{\parskip}{0pt plus 0.3pt}%
14395     }%
14396     {\end{multicols}}}%
14397   \renewcommand*{\glsgroupheading}[1]{%
14398     \par\noindent
14399     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14400     \nopagebreak\indexspace\nobreak\@afterheading}%
14401   }
14402 }%
14403 {%
14404 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{}\par which would unbalance the top of the columns.

```

14405 \ifdef{\@glsstyle@mcolalttree}
14406 {%
14407   \renewglossarystyle{mcolalttree}{%
14408     \setglossarystyle{alttree}%
14409     \renewenvironment{theglossary}%

```

```

14410   {%
14411       \glstralttreeInit
14412       \def\@gls@prevlevel{-1}%
14413       \begin{multicols}{\glsmcols}%
14414   }%
14415   {\par\end{multicols}}%
14416 }
14417 }%
14418 {%
14419 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

14420 \ifdef{\@glsstyle@mcolalmtreegroup}
14421 {%
14422   \renewglossarystyle{mcolalmtreegroup}{%
14423       \setglossarystyle{mcolalmtree}%
14424       \renewcommand{\glsgroupheading}[1]{\par
14425           \def\@gls@prevlevel{-1}%
14426           \hangindent0pt\relax
14427           \parindent0pt\relax
14428           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14429       \nopagebreak\indexspace\nopagebreak
14430   }%
14431 }
14432 }%
14433 {%
14434 }

```

Similarly for mcolalmtreehypergroup.

```

14435 \ifdef{\@glsstyle@mcolalmtreehypergroup}
14436 {%
14437   \renewglossarystyle{mcolalmtreehypergroup}{%
14438       \setglossarystyle{mcolalmtree}%
14439       \renewcommand*\glossaryheader{%
14440           \par
14441           \def\@gls@prevlevel{-1}%
14442           \hangindent0pt\relax
14443           \parindent0pt\relax
14444           \glstreenavigationfmt{\glsnavigation}%
14445           \par\indexspace
14446       }%
14447       \renewcommand*\glsgroupheading[1]{%
14448           \par
14449           \def\@gls@prevlevel{-1}%
14450           \hangindent0pt\relax
14451           \parindent0pt\relax
14452           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14453           \nopagebreak\indexspace\nopagebreak
14454       }%
14455   }

```

```

14456 }%
14457 {%
14458 }

```

Similarly for mcolalttreespannav.

```

14459 \ifdef{\@glsstyle@mcolalttreespannav}
14460 {%
14461   \renewglossarystyle{mcolalttreespannav}{%
14462     \setglossarystyle{almtree}%
14463     \renewenvironment{theglossary}%
14464     {%
14465       \glstralmtreeInit
14466       \def\@gls@prevlevel{-1}%
14467       \begin{multicols}{\glsmcols}%
14468         [\noindent\glstreenavigationfmt{\glsnavigation}]%
14469     }%
14470     {\par\end{multicols}}%
14471     \renewcommand*{\glsgroupheading}[1]{%
14472       \par
14473       \def\@gls@prevlevel{-1}%
14474       \hangindent0pt\relax
14475       \parindent0pt\relax
14476       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14477       \nopagebreak\indexspace\nopagebreak
14478     }%
14479   }
14480 }%
14481 {%
14482 }

```

Reset the default style

```

14483 \ifx\@glossary@default@style\relax
14484 \else
14485   \setglossarystyle{\@glstr@current@style}
14486 \fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
14487 \NeedsTeXFormat{LaTeX2e}
14488 \ProvidesPackage{glossary-bookindex}[2018/08/13 v1.35 (NLCT)]
```

Load required packages.

```
14489 \RequirePackage{multicol}
14490 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14491 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14492 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14493 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14494 \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.

```
14495 \newcommand*{\glstrbookindexprelocation}[1]{%
14496   \glstrifhasfield{location}{#1}%
14497   {,\glstrprelocation}%
14498   {\glstrprelocation}%
14499 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14500 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14501   \glstrbookindexprelocation{#1}%
14502 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14503 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14504 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
14505 \newcommand{\glxstrbookindexbetween}[2]{}

indexsubbetween Between two level 1 entries identified by the labels in the arguments.
14506 \newcommand{\glxstrbookindexsubbetween}[2]{}

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.
14507 \newcommand{\glxstrbookindexsubsubbetween}[2]{}

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.
14508 \newcommand{\glxstrbookindexatendgroup}[1]{}

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.
14509 \newcommand{\glxstrbookindexsubatendgroup}[1]{}

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.
14510 \newcommand{\glxstrbookindexsubsubatendgroup}[1]{}

kindexgroupskip Group separator.
14511 \newcommand{\glxstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

Format group title.

dexformatheader Group separator.
14512 \newcommand*{\glxstrbookindexformatheader}[1]{%
14513 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14514 }

bookindexbookmark Book mark group heading if supported.
14515 \ifdef\pdfbookmark
14516 {%
14517 \newcommand*{\glxstrbookindexbookmark}[2]{%
14518 \ifdefstring{\@@glossarysec}{chapter}%
14519 {\pdfbookmark[1]{#1}{#2}}%
14520 {\pdfbookmark[2]{#1}{#2}}%
14521 }
14522 }
14523 {%
14524 \newcommand*{\glxstrbookindexbookmark}[2]{}
14525 }

kindexcolspread
14526 \newcommand*{\glxstrbookindexcolspread}{}

dexmulticolenv
14527 \newcommand*{\glxstrbookindexmulticolenv}{multicols}
```

Define the style.

```

14528 \newglossarystyle{bookindex}{%
14529   \setglossarystyle{index}%
14530   \renewenvironment{theglossary}%
14531   {%
14532     \ifdefempty{glstrbookindexcolspread}
14533     {%
14534       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14535       {\glstrbookindexcols}%
14536     }%
14537     {%
14538       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14539       {\glstrbookindexcols}[\glstrbookindexcolspread]%
14540     }%
14541     \setlength{\parindent}{0pt}%
14542     \setlength{\parskip}{0pt plus 0.3pt}%
14543     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14544     \let\@glstrbookindex@subsep\glstrbookindexparentschildsep
14545     \let\@glstrbookindex@between\@gobble
14546     \let\@glstrbookindex@subbetween\@gobble
14547     \let\@glstrbookindex@subsubbetween\@gobble
14548     \let\@glstrbookindex@atendgroup\relax
14549     \let\@glstrbookindex@subatendgroup\relax
14550     \let\@glstrbookindex@subsubatendgroup\relax
14551     \let\@glstrbookindex@groupskip\relax
14552   }%
14553   {%

```

Do end group hooks.

```

14554     \@glstrbookindex@subsubatendgroup
14555     \@glstrbookindex@subatendgroup
14556     \@glstrbookindex@atendgroup

```

End multicol environment.

```

14557     \expandafter\end\expandafter{\glstrbookindexmulticolseenv}%
14558   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

14559   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

14560   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

14561     \@glstrbookindex@between{##1}%

```

Update separators.

```

14562     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14563     \let\@glstrbookindex@subsep\glstrbookindexparentschildsep
14564     \let\@glstrbookindex@subbetween\@gobble
14565     \let\@glstrbookindex@subsubbetween\@gobble
14566     \edef\@glstrbookindex@between{%

```

```

14567     \noexpand\glxstrbookindexbetween{##1}%
14568 }%
14569 \edef\@glxstr@bookindex@atendgroup{%
14570     \noexpand\glxstrbookindexatendgroup{##1}%
14571 }%
14572 \let\@glxstr@bookindex@subatendgroup\relax
14573 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14574 \glstreeitem
14575 \glstryitem{##1}%
14576 \glstarget{##1}{\glxstrbookindexname{##1}}%
14577 \glxstrbookindexprelocation{##1}##2%
14578 }%
14579 \renewcommand{\subglossentry}[3]{%
14580 \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14581 \glstreeitem
14582 \or

```

Level 1.

```

14583 \@glxstr@bookindex@sep
14584 \@glxstr@bookindex@subbetween{##2}%
14585 \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14586 \let\@glxstr@bookindex@subsubbetween\@gobble
14587 \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14588 \edef\@glxstr@bookindex@subbetween{%
14589     \noexpand\glxstrbookindexsubbetween{##2}%
14590 }%
14591 \edef\@glxstr@bookindex@atsubendgroup{%
14592     \noexpand\glxstrbookindexatsubendgroup{##1}%
14593 }%

```

Start sub-item.

```

14594 \glstreesubitem
14595 \glssubentryitem{##2}%
14596 \else

```

All other levels.

```

14597 \@glxstr@bookindex@subsep
14598 \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14599 \let\@glxstr@bookindex@subsep\relax
14600 \edef\@glxstr@bookindex@subsubbetween{%
14601     \noexpand\glxstrbookindexsubsubbetween{##2}%
14602 }%
14603 \edef\@glxstr@bookindex@atsubsubendgroup{%
14604     \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14605 }%

```

Start sub-sub-item.

```
14606      \glstreesubsubitem
14607      \fi
```

Format entry.

```
14608      \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14609      \glxtrbookindexsubprelocation{##2}##3%
14610  }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14611  \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14612  \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14613      \@glxtr@bookindex@subsubatendgroup
14614      \@glxtr@bookindex@subatendgroup
14615      \@glxtr@bookindex@atendgroup
14616      \@glxtr@bookindexgroupskip
```

Update separators.

```
14617      \let\@glxtr@bookindexgroupskip\glxtrbookindexgroupskip
14618      \let\@glxtr@bookindex@between\@gobble
14619      \let\@glxtr@bookindex@atendgroup\relax
14620      \let\@glxtr@bookindex@subatendgroup\relax
14621      \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14622      \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14623      \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14624      \glxtrbookindexformatheader{\thisgrptitle}%
14625      \nopagebreak\indexspace\nopagebreak\@afterheading
14626  }%
14627 }
```

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.

```
14628 \newcommand{\glxtrbookindexthepage}{%
14629 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14630 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14631 \newcommand*{\glxtrbookindexmarkentry}[1]{%
14632   \protected@write\@auxout
14633   {\let\glxtrbookindexthepage\relax}%
14634   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14635 }
```

etbookindexmark

```
14636 \newcommand*{\glxtr@setbookindexmark}[2]{%
14637   \ifcsundef{glxtr@idxfirstmark@#1}%
14638   {\csgdef{glxtr@idxfirstmark@#1}{#2}}%
14639   {}%
14640   \csgdef{glxtr@idxlastmark@#1}{#2}%
14641 }
```

indexfirstmarkfmt

```
14642 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
14643   \glsentryname{#1}%
14644 }
```

indexfirstmark

```
14645 \newcommand*{\glxtrbookindexfirstmark}{%
14646   \letcs{\glxtr@label}{glxtr@idxfirstmark@\glxtrbookindexthepage}%
14647   \ifdef\glxtr@label
14648   {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%
14649   {}%
14650 }
```

indexlastmarkfmt

```
14651 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
14652   \glsentryname{#1}%
14653 }
```

okindexlastmark

```
14654 \newcommand*{\glxtrbookindexlastmark}{%
14655   \letcs{\glxtr@label}{glxtr@idxlastmark@\glxtrbookindexthepage}%
14656   \ifdef\glxtr@label
14657   {\glxtrbookindexlastmarkfmt{\glxtr@label}}%
14658   {}%
14659 }
```

Glossary

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

First use flag A conditional that determines whether or not the entry has been used according to the rules of **first use**.

First use text The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.

Change History

0.1 (2015-11-22)		\@Glsymbol@: added redefinition	74
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	74
\Glsfmtshort: new	324	\@Glstext@: added redefinition	70
\glsfmtshort: new	323	\@Glsuseri@: added redefinition	75
\Glsfmtshortpl: new	324	\@Glsuserii@: added redefinition	75
\glsfmtshortpl: new	323	\@Glsuseriii@: added redefinition	75
short: switched inline full form to short		\@Glsuseriv@: added redefinition	75
(long)	226	\@Glsuserv@: added redefinition	76
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	76
\@ACRlong: added redefinition	79	\@acrlong: added redefinition	79
\@ACRlongpl: added redefinition	80	\@acrlongpl: added redefinition	80
\@ACRshort: added redefinition	77	\@acrshort: added redefinition	76
\@ACRshortpl: added redefinition	78	\@acrshortpl: added redefinition	77
\@Acrlong: added redefinition	79	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	80	argument	62
\@Acrshort: added redefinition	77	\@glsdescplural@: added redefinition	73
\@Acrshortpl: added redefinition	78	\@glsfirst@: added redefinition	70
\@GLSdesc@: added redefinition	73	\@glsfirstplural@: added redefinition	71
\@GLSdescplural@: added redefinition	73	\@glsplural@: added redefinition	71
\@GLSfirst@: added redefinition	70	\@glssymbolplural@: added	
\@GLSfirstplural@: added redefinition	72	redefinition	74
\@GLSname@: added redefinition	72	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	71	new	133
\@GLSSymbol@: added redefinition	74	\@glsxtr@field@linkdefs: new	69
\@GLSSymbolplural@: added		\@glsxtr@insertdots: new	194
redefinition	74	\@print@glossary: added redefinition	129
\@GLStext@: added redefinition	69	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	75	\abbrvdefaultfont	200
\@GLSuserii@: added redefinition	75	\glsaccessdesc: new	158
\@GLSuseriii@: added redefinition	75	\glsaccessdescplural: new	159
\@GLSuseriv@: added redefinition	76	\glsaccessfirst: new	156
\@GLSuserv@: added redefinition	76	\glsaccessfirstplural: new	156
\@GLSuservi@: added redefinition	76	\Glsaccesslong: new	161
\@Glsdesc@: added redefinition	73	\glsaccesslong: new	160
\@Glsdescplural@: added redefinition	73	\glsaccessname: new	154
\@Glsfirst@: added redefinition	70	\glsaccessplural: new	155
\@Glsfirstplural@: added redefinition	72	\Glsaccessshort: new	159
\@Glsname@: added redefinition	72	\glsaccessshort: new	159
\@Glsplural@: added redefinition	71	\Glsaccessshortpl: new	160

\glsaccesssshortpl: new	160	\cGLSpl: new	106
\glsaccessssymbol: new	157	\cGLSpl@: new	106
\glsaccessssymbolplural: new	157	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	155	new	101
\glstentryfmt: added check for short ..	61	\cGLS: new	106
\glslongpltok: new	194	\cGLSformat: new	106
\glsshortpltok: new	194	\cGLSpl: new	106
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	106
name in \setkeys	196	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	16
for plural	191	\glsenableentrycount: new	101
\GLSxtrlongpl: new	211	\glsfirstabbrvdefaultfont: new ..	200
\Glsxtrlongpl: new	210	\glsfirstlongdefaultfont: new ...	200
\glsxtrlongpl: new	209	\Glsfmtfirst: new	326
\glsxtrNoGlossaryWarning: new	21	\glsfmtfirst: new	326
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	327
new	191	\glsfmtfirstpl: new	326
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	326
new	191	\glsfmtplural: new	325
\glsxtrpostlinkendsentence: new ..	190	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	209	\Glsxtrtitleshort	324
\Glsxtrshortpl: new	208	renamed from \glstentryfmtshort ..	324
\glsxtrshortpl: new	207	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	323
\glslabeltok	221	renamed from \glstentryfmtshort ..	323
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	219	\Glsxtrtitleshortpl	324
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	324
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	323
\Glsxtrshort	324	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	323
\glsxtrshort	323	\Glsfmttext: new	325
\Glsfmtshortpl: changed to use		\glsfmttext: new	325
\glsxtrshortpl	324	\glshasattribute: new	169
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	168
\glsxtrshortpl	323	\glsxtremsuffix: new	263
\glsxtrifemptyglossary: new	28	\GlsXtrEnableEntryCounting: new ..	101
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	103
argument	172	\glsxtrscfont: new	234
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	234
argument	172	\glsxtrsmfont: new	248
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	249
default type to \acronymtype	114	short-em: new	270
\newterm: fixed name argument	171	short-em-desc: new	271
0.5 (2015-12-07)		short-em-footnote: new	280
\cGLS: new	106	short-em-long: new	266
\cGLS@: new	106	short-em-long-desc: new	267

short-em-postfootnote: new	282	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	245	attribute	314
short-sc-postfootnote: new	246	\Glsxtrheadtext: now uses headuc	
short-sm: new	252	attribute	317
short-sm-desc: new	254	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	259	attribute	316
short-sm-long: new	250	short-em-footnote: switch off regular	
short-sm-long-desc: new	252	attribute if set	281
short-sm-postfootnote: new	260	short-long: switch off regular attribute	
long-noshort-em: new	273	if set	220
long-noshort-em-desc: new	277	short-long-desc: switch off regular	
long-noshort-sm: new	256	attribute if set	221
long-noshort-sm-desc: new	257	short-sc-footnote: switch off regular	
long-short-em: new	263	attribute if set	245
long-short-em-desc: new	264	short-sm-footnote: switch off regular	
long-short-sm: new	249	attribute if set	259
long-short-sm-desc: new	250	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	218
\Glsaccessstext: new	155	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	219
\@gls@setup@default@short@access:		long-short-sc-desc: switch off regular	
removed \ifglxtruseuchead ...	314	attribute if set	236
\@glxtr@doaccsupp: new	21	footnote: switch off regular attribute if	
\Glsaccessdesc: new	158	set	223
\Glsaccessdescplural: new	159	postfootnote: switch off regular	
\Glsaccessfirst: new	156	attribute if set	224
\Glsaccessfirstplural: new	157	0.5.2 (2015-12-08)	
\Glsaccessname: new	154	\@GLSdesc@: added accessibility support	73
\Glsaccessplural: new	155	\@GLSdescplural@: added accessibility	
\Glsaccesssymbol: new	157	support	73
\Glsaccesssymbolplural: new	158	\@GLSfirst@: added accessibility	
\Glsxtrheadfirst: now uses headuc		support	70
attribute	318	\@GLSfirstplural@: added accessibility	
\glxtrheadfirst: now uses headuc		support	72
attribute	318	\@GLSname@: added accessibility support	72
\Glsxtrheadfirstplural: now uses		\@GLSplural@: added accessibility	
headuc attribute	319	support	71
\glxtrheadfirstplural: now uses		\@GLSsymbol@: added accessibility	
headuc attribute	319	support	74
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute	318	accessibility support	74
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	69
attribute	317	\@GLSdesc@: added accessibility support	73
\Glsxtrheadshort: now uses headuc		\@GLSdescplural@: added accessibility	
attribute	315	support	73
\glxtrheadshort: now uses headuc		\@GLSfirst@: added accessibility	
attribute	314	support	70
\Glsxtrheadshortpl: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	315	support	72

\@Glsname@: add accessibility support ..	72	\glsdohypertarget: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	154
\@Glsplural@: added accessibility support	71	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here	61
\@Glsymbol@: added accessibility support	74	\GlsXtrEnableInitialTagging: new	186
\@Glsymbolplural@: added accessibility support	74	\glsxtrfieldtitlecase: new	173
\@Glstext@: added accessibility support	70	\GlsXtrFormatLocationList: new ...	59
\@gls@setup@default@short@access: removed \glsxtrabbrvfmt	211	\glsxtrnewabbrevpresetkeyhook: new	198
\@glsdesc@: added accessibility support	73	\glsxtrtagfont: new	188
\@glsdescplural@: added accessibility support	73	\KV@printgloss@nonumberlist: added	61
\@glsfirst@: added accessibility support	70	\mfu@checkword@do: added	187
\@glsfirstplural@: added accessibility support	71	\setabbreviationstyle: added check for post-definition style switch	214
\@Glsname@: added accessibility support	72	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	71	\@glsxtr@autoindex@at: new	184
\@Glsymbol@: added accessibility support	74	\@glsxtr@autoindex@encap: new ...	184
\@Glsymbolplural@: added accessibility support	74	\@glsxtr@autoindex@esc: new	185
\@Glstext@: added accessibility support	69	\@glsxtr@autoindex@level: new ...	184
\@glsxtr@activate@initialtagging: new	188	\@glsxtr@autoindex@setname: new .	182
\@glsxtr@do@titlecaps@warn: new .	188	\@glsxtr@doabbreviationsdef: new .	17
\@glsxtr@tag: new	188	\glsdescwidth: added	58
\glossaryentrynumbers: added	59	\glsdohypertarget: removed \GlsXtrNoGlsWarningNoAutoMakeMain	132
\Glossentrydesc: added	186	\glsPagelistwidth: added	58
\Glossentryname: added	178	\glsxtrdoautoindexname: new	182
\Glossentrysymbol: added	186	\glsxtrpostnamehook: new	179
\glossentrysymbol: added	186	\if@glsxtr@format@override: new .	181
\GLSaccessdesc: new	158, 166	\ProvidesGlossariesExtraLang: new	330
\GLSaccessdescplural: new ...	159, 166	\RequireGlossariesExtraLang: new	329
\GLSaccessfirst: new	156, 165	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	157, 165	\@newglossaryentry@defunitcounters: new	107
\GLSaccesslong: new	161, 167	\@GLSxtr@p@acrlong@: new	92
\GLSaccesslongpl: new	161, 167	\@GLSxtr@p@acrlongpl@: new	93
\Glsaccesslongpl: new	161	\@GLSxtr@p@acrshort@: new	92
\glsaccesslongpl: new	161	\@GLSxtr@p@acrshortpl@: new	92
\GLSaccessname: new	154, 164	\@GLSxtr@p@long@: new	91
\GLSaccessplural: new	156, 165	\@GLSxtr@p@longpl@: new	92
\GLSaccessshort: new	160, 167	\@GLSxtr@p@plural@: new	90
\GLSaccessshortpl: new	160, 167	\@GLSxtr@p@short@: new	91
\GLSaccesssymbol: new	157, 166	\@GLSxtr@p@shortpl@: new	91
\GLSaccesssymbolplural: new .	158, 166	\@GLSxtr@p@text@: new	90
\GLSaccessstext: new	155, 165	\@GlsXtrEnableOnTheFly: new	54
		\@Glsxtr: new	55
		\@Glsxtr@p@acrlong@: new	92

\@Glsxtr@p@acrlongpl@: new	92	\glsdonohyperlink: added	89
\@Glsxtr@p@acrshort@: new	92	\glsenableentryunitcount: new	109
\@Glsxtr@p@acrshortpl@: new	92	\glsasattribute: added check for	
\@Glsxtr@p@long@: new	91	entry's existence	169
\@Glsxtr@p@longpl@: new	92	\glsifattribute: added check for	
\@Glsxtr@p@plural@: new	90	entry's existence	169
\@Glsxtr@p@short@: new	90	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	91	check	189
\@Glsxtr@p@text@: new	90	\Glsxtr: new	55
\@Glsxtrpl: new	56	\glsxtr: new	55
\@alt@glshyp@opt: new	86	\glsxtrcat: new	55
\@glscalt@hyp@opt: new	86	\glsxtrdowrglossaryhook: new	86
\@glscalt@hyp@opt@char: new	86	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	86	new	112
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	54
new	108	\Glsxtrpl: new	56
\@glsc@local@increment@currunitcount:		\glsxtrpl: new	55
new	108	\glsxtrpostlocalreset: new	100
\@glsc@setdefault@glslink@opts:		\glsxtrpostlocalunset: new	100
new	83	\glsxtrpostreset: new	100
\@glsxtr: new	55	\glsxtrpostunset: new	98
\@glsxtr@addunitcounter: new	107	\glsxtrprotectlinks: new	89
\@glsxtr@currunitcount: new	109	\GlsXtrSetAltModifier: new	86
\@glsxtr@ifunitcounter: new	108	\GlsXtrSetDefaultGlsOpts: new	85
\@glsxtr@p@acrlong@: new	92	\glsxtrstarflywarn: new	54
\@glsxtr@p@acrlongpl@: new	92	\GlsXtrWarning: new	56
\@glsxtr@p@acrshort@: new	92	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new	92	disables \setacronymstyle	114
\@glsxtr@p@long@: new	91	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new	92	\@glsxtr@autoindexcrossrefs: new	16
\@glsxtr@p@plural@: new	90	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new	90	new	123
\@glsxtr@p@shortpl@: new	91	\@glsxtr@idx@entrynumberlist: new	125
\@glsxtr@p@text@: new	90	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new	109	new	123
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new	113	new	124
\@glsxtr@unitcountlist: new	107	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new	55	new	124
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new	116
empty see value if not set and added		\makeglossaries: new	116
'see' to field key map	45	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	54	\glsxtrdiscardperiod: added check	
\cGlsformat: added	107	for first use	191
\cGlsformat: added	106	short-desc: fixed typo in	
\cGlsplformat: added	107	\glsxtrinlinefullformat and	
\cGlsplformat: added	107	added missing second argument	228
\glsdisablehyper: added	88	1.02 (2016-04-25)	
\glsdohyperlink: added	87	\@glsxtr@current@style: new	57

\Glsfmtfull: new	328	\@GLSdesc@: set abbreviation and regular format	73
\glsfmtfull: new	328	\@GLSdescplural@: set abbreviation and regular format	73
\Glsfmtfullpl: new	329	\@GLSfirst@: set abbreviation format ..	70
\glsfmtfullpl: new	329	\@GLSfirstplural@: set abbreviation and regular format	72
\Glsfmtlong: new	327	\@GLSname@: set abbreviation and regular format	72
\glsfmtlong: new	327	\@GLSplural@: set abbreviation and regular format	71
\Glsfmtlongpl: new	328	\@GLSsymbol@: set regular format	74
\glsfmtlongpl: new	328	\@GLSsymbolplural@: set regular format	74
\Glsxtrheadfull: new	322	\@GLStext@: set abbreviation and regular format	69
\glsxtrheadfull: new	321	\@GLSuseri@: set regular format	75
\Glsxtrheadfullpl: new	323	\@GLSuserii@: set regular format	75
\glsxtrheadfullpl: new	322	\@GLSuseriii@: set regular format	75
\Glsxtrheadlong: new	320	\@GLSuseriv@: set regular format	76
\glsxtrheadlong: new	320	\@GLSuseriv@: set regular format	76
\Glsxtrheadlongpl: new	321	\@GLSdesc@: set abbreviation and regular format	73
\glsxtrheadlongpl: new	320	\@GLSdescplural@: set abbreviation and regular format	73
\Glsxtrtitlefull: new	322	\@GLSfirst@: set abbreviation and regular format	70
\glsxtrtitlefull: new	322	\@GLSfirstplural@: set abbreviation and regular format	72
\Glsxtrtitlefullpl: new	323	\@GLSname@: set abbreviation and regular format	72
\glsxtrtitlefullpl: new	322	\@GLSplural@: set abbreviation and regular format	71
\Glsxtrtitlelong: new	321	\@GLSsymbol@: set regular format	74
\glsxtrtitlelong: new	320	\@GLSsymbolplural@: set regular format	74
\Glsxtrtitlelongpl: new	321	\@GLStext@: set abbreviation and regular format	70
\glsxtrtitlelongpl: new	320	\@GLSuseri@: set regular format	75
\ifglsxtrinsertinside: new	217	\@GLSuserii@: set regular format	75
postfootnote: added redef of \glsxtrsetupfulldefs	225	\@GLSuseriii@: set regular format	75
stylemods: new	22	\@GLSuseriv@: set regular format	76
1.03 (2016-04-27)		\@GLSuseriv@: set regular format	76
\@GLSfirstplural@: bug fix: misspelt cs name	72	\@GLSdesc@: set abbreviation and regular format	73
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	71	\@GLSdescplural@: set abbreviation and regular format	73
\@GLSfirstplural@: bug fix: misspelt cs name	72	\@GLSfirst@: set abbreviation and regular format	70
\@GLsplural@: fixed bug \@GLsplural@ should be redefined not \@GLsplural	71	\@GLSfirstplural@: set abbreviation and regular format	72
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	71	\@GLSname@: set abbreviation and regular format	72
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl ..	320	\@GLSplural@: set abbreviation and regular format	71
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl	315	\@GLSsymbol@: set regular format	74
1.04 (2015-04-30)		\@GLSsymbolplural@: set regular format	74
short-em-footnote: renamed from “footnote-em”	280	\@GLStext@: set abbreviation and regular format	70
1.04 (2016-05-02)		\@GLSuseri@: set regular format	75
\@glsxtrpostloctag: new	60	\@GLSuserii@: set regular format	75
		\@GLSuseriii@: set regular format	75
		\@GLSuseriv@: set regular format	75
		\@GLSuseriv@: set regular format	76
		\@GLSuseriv@: set regular format	76
		\@GLSdesc@: set abbreviation and regular format	73
		\@GLSdescplural@: set abbreviation and regular format	73

\@glsfirst@: set abbreviation and regular format	70	\glsxtrregularfont: new	61
\@glsfirstplural@: set abbreviation and regular format	71	\glsxtruserfield: new	284
\@glsname@: set abbreviation and regular format	72	\glsxtruserparen: new	284
\@glsplural@: set abbreviation and regular format	71	\glsxtrusersuffix: new	285
\@glsymbol@: set regular format	74	\GlsXtrWarnDeprecatedAbbrStyle:	
\@glsymbolplural@: set regular format	74	new	217
\@glstext@: set abbreviation and regular format	69	short-em-long-em: new	268
\@glsxtr@deprecated@abbrstyle:		short-em-long-em-desc: new	269
new	216	short-em-nolong: new	271
\@glsxtr@do@style: new	22	short-em-nolong-desc: new	272
\@glsxtr@doloctag: new	61	short-em-postfootnote: renamed from “postfootnote-em”	282
\@glsxtr@idx@entrynumberlist:		short-footnote: new	224
switched from \let to \newcommand	125	short-long-user: new	291
\@glsxtr@pagetag: new	60	short-long-user-desc: new	293
\@glsxtr@pagetag: new	60	short-nolong: new	227
\@glsxtr@preloctag: new	61	short-nolong-desc: new	229
\@glsxtr@postloctag: new	60	short-postfootnote: new	226
\@glsxtr@preloctag: new	60	short-sc-footnote: renamed from “footnote-sc”	245
\glossentrydesc: added glossdescfont attribute check	174	short-sc-nolong: new	239
\Glossentryname: added glossnamefont attribute check	178	short-sc-nolong-desc: new	241
\glossentryname: added glossnamefont attribute check	175	short-sc-postfootnote: renamed from “postfootnote-sc”	246
moved post name hook inside condition	177	short-sm-footnote: renamed from “footnote-sm”	259
\glsabbrvmfont: new	262	short-sm-nolong: new	253
\glsabbrvuserfont: new	284	short-sm-nolong-desc: new	255
\glsfirstabbrvmfont: new	262	short-sm-postfootnote: renamed from “postfootnote-sm”	260
\glsfirstabbrvuserfont: new	284	\letabbreviationstyle: new	216
\glsfirstlongemfont: new	263	\newabbreviationstyle: bug fix:	
\glsfirstlonguserfont: new	285	corrected test for existence	215
\glsifnotregularcategory: new ...	170	long-em-noshort-em: new	275
\glslongdefaultfont: new	200	long-em-noshort-em-desc: new	278
\glslongemfont: new	263	long-em-short-em: new	264
\glslongfont: new	200	long-em-short-em-desc: new	266
\glslonguserfont: new	285	long-noshort: new	234
\glsxtrassignfieldfont: new	69	long-noshort-desc: new	233
\GlsXtrEnablePreLocationTag: new .	59	long-noshort-em: renamed from “long-em”	273
\glsxtrfirstscfont: new	234	long-noshort-em-desc: renamed from “long-desc-em”	277
\glsxtrfirstsmfont: new	248	long-noshort-sc: renamed from “long-sc”	241
\glsxtrlongshortdescsort: new ...	219	long-noshort-sc-desc: renamed from “long-desc-sc”	243
\glsxtrpostnamehook: added category check	179	long-noshort-sm: renamed from “long-sm”	256

long-noshort-sm-desc: renamed from	General: disabled docdef key at the start
\long-desc-sm 257	of the document 27
long-short-user: new 285	docdef option changed to choice 14
long-short-user-desc: new 291	\glstr@usesee: new 46
\renewabbreviationstyle: new 216	\glstrusesee: new 45
style: new 22	\glstruseseeformat: new 46
1.05 (2016-06-10)	\if@glstrdocdefrestricted: new .. 15
\eglssetwidest: new 385	1.07 (2016-08-15)
\glFindWidestAnyName: new 387	@@glstrp: new 93
\glFindWidestAnyNameLocation:	\GLSfirst@: added check for
new 393	nohyperfirst attribute 71
\glFindWidestAnyNameSymbol: new 390	\GLSfirstplural@: added check for
\glFindWidestAnyNameSymbolLocation:	nohyperfirst attribute 72
new 392	\GLSxtrp: new 94
\glFindWidestLevelTwo: new 389	\@Glsfirst@: added check for
\glFindWidestUsedAnyName: new .. 387	nohyperfirst attribute 70
\glFindWidestUsedAnyNameLocation:	\@Glsfirstplural@: added check for
new 392	nohyperfirst attribute 72
\glFindWidestUsedAnyNameSymbol:	\@Glsxtrp: new 93
new 390	\@glspreglossaryhook: added
\glFindWidestUsedAnyNameSymbolLocation:	\glossxtrsetpopts 189
new 391	\@glfirst@: added check for
\glFindWidestUsedLevelTwo: new . 388	nohyperfirst attribute 70
\glFindWidestUsedTopLevelName:	\@glfirstplural@: added check for
new 386	nohyperfirst attribute 71
\glfirstlongfootnotefont: new .. 222	\@glxtrinmark: new 312
\glsgetwidestname: new 386	\@glxtrnotinmark: new 312
\glsgetwidestsubname: new 386	\@glxtrp: new 93
\glslongfootnotefont: new 222	\@glxtrp@opt: new 93
\glxtrAltTreeIndent: new 384	\glossxtrsetpopts: new 93
\glxtralttreeInit: new 385	\glsp: new 95
\glxtrAltTreePar: new 384	\glsp: new 95
\glxtrAltTreeSetHangIndent: new 394	\glxtr@entry@p: new 94
\glxtrAltTreeSetSubHangIndent:	\glxtrabbrvfootnote: new 222
new 394	\glxtrchecknohyperfirst: new 70
\glxtralttreeSubSymbolDescLocation:	\glxtrfieldtitlecasecs: new 173
new 384	\glxtrifinmark: new 311
\glxtralttreeSymbolDescLocation:	\GLSxtrp: new 97
new 384	\Glsxtrp: new 96
\glxtrComputeTreeIndent: new ... 393	\glxtrp: new 94
\glxtrComputeTreeSubIndent: new 394	\glxtrsetpopts: new 93
\glxtrtreetopindent: new 385	short-long-desc: added text key 221
short-em-long: fixed incorrect font used	fixed misspelling of \glabbrvfont in
by long form 267	plural key 221
\xglsetwidest: new 385	long-short-desc: added missing text
1.06 (2016-06-18)	key 219
\@glsoifexistsorwarn: new 15	fixed misspelling of \glabbrvfont . 219
\@glxtr@docdefval: new 15	footnote: changed first forms to use
\@glxtr@usesee: new 46	\glfirstlongfootnotefont ... 222

postfootnote: removed \footnote		\@printglossary: redefined to save	
from first keys	224	options	122
switched from \glsfirstlongfont to		\glsxtr@makeglossaries: new	122
\glsfirstlongfootnotefont ...	225	1.10 (2016-12-17)	
\RestoreAcronyms: modified		\@GLSp1@: fixed bug caused by typo in	
\@gls@link@checkfirsthyper to		command name	63
set \glsxtrifwasfirstuse	115	1.11 (2017-01-19)	
1.08 (2016-12-13)		\@glsxtr@do@redef@forglsentries:	
\@glsxtr@record: new	8	new	6
\@GLS@: added \@glsxtr@record	63	\@glsxtr@noidx@do: new	143
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@redef@forglsentries: new .	6
\@Gls@: added \@glsxtr@record	63	\@glsxtr@shortcutsval: new	20
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@unsrt@getgrouptitle: new	142
\@Gls@: added \@glsxtr@record	62	\@print@noidx@glossary: added	
\@gls@link@: added		redefinition	126
\@glsxtr@record	64	\glsxtr@addloclistfield: added	
\@gls@field@link: added		group key	13
\@glsxtr@record	62	added location key	12
\@gls@saveentrycounter: new	27	\glsxtr@fields: new	135
\@glsdisp: added \@glsxtr@record ..	63	\glsxtr@linkprefix: new	135
\@GLSp1@: added \@glsxtr@record ...	62	\glsxtr@org@newignoredglossary:	
\@glsxtr@dorecord: new	10	new	40
\@glsxtr@err@undefaction: new	6	\glsxtr@s@newignoredglossary: new	41
\@glsxtr@record: new	7	\glsxtr@shortcutsval: new	135
\@glsxtr@warn@onexistsordo: new ...	6	\glsxtr@texencoding: new	135
\@glsxtr@warn@undefaction: new	6	\glsxtr@writefields: new	135
\@print@unsrt@glossary: new	140	\GlsXtrLoadResources: new	135
record: added record package option ...	13	\glsxtrpageref: new	38
\glsadd: added \@glsxtr@record	68	\glsxtrresourcefile: changed	
\glsdoifexists: now defines		extension to .gls tex	134
\glslabel	44	\newignoredglossary: added starred	
\glsxtr@do@wrglossary: new	27	version	40
\glsxtr@addloclistfield: new	12	1.12 (2017-02-03)	
\glsxtr@indexonly@saveentrycounter:		\@glsxtr@recordcounter: new	11
new	12	\@gls@preglossaryhook: check for	
\glsxtr@record: new	137	definition	188
\glsxtr@resource: new	135	\@glsxtr@counterrecordhook: new .	137
\glsxtr@saveentrycounter: new	27	\@glsxtr@display@loc: new	127
\glsxtr@setup@record: new	12	\@glsxtr@docounterrecord: new ...	137
\glsxtrassignfieldfont: added check		\@glsxtr@longnewglossaryentry:	
for existence	69	new	40
\glsxtrresourcefile: new	134	\@glsxtr@noop@recordcounter: new .	11
\printunsrtglossaries: new	140	\@glsxtr@op@recordcounter: new ...	11
\printunsrtglossary: new	139	\@glsxtr@provide@storagekey: new .	28
1.09 (2016-12-16)		\@glsxtr@s@longnewglossaryentry:	
\@glsxtr@gettype: new	123	new	39
\@glsxtr@mixed@assign@sortkey:		\@glsxtrentryfmt: new	31
new	123	\@glsxtrindexaliased: new	84
		\@glsxtrsetaliasnoindex: new	84

\@newglossaryentryposthook: added check for alias key	49	\glxtrindexaliased: new	85
\@no@glxtrindexaliased: new	84	\GlsXtrLetField: new	34
\@printunsrtglossary: new	139	\GlsXtrLetFieldToField: new	34
General: added target key to printgloss family	122	\GlsXtrLoadResources: removed restriction on only one per document	135
\apptoglossarypreamble: new	38	\glxtrlocrangefmt: new	128
\csGlsXtrLetField: new	34	\glxtrpostlongdescription: new ..	40
\eGlsXtrSetField: new	35	\glxtrprovidestoragekey: new	28
\gGlsXtrSetField: new	35	\GlsXtrRecordCounter: new	137
\glsdohyperlink: added check for alias field	88	\glxtrresourcecount: new	135
\glsnoidxdisplayloc: added redefinition	127	\glxtrresourcefile: added catcode change for @	134
\glissettoctitle: added patch	41	\glxtrsetaliasnoindex: new	84
\glxtr@counterrecord: new	137	\GlsXtrSetField: new	34
\glxtr@langtag: new	135	\glxtrsetfieldifexists: new	34
\glxtr@newabbreviation: new	196	\glxtrunsrtdo: new	143
\glxtr@org@newignoredglossary: Added check for existence	40	\glxtrusefield: new	34
\glxtr@pluralsuffixes: new	135	\glxtrusefield: new	34
\glxtr@provideignoredglossary: new	42	short-postlong-user: new	288
\glxtr@s@newignoredglossary: Added check for existence	41	short-postlong-user-desc: new ...	290
\glxtr@s@provideignoredglossary: new	43	\longnewglossaryentry: added starred version	39
\glxtrabbrvpluralsuffix: new ...	200	long-postshort-user: new	286
\glxtralias: new	49	long-postshort-user-desc: new ...	288
\glxtrcopytoglossary: new	43	postdot: new	16
\glxtrdeffield: new	34	\pretoglossarypreamble: new	38
\glxtrdisplayendloc: new	128	\print@noop@unsrtglossaryunit: new	142
\glxtrdisplayendlochook: new ...	128	\print@op@unsrtglossaryunit: new	142
\glxtrdisplaysingleloc: new	127	\printunsrtglossary: added starred form	139
\glxtrdisplaystartloc: new	127	\printunsrtglossaryhandler: new .	141
\glxtrredefield: new	34	\printunsrtglossaryunit: new	12
\glxtrentryfmt: new	30	\printunsrtglossaryunitsetup: new	142
\glxtrfieldddolistloop: new	31	\provideignoredglossary: new	42
\glxtrfieldforlistloop: new	32	\s@glxtr@provide@storagekey: new	29
\glxtrfielddininlist: new	32	\s@printunsrtglossary: new	140
\glxtrfieldlistadd: new	31	\xGlsXtrSetField: new	35
\glxtrfieldlistadd: new	31	1.13 (2017-02-07)	
\glxtrfieldlistgadd: new	31	\@glsdisp: removed	
\glxtrfieldlistxadd: new	31	\@glxtr@org@glsdisp	63
\glxtrfieldxifinlist: new	32	\glxtrsetaliasnoindex: switched to \providecommand	84
\glxtrfmt: new	29	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	29	\@gls@link: added redefinition	65
\GlsXtrFmtField: new	29	\@gls@noidx@getgrouptitle: new ..	125
\glxtrifkeydefined: new	28	\@gls@removespaces: new	128
		\@glxtr@do@automake@err: new ...	137
		\@glxtr@org@gloautosee: new	25

\@glxtr@record: added third arg	7	postfootnote: fixed spelling of	
\@glxtr@recordsee: new	11	\glsabbrvfont	224
General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	68	\@glo@autosee: added redefinition	26
added \glsadd option thevalue	68	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	88	bug	125
\glsenableentrycount: fixed		\@glxtr@addunusedxrefs: added	
assignment of \@cGls@	102	check for seealso field	50
\glsenableentryunitcount: fixed		\@glxtr@checkgroup: use \csuse	
assignment of \@cGls@	111	instead of \csname	143
\glsnavigation: new	126	\@glxtr@dorecordnodefer: new	11
\glxtr@org@getgrouptitle: new ..	125	\@print@unsrt@glossary: corrected	
\glxtr@recordsee: new	7	misspelt command	140
\glxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	136	new	141
\glxtrdisplayendloc: added check		record: added check for	
for empty format	128	\@gls@setupsort@none	14
\glxtrgetgrouptitle: new	125	\gls@checkseeallowed: added	
\glxtrinitwrgloss: new	64	redefinition	26
\glxtrlocationhyperlink: new ...	129	\glxtr@writefields: added	
\glxtrsetgrouptitle: new	125	\providecommand lines	135
\glxtrsupphypernumber: new	129	\glxtrautoindex: new	182
\ifglxtrwrglossbefore: new	64	\glxtrautoindexassignsort: new .	183
1.15 (2017-05-10)		\glxtrautoindexentry: new	183
\@glxtr@dorecord: corrected		\glxtrindexseealso: new	47
premature expansion of \@glslocref	10	\glxtrseealso: new	49
short-em-long-em: fixed spelling of		\glxtrseelist: new	47
\glsabbrvfont	268	\glxtruseseealso: new	46
short-long: fixed spelling of		\glxtruseseealsoformat: new	46
\glsabbrvfont	220	\seealso: new	47
short-long-user: fixed spelling of		autoseeindex: new	16
\glsabbrvfont	292	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@gls@setup@default@short@access:	
\glsabbrvfont	289	removed some inconsistencies in the	
short-postlong-user-desc: fixed		abbreviation styles	218
spelling of \glsabbrvfont	290	\@glxtr@mark@wordseps: new	195
long-em-short-em: fixed spelling of		\@glxtr@markwordseps: new	195
\glsabbrvfont	265	\@glxtr@noidx@displaynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	286	\glxtrundeftag	123
long-postshort-user-desc: fixed		\@glxtr@noidx@entrynumberlist:	
spelling of \glsabbrvfont	288	replace hard-coded ?? with	
long-short: fixed spelling of		\glxtrundeftag	124
\glsabbrvfont	218	\@glxtr@noidx@numberlistloop:	
long-short-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	285	\glxtrundeftag	124
footnote: fixed spelling of		\@glxtrtrifhyphenstart: new	294
\glsabbrvfont	223	\glsabbrvhyphenfont: new	294
		\glsabbrvonlyfont: new	307

<code>\glsabbrvscfont:new</code>	234	<code>short-hyphen-postlong-hyphen-desc:</code>	
<code>\glsabbrvsmfont:new</code>	248	<code>new</code>	307
<code>\glsabbrvuserfont:initialised to</code>		<code>short-long-user-desc:corrected first</code>	
<code>default font</code>	284	<code>forms</code>	293
<code>\glsfirstabbrvhyphenfont:new</code> ...	294	<code>short-nolong-desc-noreg:new</code> ...	229
<code>\glsfirstabbrvonlyfont:new</code>	307	<code>short-nolong-noreg:new</code>	228
<code>\glsfirstabbrvscfont:new</code>	234	<code>long-em-noshort-em-desc-noreg:</code>	
<code>\glsfirstabbrvsmfont:new</code>	249	<code>new</code>	280
<code>\glsfirstlonghyphenfont:new</code> ...	294	<code>long-em-noshort-em-noreg:new</code> ...	276
<code>\glsfirstlongonlyfont:new</code>	308	<code>long-hyphen-noshort-desc-noreg:</code>	
<code>\glslonghyphenfont:new</code>	294	<code>new</code>	296
<code>\glslongonlyfont:new</code>	307	<code>long-hyphen-postshort-hyphen:new</code>	300
<code>\glslonguserfont:initialised to default</code>		<code>long-hyphen-postshort-hyphen-desc:</code>	
<code>font</code>	285	<code>new</code>	301
<code>\glxtr@newabbreviation:added</code>		<code>long-hyphen-short-hyphen:new</code> ...	295
<code>\glxtrorgshort and</code>		<code>long-hyphen-short-hyphen-desc:</code>	
<code>\glxtrorglong</code>	196	<code>new</code>	296
<code>\GlsXtrDefineAcShortcuts:new</code> ...	18	<code>long-noshort-desc-noreg:new</code> ...	233
<code>\glxtrgenabbrvfmt:added check for</code>		<code>long-noshort-noreg:new</code>	234
<code>\ifglxtrininsertinside</code>	211	<code>long-only-short-only:new</code>	308
<code>\glxtrhyphensuffix:new</code>	295	<code>long-only-short-only-desc:new</code> ..	309
<code>\glxtrifhyphenstart:new</code>	293	<code>long-short-user-desc:corrected first</code>	
<code>\glxtrlonghyphen:new</code>	299	<code>forms</code>	291
<code>\glxtrlonghyphennoshort:new</code> ...	296	1.18 (2017-08-10)	
<code>\glxtrlonghyphenshort:new</code>	294	<code>stylemods:changed default value to</code>	
<code>\glxtrlongshortdescname:new</code> ...	219	<code>"default"</code>	22
<code>\glxtronlydescname:new</code>	309	1.19 (2017-09-09)	
<code>\glxtronlydescsort:new</code>	309	<code>\@glxtr@defaultnumberformat:new</code> .	7
<code>\glxtronlysuffix:new</code>	308	<code>\@glxtr@dorecord:Use</code>	
<code>\glxtrparen:new</code>	198	<code>\@glxtr@recordlocf instead of</code>	
<code>\glxtrposthyphenlong:new</code>	304	<code>\@glxtr@locf</code>	10
<code>\glxtrposthyphenshort:new</code>	299	<code>\@glxtr@dorecordnodefer:Use</code>	
<code>\glxtrposthyphensubsequent:new</code>	299	<code>\theglentrycounter for the</code>	
<code>\glxtrshortdescname:new</code>	228	<code>location rather than \@glxtr@locf</code> ..	11
<code>\glxtrshorthyphen:new</code>	304	<code>\@glxtr@record@setting:new</code>	13
<code>\glxtrshorthyphenlong:new</code>	302	<code>\@glxtr@record@setting@alsoindex:</code>	
<code>\glxtrshortlongdescname:new</code> ...	221	<code>new</code>	13
<code>\glxtrshortlongdescsort:new</code> ...	221	<code>\@glxtrifhasfield:new</code>	32
<code>\GlsXtrsubsequentfmt:new</code>	214	General:added <code>\glslink</code> option	
<code>\glxtrsubsequentfmt:new</code>	213	<code>theHvalue</code>	65
<code>\GlsXtrsubsequentplfmt:new</code>	214	added <code>\glslink</code> option <code>thevalue</code> ...	65
<code>\glxtrsubsequentplfmt:new</code>	214	<code>\glxtr@writefields:removed</code>	
<code>\glxtrword:new</code>	195	<code>double-quotes around \@jobname</code> ..	136
<code>\glxtrwordsep:new</code>	195	<code>\glxtrdoautoindexname:changed</code>	
<code>short-hyphen-long-hyphen:new</code> ...	303	<code>format test</code>	182
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink:new</code>	88
<code>new</code>	304	<code>\glxtrifhasfield:new</code>	32
<code>short-hyphen-postlong-hyphen:new</code>	305	<code>\GlsXtrSetDefaultNumberFormat:</code>	
		<code>new</code>	7

\s@glxstrifhasfield: new	32	\@glxstrsetaliasnoindex: changed to use \glxstrifhasfield instead of \ifglshasfield	84
1.20 (2017-09-11)		\@glxstrwrglossmark: new	24
\@glxstrhypernameprefix: new	122	\@rGLS: new	151
\glsdohypertarget: added redefinition	122	\@rGLS@: new	151
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glxstrhypernameprefix	142	\@rGLSpl: new	151
1.21 (2017-11-03)		\@rGLSpl@: new	151
\@@glxstr@record: added check for default options	9	\@rGLs: new	150
\@@glxstrwrglossmark: new	24	\@rGLspl: new	150
\@gls@setup@default@short@access: modified index to remove hard coded \space	379	\@rGLspl@: new	150
modified list to remove hard coded \space	368	\@rgls: new	149
moved conditional outside of \glsgroupskip	371–378	\@rgls@: new	149
redefined altlistgroup to discourage breaks after group headings	369	\@rglspl: new	149
redefined altlisthypergroup to discourage breaks after group headings	370	\@rglspl@: new	149
redefined indexgroup to discourage breaks after group headings	380	General: adjusted mcolalmtree	399
redefined indexhypergroup to discourage breaks after group headings	380	new	402
redefined listgroup to discourage breaks after group headings	369	redefined alttreegroup to discourage breaks after group headings	395
redefined listhypergroup to discourage breaks after group headings	369	redefined alttreehypergroup to discourage breaks after group headings	396
\@glslink: changed \let to \def	89	redefined mcolalmtreegroup to discourage breaks after group headings	400
\@glxstr@checkgroup: new	143	redefined mcolalmtreehypergroup to discourage breaks after group headings	400
\@glxstr@defpostpunc: new	16	redefined mcolalmtreepannav to discourage breaks after group headings	401
\@glxstr@do@record@wrglossary: new	8	redefined mcolindexgroup to discourage breaks after group headings	396
\@glxstr@dosee@alsoindex@glossary: new	25	redefined mcolindexhypergroup to discourage breaks after group headings	396
\@glxstr@doseeglossary: new	25	redefined mcolindexspannav to discourage breaks after group headings	397
\@glxstr@noidx@do: removed code dealing with the group	144	redefined mcoltreegroup to discourage breaks after group headings	397
\@glxstr@record@setting@off: new .	13	redefined mcoltreehypergroup to discourage breaks after group headings	398
\@glxstr@record@setting@only: new	13	redefined mcoltreenamegroup to discourage breaks after group	
\@glxstr@rglstrigger@record: new	148		
\@glxstrglossentry: new	138		
\@glxstrnewgls: new	145		

headings	398	\glstrbookindexatendgroup: new ..	403
redefined		\glstrbookindexbetween: new	403
mcoltreenonamehypergroup to		\glstrbookindexbookmark: new ...	403
discourage breaks after group		\glstrbookindexcols: new	402
headings	399	\glstrbookindexcolspread: new ..	403
redefined mcoltreenonamespannav to		\glstrbookindexfirstmark: new ..	407
discourage breaks after group		\glstrbookindexfirstmarkfmt: new	407
headings	399	\glstrbookindexformatheader: new	403
redefined mcoltreespannav to		\glstrbookindexgroupskip: new ..	403
discourage breaks after group		\glstrbookindexlastmark: new ...	407
headings	398	\glstrbookindexlastmarkfmt: new	407
redefined treenonamegroup to		\glstrbookindexmarkentry: new ..	407
discourage breaks after group		\glstrbookindexname: new	402
headings	383	\glstrbookindexparentchildsep:	
redefined treenonamehypergroup to		new	402
discourage breaks after group		\glstrbookindexparentschildsep:	
headings	383	new	402
debug: new	24	\glstrbookindexprelocation: new	402
\gglsssetwidest: new	385	\glstrbookindexsubatendgroup:	
\gl:disablehyper: added check for		new	403
existence	88	\glstrbookindexsubbetween: new ..	403
changed to use \def rather than \let ..	88	\glstrbookindexsubname: new	402
\gl:dohypertarget: redefined		\glstrbookindexsubprelocation:	
treegroup to discourage breaks after		new	402
group headings	381	\glstrbookindexsubsubatendgroup:	
redefined treehypergroup to		new	403
discourage breaks after group		\glstrbookindexsubsubbetween:	
headings	382	new	403
\gl:enablehyper: changed to use \def		\glstrbookindexthepage: new	406
rather than \let	89	\glstrdetoklocation: new	147
\Glsfmtname: new	324	\glstrenablerecordcount: new ...	147
\gl:fmtname: new	324	\glstrglossentry: new	137
\glshex: new	331	\glstrgroupfield: new	143
\glslstchildpostlocation: new ..	368	\Glsxrheadname: new	316
\glslstchildprelocation: new ...	368	\glstrheadname: new	315
\glslstprelocation: new	368	\GlsXtrIfFieldEqStr: new	35
\gl:navhyperlink: patched	87	\glstriflabelinlist: new	142
\gl:seeitemformat: new	46	\glstrifrecordtrigger: new	148
\gl:sshowtarget: new	25	\glstrindexseealso: added check	
\gl:treechildprelocation: new ...	379	that the entry exists	47
\gl:treeprelocation: new	379	\glstrinithyperoutside: new	65
\glstriggerrecordformat: new	149	\GlsXtrLocationRecordCount: new ..	147
\gl:suseabbrvfont: new	211	\glstrnewgls: new	145, 146
\gl:suselongfont: new	211	\glstrnewGLSlike: new	146
\gl:xttr@do@alsoindex@wrglossary:		\glstrnewglslike: new	146
new	8	\glstrnewrgls: new	146
\gl:xttr@org@@do@wrglossary: new ..	27	\glstrnewrGLSlike: new	146
\gl:xttr@org@dohyperlink: new	87	\glstrnewrglslike: new	146
\gl:xttr@setbookindexmark: new ...	407	\glstrprelocation: new	367, 402

\GlsXtrRecordCount: new	147	\@glxtr@orgprintglossary: changed	
\glxtrrecordtriggervalue: new ..	147	explicit \let for \nopostdesc to	
\glxtrresourcefile: now disables		\glxtractivatenopost	120
record key	134	\@glxtrglossentryother: new	139
\glxtrresourceinit: new	135	\glossentrynameother: new	180
\GlsXtrSetRecordCountAttribute:		\glseeitemformat: switched check	
new	147	from regular to short	46
\glxtrtitlename: new	316	\glxtr@setaccessdisplay: new ...	179
\glxtrtitleorpdforheading: new .	312	\glxtr@writefields: provide	
\GlsXtrTotalRecordCount: new	147	\glxtr@record in aux file	136
\glxtrwrglossmark: new	24	\glxtractivatenopost: new	121
short-em: new	270	\glxtrbookindexprelocation:	
short-sc: corrected first letter		removed check for no post dot	402
uppercasing	239	\glxtrglossentryother: new	138
short-sm: corrected first letter		\glxtrnopostpunc: new	121
uppercasing	253	1.23 (2017-11-12)	
shortcuts: ac	21	\@@glxtrfmt: added check for indexing	30
\ifglxtr@hyperoutside: new	65	added grouping	30
all: new	366	new	30
nolong-short: new	230	\@glxtr@nopostpunc@postdesc: new	121
nolong-short-em: new	272	\@glxtr@restore@postpunc: new ..	121
nolong-short-noreg: new	230	\@glxtrentryfmt: fixed missing label	
nolong-short-sc: new	241	argument	31
nolong-short-sm: new	255	\@glxtrfmt: new	29
nopostdot: new	16	\eglupdatewidest: new	386
postpunc: new	16	\gglupdatewidest: new	385
\printunsrtglossaryentryprocesshook:		\glupdatewidest: new	385
new	141	\GlsXtrDefineAbbreviationShortcuts:	
\printunsrtglossarypredoglossary:		changed \newabbr definition to use	
new	141	\providecommand	18
\printunsrtglossaryskipentry: new	141	\GlsXtrDefineAcShortcuts: changed	
\rGLS: new	150	\newabbr definition to use	
\rGls: new	150	\providecommand	19
\rgls: new	149	\glxtrfmtdisplay: new	30
\rGLSformat: new	152	\glxtrifcustomdiscardperiod: new	190
\rGlsformat: new	152	\GlsXtrIfFieldUndef: new	33
\rglsformat: new	151	\glxtrrestorepostpunc: new	122
\rGLSpl: new	151	\s@glxtrfmt: new	30
\rGlspl: new	150	\s@glxtrfmt: new	30
\rglspl: new	149	\xglupdatewidest: new	386
\rGLSplformat: new	152	1.24 (2017-11-14)	
\rglsplformat: new	152	\gl sadd: added \@gls@set sort	68
\rglsplformat: new	151	\glxtrforcsvfield: new	32
\s@glxtrifhasfield: switched from		\glxtrlocalsetgrouptitle: new ..	126
\ifdef to \ifundef	32	1.25 (2017-11-14)	
1.22 (2017-11-08)		\glxtrbookindexmulticol env: new	403
\@glxtr@nopostpunc: new	121	1.25 (2017-11-24)	
		\glsextrapostnamehook: new	179
		\glxtrfootnotename: new	222

\glxtrlongnoshortdescname: new .	231	\glxtrGeneralLatinIIrules: new .	341
\glxtrlongnoshortname: new	233	\glxtrGeneralLatinIrules: new ..	341
\glxtrlongshortname: new	217	\glxtrGeneralLatinIVrules: new .	343
\glxtrlongshortuserdescname: new	288	\glxtrGeneralLatinVIIrules: new	346
\glxtronlyname: new	308	\glxtrGeneralLatinVIIrules: new	345
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVrules: new .	344
changed to use \glxtrparen	191	\glxtrGeneralLatinVrules: new ..	343
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules: new ..	340
changed to use \glxtrparen	191	\glxtrgeneralpuncIrules: new ...	339
\glxtrshortlongname: new	220	\glxtrgeneralpuncrules: new	339
\glxtrshortlonguserdescname: new	290	\glxtrhyphenrules: new	339
\glxtrshortnolongname: new	226	\glxtrLatinA: new	346
1.26 (2018-01-05)		\glxtrLatinAA: new	348
\@glxtr@do@inc@linkcount: new ..	152	\glxtrLatinAELigature: new	348
\glslinkpresetkeys: new	65	\glxtrLatinE: new	346
\glxtr@inc@linkcount: new	65	\glxtrLatinEszettSs: new	348
\GlsXtrEnableLinkCounting: new ..	153	\glxtrLatinEszettSz: new	348
\GlsXtrIfLinkCounterDef: new	153	\glxtrLatinEth: new	348
\glxtrinclinkcounter: new	153	\glxtrLatinH: new	347
\GlsXtrLinkCounterName: new	153	\glxtrLatinI: new	347
\GlsXtrLinkCounterValue: new	153	\glxtrLatinInsularG: new	349
\GlsXtrTheLinkCounter: new	153	\glxtrLatinK: new	347
1.27 (2018-02-26)		\glxtrLatinL: new	347
\@glset@setup@default@short@access:		\glxtrLatinLslash: new	349
added glossaries-extra-bib2gls.sty .	330	\glxtrLatinM: new	347
\@glxtrdialecthook: new	27	\glxtrLatinN: new	347
\Alpha: new	333	\glxtrLatinO: new	347
\Beta: new	333	\glxtrLatinOELigature: new	348
\Chi: new	334	\glxtrLatinOslash: new	349
\Digamma: new	334	\glxtrLatinP: new	347
\Epsilon: new	333	\glxtrLatinS: new	347
\Eta: new	333	\glxtrLatinSchwa: new	348
\glxtr@loaddialect: new	330	\glxtrLatinT: new	347
\glxtrBasicDigitrules: new	363	\glxtrLatinThorn: new	348
\glxtrcombiningdiacriticIIrules:		\glxtrLatinWynn: new	349
new	337	\glxtrLatinX: new	348
\glxtrcombiningdiacriticIrules:		\glxtrMathGreekIIrules: new	355
new	337	\glxtrMathGreekIrules: new	354
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new	359
new	337	\glxtrMathItalicBeta: new	359
\glxtrcombiningdiacriticIVrules:		\glxtrMathItalicChi: new	362
new	338	\glxtrMathItalicDelta: new	360
\glxtrcombiningdiacriticrules:		\glxtrMathItalicEpsilon: new ...	360
new	336	\glxtrMathItalicEta: new	360
\glxtrcontrolrules: new	335	\glxtrMathItalicGamma: new	360
\glxtrcurrencyrules: new	340	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new	363	new	351
\glxtrfractionrules: new	364	\glxtrMathItalicGreekIrules: new	350
\glxtrGeneralLatinIIrules: new	342	\glxtrMathItalicIota: new	360

<code>\glxtrMathItalicKappa:new</code>	361	<code>\glxtrUpPi:new</code>	358
<code>\glxtrMathItalicLambda:new</code>	361	<code>\glxtrUpPsi:new</code>	359
<code>\glxtrMathItalicLowerGreekIIrules:</code>		<code>\glxtrUpRho:new</code>	358
<code>new</code>	353	<code>\glxtrUpSigma:new</code>	358
<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau:new</code>	359
<code>new</code>	352	<code>\glxtrUpTheta:new</code>	357
<code>\glxtrMathItalicMu:new</code>	361	<code>\glxtrUpUpsilon:new</code>	359
<code>\glxtrMathItalicNabla:new</code>	363	<code>\glxtrUpXi:new</code>	358
<code>\glxtrMathItalicNu:new</code>	361	<code>\glxtrUpZeta:new</code>	357
<code>\glxtrMathItalicOmega:new</code>	362	<code>\Iota:new</code>	333
<code>\glxtrMathItalicOmicron:new</code>	361	<code>\Kappa:new</code>	333
<code>\glxtrMathItalicPartial:new</code>	363	<code>\Mu:new</code>	333
<code>\glxtrMathItalicPhi:new</code>	362	<code>\Nu:new</code>	333
<code>\glxtrMathItalicPi:new</code>	361	<code>\Omicron:new</code>	333
<code>\glxtrMathItalicPsi:new</code>	362	<code>\omicron:new</code>	334
<code>\glxtrMathItalicRho:new</code>	361	<code>\Rho:new</code>	333
<code>\glxtrMathItalicSigma:new</code>	362	<code>\Tau:new</code>	333
<code>\glxtrMathItalicTau:new</code>	362	<code>\Upalpha:new</code>	334
<code>\glxtrMathItalicTheta:new</code>	360	<code>\Upbeta:new</code>	334
<code>\glxtrMathItalicUpperGreekIIrules:</code>		<code>\Upchi:new</code>	335
<code>new</code>	352	<code>\Upsilon:new</code>	334
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta:new</code>	334
<code>new</code>	351	<code>\Upiota:new</code>	334
<code>\glxtrMathItalicUpsilon:new</code>	362	<code>\Upkappa:new</code>	334
<code>\glxtrMathItalicXi:new</code>	361	<code>\Upmu:new</code>	334
<code>\glxtrMathItalicZeta:new</code>	360	<code>\Upnu:new</code>	334
<code>\glxtrMathUpGreekIIrules:new</code>	350	<code>\Upomicron:new</code>	334
<code>\glxtrMathUpGreekIrules:new</code>	349	<code>\upomicron:new</code>	335
<code>\glxtrnonprintablerules:new</code>	336	<code>\Uprho:new</code>	334
<code>\glxtrprovidecommand:new</code>	331	<code>\Uptau:new</code>	335
<code>\glxtrspacerules:new</code>	336	<code>\Upzeta:new</code>	334
<code>\glxtrSubScriptDigitrules:new</code>	363	<code>\Zeta:new</code>	333
<code>\glxtrSuperScriptDigitrules:new</code>	363		
<code>\glxtrUpAlpha:new</code>	356	1.28 (2018-03-06)	
<code>\glxtrUpBeta:new</code>	356	<code>\@glxtr@docdefval</code> : changed from	
<code>\glxtrUpChi:new</code>	359	count register to macro	15
<code>\glxtrUpDelta:new</code>	356	<code>\@glxtrdialecthook</code> : save and restore	
<code>\glxtrUpDigamma:new</code>	357	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon:new</code>	357	364
<code>\glxtrUpEta:new</code>	357	<code>\glxtrredeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code>	356	<code>\protected@csedef</code>	34
<code>\glxtrUpIota:new</code>	357	<code>\glxtrlocalsetgrouptitle</code> : changed	
<code>\glxtrUpKappa:new</code>	357	<code>\csedef\protected@csedef</code>	126
<code>\glxtrUpLambda:new</code>	358	<code>\glxtrsetgrouptitle</code> : changed	
<code>\glxtrUpMu:new</code>	358	<code>\csxdef\protected@csxdef</code>	125
<code>\glxtrUpNu:new</code>	358	1.29 (2018-04-09)	
<code>\glxtrUpOmega:new</code>	359	<code>\@glxtr@removespaces</code> : added expansion	128
<code>\glxtrUpOmicron:new</code>	358	<code>\@glxtr@dorecord</code> : don't suppress	
<code>\glxtrUpPhi:new</code>	359	expansion of <code>\@glxtrrecordloc</code> if	
		counter isn't page	11

\@glxstr@wrglossary@locationhyperlink: new	23	\Glsxtrlongpl: added \@glxstr@record	210
\glxstr@inc@wrglossaryctr: new ...	23	\glxtrlongpl: added \@glxstr@record	210
\glxstr@wrglossarylocation: new ..	331	\GLSxtrshort: added \@glxstr@record	205
\GlsXtrBibTeXEntryAliases: new ..	332	\Glsxtrshort: added \@glxstr@record	205
\glxtrfieldforlistloop: corrected argument order in \forlistcsloop	32	\glxtrshort: added \@glxstr@record	204
\GlsXtrIndexCounterLink: new	331	\GLSxtrshortpl: added \@glxstr@record	209
\GlsXtrInternalLocationHyperlink: new	23	\Glsxtrshortpl: added \@glxstr@record	208
\GlsXtrProvideBibTeXFields: new ..	332	\glxtrshortpl: added \@glxstr@record	208
indexcounter: new	23	\GlsXtrStartUnsetBuffering: new ..	98
\setentrycounter: new	128	\GlsXtrStopUnsetBuffering: new ...	99
1.30 (2018-04-25)		indexcounter: added check for wrglossary counter	23
\@@glxstr@record: added check for post-key hook	9	\s@GlsXtrStopUnsetBuffering: new ..	99
added check for pre-key hook	9	1.31 (2018-05-09)	
\@GLSxtr@fullpl: added \@glxstr@record	203	\@GlsXtrStartUnsetBuffering: new ..	98
\@GlsXtrStopUnsetBuffering: new ..	99	\@gl@ifaccessattribute@set: new	162
\@Glsxtr@fullpl: added \@glxstr@record	203	\@gl@initaccesskeys: new ...	162, 167
\@glxstr@dorecord: don't suppress expansion of \@glarecordlocoref ..	11	\@gl@setup@default@short@access: new	162, 167
\@glxstr@full: added \@glxstr@record	201	\@glxstr@record@noglossarywarning: new	134
\@glxstr@fullpl: added \@glxstr@record	202	\@glxtrbuffer@nodup@unset: new ..	99
\@glxstr@glossadd@postkeys: new ..	10	General: added prefix key for glslink	65
\@glxstr@glossadd@prekeys: new ...	10	added prefix key for printgloss ..	122
\@glxstr@glslink@postkeys: new ...	10	changed \let to \def	122
\@glxstr@glslink@prekeys: new	10	\glsaddeach: new	69
\@glxstr@local@textformat: new ...	65	\glsapturedgroup: new	331
\@glxstr@unset: new	98	\glsdefpostdesc: new	189
\@glxtrbuffer@unset: new	99	\glsdefpostlink: new	190
\glsadd: added \glsaddpostsetkeys	68	\glsdefpostname: new	179
added \glsaddpresetkeys	68	\glsdohypertarget: bug fix: ensure that new version is picked up	122
\glsaddpostsetkeys: new	68	\glslistdesc: new	368
\glsaddpresetkeys: new	68	\glslocalreseteach: new	100
\glsuserdescription: new	285	\glslocalunseteach: new	100
\glxtrabbreviationfont: new	62	\glstreechilddesc: new	381
\GlsXtrDualBackLink: new	332	\glstreechildsymbol: new	381
\GlsXtrDualField: new	331	\glstreedefaultnamefmt: new	378
\GlsXtrExpandedFmt: new	65	\glstreegroupheaderfmt: added redefinition	378
\GLSxtrlong: added \@glxstr@record	207	\glstreenamefmt: added redefinition	378
\Glsxtrlong: added \@glxstr@record	206		
\glxtrlong: added \@glxstr@record	206		
\GLSxtrlongpl: added \@glxstr@record	211		

\glstreenavigationfmt: added		\GlsXtrStandaloneSubEntryItem:	
redefinition	379	new	138
\glstreenonamechilddesc: new	382	\s@GlsXtrStartUnsetBuffering: new	99
\glstreenonamesymbol: new	382	1.32 (2018-05-24)	
\glstreesymbol: new	381	\GlsXtrForeignText: new	36
\glsxtr@newabbreviation: added		\GlsXtrForeignTextField: new	37
\ExtraCustomAbbreviationFields		\GlsXtrUnknownDialectWarning: new	37
.....	196	1.33 (2018-07-26)	
\GlsXtrForUnsetBufferedList: new .	99	\ifglused: added redefinition	39
\GlsXtrIfFieldCmpNum: new	33	1.34 (2018-07-29)	
\GlsXtrIfFieldEqNum: new	33	\gls@begindocdefs: atom	51
\GlsXtrIfFieldEqXpStr: new	35	\GlsXtrIfUnusedOrUndefined: new ..	27
\GlsXtrIfFieldNonZero: new	33	\glsxtrNoGlossaryWarning: added	
\GlsXtrIfHasNonZeroChildCount:		package warning	21
new	331	\if@glxtrdocdefrestricted:	
\GlsXtrIfXpFieldEqXpStr: new	35	changed to allow for atom as well ...	15
\glsxtrpostlinkAddSymbolDescOnFirstUse:		docdef: atom	15
new	191	1.35 (2018-08-13)	
\GlsXtrRecordWarning: new	132	\@gls@@link@: initialise post-link hook	
\glsxtrRevertTocMarks: new	311	commands	64
\GlsXtrStandaloneGlossaryType:			
new	138		

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\\$	331
\.	16, 17, 190, 378
\@	51, 134
\@cGLS@	102, 111
\@cGLSpl@	103, 111
\@ccGLs@	102, 111
\@ccGLspl@	102, 111
\@ccgls@	102, 111
\@ccglspl@	102, 111
\@do@wrglossary	8, 10, 117
\@do@wrglossary	12–14, 27, 68, 84
\@glo@assign@sortkey	123
\@glo@list	6
\@glo@type	140
\@glossarysec	403
\@gls@expand@field	29
\@glslocalreset	100
\@glslocalunset	100
\@glsreset	100
\@glsunset	98
\@glsxtr@autoindex@escspch	184, 185
\@glsxtr@checkspch	183, 184, 186
\@glsxtr@disabledflycommand	57
\@glsxtr@org@postdescription	121
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 137
\@glsxtrfmt	29, 30
\@glsxtrp	93, 94
\@glsxtrpostloctag	59
\@glsxtrpreloctag	59, 60
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 47, 52, 117
\@newglossaryentry@defcounters	101
\@newglossaryentry@defunitcounters	109
\@par	384
\@ACRlong	90
\@ACRlongpl	90
\@ACRshort	90
\@ACRshortpl	90
\@Acrlong	90
\@Acrlongpl	90
\@Acrshort	90
\@Acrshortpl	90
\@GLS@	89, 105, 106, 151
\@GLSdesc@	73
\@GLSpl@	89, 105, 106, 151
\@GLSplural@	90
\@GLSSymbol@	74
\@GLSxtr@full	202
\@GLSxtr@fullpl	203
\@GLSxtr@p@acrlong@	90
\@GLSxtr@p@acrlongpl@	90
\@GLSxtr@p@acrshort@	90
\@GLSxtr@p@acrshortpl@	90
\@GLSxtr@p@long@	89
\@GLSxtr@p@longpl@	90
\@GLSxtr@p@plural@	89
\@GLSxtr@p@short@	89
\@GLSxtr@p@shortpl@	90
\@GLSxtr@p@text@	89
\@GLSxtrlong	89, 207
\@GLSxtrlongpl	90, 211
\@GLSxtrp	97, 98
\@GLSxtrshort	89, 205
\@GLSxtrshortpl	90, 209
\@Gls@	89, 104, 105, 150
\@Gls@acentryname	113
\@Gls@entry@field	81, 96, 180
\@Gls@entryname	113
\@GlsXtrEnableOnTheFly	54
\@GlsXtrStartUnsetBuffering	98
\@GlsXtrStopUnsetBuffering	99

<code>\@Glspl@</code>	89, 105, 106, 150	<code>\@end@glstr@addunused</code>	50, 51
<code>\@Glsplural@</code>	90	<code>\@end@glstr@gettype</code>	119, 123
<code>\@Glstext@</code>	90	<code>\@end@glstr@usesee</code>	45, 46
<code>\@Glsxtr</code>	55, 56	<code>\@end@glstrifhyphenstart</code>	294
<code>\@Glsxtr@full</code>	201	<code>\@endfortrue</code>	32, 180, 215
<code>\@Glsxtr@fullpl</code>	203	<code>\@firstofone</code>	69, 140, 174, 175, 181, 187
<code>\@Glsxtr@p@acrlong@</code>	90	<code>\@firstofthree</code>	63,
<code>\@Glsxtr@p@acrlongpl@</code>	90		69, 77–80, 86, 201, 202, 204, 206, 208, 210
<code>\@Glsxtr@p@acrshort@</code>	90	<code>\@firstoftwo</code> ..	70–74, 78, 80, 83, 86, 115,
<code>\@Glsxtr@p@acrshortpl@</code>	90		180, 192, 193, 201–203, 208–211, 312, 313
<code>\@Glsxtr@p@long@</code>	89	<code>\@for</code>	6, 22, 32, 51, 69, 100, 101, 113,
<code>\@Glsxtr@p@longpl@</code>	90		116, 119, 126, 140, 147, 154, 173, 179, 187
<code>\@Glsxtr@p@plural@</code>	89	<code>\@glo@alias</code>	48, 49
<code>\@Glsxtr@p@short@</code>	89	<code>\@glo@assign@sortkey</code>	119
<code>\@Glsxtr@p@shortpl@</code>	89	<code>\@glo@autosee</code>	25
<code>\@Glsxtr@p@text@</code>	89	<code>\@glo@autoseehook</code>	49
<code>\@Glsxtrlong</code>	89, 206	<code>\@glo@category</code>	107
<code>\@Glsxtrlongpl</code>	90, 210	<code>\@glo@check@sortallowed</code>	119
<code>\@Glsxtrp</code>	96	<code>\@glo@counterprefix</code>	10, 11, 128
<code>\@Glsxtrpl</code>	56	<code>\@glo@countunit</code>	107
<code>\@Glsxtrshort</code>	89, 204, 205	<code>\@glo@default@sorttype</code>	119
<code>\@Glsxtrshortpl</code>	89, 208	<code>\@glo@desc</code>	40
<code>\@acrlong</code>	90	<code>\@glo@descplural</code>	40
<code>\@acrlongpl</code>	90	<code>\@glo@group</code>	13
<code>\@acrshort</code>	90	<code>\@glo@label</code>	
<code>\@acrshortpl</code>	90		12, 13, 28, 45, 48–50, 81, 88, 387–393
<code>\@addtoreset</code>	153	<code>\@glo@location</code>	12
<code>\@afterheading</code>		<code>\@glo@loclist</code>	12
	369, 370, 380, 382–384, 396–399, 406	<code>\@glo@name</code>	182
<code>\@alt@glshyp@opt</code>	86	<code>\@glo@no@assign@sortkey</code>	123
<code>\@auxout</code>	11, 12, 52, 60, 61, 103,	<code>\@glo@parent</code>	388, 389
	111, 112, 116, 117, 129, 130, 134–137, 407	<code>\@glo@see</code>	45, 46, 49–51
<code>\@bibgls@restoreat</code>	134	<code>\@glo@seealso</code>	48, 49
<code>\@cGLS</code>	106	<code>\@glo@sort</code>	182
<code>\@cGLS@</code>	102, 106, 111	<code>\@glo@sorttype</code>	119, 126
<code>\@cGLSpl</code>	106	<code>\@glo@text</code>	63
<code>\@cGLSpl@</code>	103, 106, 111	<code>\@glo@thislettergrp</code>	143
<code>\@cGls@</code>	102, 111	<code>\@glo@thisvalue</code>	284
<code>\@cGlspl@</code>	102, 111	<code>\@glo@tmp</code>	28, 47, 81
<code>\@cgls@</code>	102, 111	<code>\@glo@type</code>	50, 87, 113, 116,
<code>\@cglspl@</code>	102, 111		120, 123, 126, 127, 129, 130, 133, 134, 140
<code>\@disable@onlypremakeg</code>	117	<code>\@glo@types</code>	171, 386–393
<code>\@do@auxoutstuff</code>	129, 130	<code>\@glossary@default@style</code> ..	57, 58, 120, 401
<code>\@do@glsgetcounterprefix</code>	10, 11	<code>\@glossarystyle</code>	120
<code>\@do@glsssee</code>	49, 50	<code>\@gls@</code>	89, 104, 105, 149
<code>\@do@newglossaryentry</code>	114, 198	<code>\@gls@@link</code>	64
<code>\@do@seeglossary</code>	13, 14, 25, 52, 117	<code>\@gls@ReturnAfterFi</code>	129
<code>\@do@wrglossary</code>	67, 148, 149	<code>\@gls@actualchar</code>	183
<code>\@empty</code> ..	69, 77–80, 121, 128, 183, 184, 201–211	<code>\@gls@adjustmode</code>	68

\@gls@alt@hyp@opt	86	\@gls@long	196
\@gls@alt@hyp@opt@char	86	\@gls@longpl	194, 196–198
\@gls@alt@hyp@opt@keys	86	\@gls@map	179, 180
\@gls@automake	119	\@gls@nameaccess	161–163
\@gls@between	126	\@gls@nohyperlist	41–43
\@gls@checkedmkidx	183–186	\@gls@noidx@do	127
\@gls@checkmkidxchars	47, 182	\@gls@noidx@getgrouptitle	140
\@gls@codepage	130	\@gls@noidx@nosanitizesort	119
\@gls@counter	9, 11, 23, 66, 68, 84, 148	\@gls@noidx@sanitizesort	119
\@gls@currentlettergroup ...	126, 140, 143	\@gls@noidx@loclist@finalsep	123
\@gls@declareoption	5	\@gls@noidx@loclist@prev	123
\@gls@default@longpl	196, 197	\@gls@noidx@loclist@sep	123
\@gls@deffile	52	\@gls@noref@warn	118, 127
\@gls@doautomake	119, 137	\@gls@org@glsnoidxdisplayloc	124
\@gls@doautomake@err	137	\@gls@org@glsseeformat	124
\@gls@enablesavenonumberlist	51	\@gls@preglossaryhook	120, 187
\@gls@encapchar	183	\@gls@prevlevel	394–396, 400, 401
\@gls@entry@count	103	\@gls@quotechar	183
\@gls@entry@field		\@gls@reference	52, 53, 116, 117
.....	29, 34, 49, 81, 94–97, 102, 139	\@gls@restoreat	51
\@gls@entry@unitcount	111, 112	\@gls@saveentrycounter	13, 14, 27, 66, 68, 148
\@gls@field@font	69–76	\@gls@see@noindex	26, 134
\@gls@field@link	69–76, 81, 82	\@gls@setdefault@glslink@opts	
\@gls@firstaccess	161, 162, 164	9, 30, 66, 85
\@gls@getcounterprefix	10, 11	\@gls@setsort	66, 68
\@gls@getgrouptitle	125, 140	\@gls@setupsort@none	14
\@gls@grptitle	87, 126	\@gls@short	196, 197
\@gls@hyp@opt		\@gls@shortaccess	161–164
...	81, 82, 86, 106, 145, 149–151, 200–211	\@gls@shortaccesspl	162, 163
\@gls@hyp@opt@cs	86	\@gls@shortpl	194, 197, 198
\@gls@ifaccessattribute@set	163	\@gls@sort	143
\@gls@ifinlist	142	\@gls@textaccess	161–163
\@gls@increment@currcount	102	\@gls@thislabel	69, 100
\@gls@increment@currunitcount	110	\@gls@thisval	179, 180
\@gls@initaccesskeys	196	\@gls@tmp	35, 126
\@gls@keymap ..	12, 13, 28, 45, 48, 81, 136, 179	\@gls@tmpb	185, 186
\@gls@label ..	8, 9, 11, 52, 85, 86, 117, 137, 214	\@gls@type	117–119, 214, 387–393
\@gls@levelchar	183	\@gls@write@entrycounts	103
\@gls@link	30, 62–64, 77–81, 201–211	\@gls@write@entryunitcounts	111
\@gls@link@checkfirsthyper	63, 115	\@gls@write@entryunitcounts@do	112
\@gls@link@label	66, 148	\@gls@writedef	52
\@gls@link@nocheckfirsthyper		\@gls@xref	12, 47
.....	62, 77–80, 201–211	\@gls@abbrv@current@abbreviation	196, 211
\@gls@link@opts	66	\@gls@sacronymlists	113
\@gls@list	126	\@gls@doifexistsorwarn	15, 175, 176, 178, 180
\@gls@local@increment@currcount ...	102	\@gls@entry	52, 103, 112
\@gls@local@increment@currunitcount	110	\@gls@link	67, 87–89
\@gls@location	144	\@gls@localreset	100
\@gls@loclist	123, 124, 144	\@gls@localunset	99, 100

<code>\@glsnextpages</code>	120	<code>\@glsxtr@abbreviationsdef</code>	18, 26
<code>\@glsnonextpages</code>	120	<code>\@glsxtr@accessdisplay</code>	179–181
<code>\@glsnumberformat</code>		<code>\@glsxtr@activate@initialtagging</code> ..	
..... 9, 11, 66, 68, 84, 148, 179, 182		187, 188
<code>\@glsorder</code>	116	<code>\@glsxtr@addunitcounter</code>	107
<code>\@glspl@</code>	89, 104, 105, 150	<code>\@glsxtr@addunused</code>	51
<code>\@glsplural@</code>	90	<code>\@glsxtr@addunusedxrefs</code>	50, 51
<code>\@gls punc@token</code>	193	<code>\@glsxtr@attrval</code>	
<code>\@glsrecordlocref</code>	10, 11	67, 174, 175, 177, 178, 180, 182
<code>\@gls showtarget</code>	88	<code>\@glsxtr@autoindex@at</code>	182–184
<code>\@glsstyle@altlist</code>	368	<code>\@glsxtr@autoindex@doextra@esc</code>	182
<code>\@glsstyle@altlistgroup</code>	369	<code>\@glsxtr@autoindex@encap</code>	182–184
<code>\@glsstyle@altlisthypergroup</code>	370	<code>\@glsxtr@autoindex@esc</code>	183, 185, 186
<code>\@glsstyle@alttree</code>	384	<code>\@glsxtr@autoindex@escat</code>	183, 184
<code>\@glsstyle@alttreegroup</code>	395	<code>\@glsxtr@autoindex@escencap</code>	184
<code>\@glsstyle@alttreehypergroup</code>	396	<code>\@glsxtr@autoindex@esclevel</code> ...	183–185
<code>\@glsstyle@index</code>	379	<code>\@glsxtr@autoindex@escquote</code> ...	183, 185
<code>\@glsstyle@indexgroup</code>	380	<code>\@glsxtr@autoindex@level</code>	183, 184
<code>\@glsstyle@indexhypergroup</code>	380	<code>\@glsxtr@autoindex@setname</code>	182
<code>\@glsstyle@inline</code>	378	<code>\@glsxtr@autoindex@crossrefs</code> 14, 15, 45, 48	
<code>\@glsstyle@list</code>	368	<code>\@glsxtr@autoindex@false</code>	14
<code>\@glsstyle@listdotted</code>	367	<code>\@glsxtr@autoindex@true</code>	16
<code>\@glsstyle@listgroup</code>	369	<code>\@glsxtr@bookindex@atendgroup</code> .	404–406
<code>\@glsstyle@listhypergroup</code>	369	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	405
<code>\@glsstyle@mc colalttree</code>	399	<code>\@glsxtr@bookindex@atsubsubendgroup</code>	405
<code>\@glsstyle@mc colalttreegroup</code>	400	<code>\@glsxtr@bookindex@between</code>	404, 406
<code>\@glsstyle@mc colalttreehypergroup</code> ..	400	<code>\@glsxtr@bookindex@sep</code>	404, 405
<code>\@glsstyle@mc colalttreesspannav</code>	401	<code>\@glsxtr@bookindex@subatendgroup</code> ..	
<code>\@glsstyle@mc colindexgroup</code>	396	404–406
<code>\@glsstyle@mc colindexhypergroup</code>	396	<code>\@glsxtr@bookindex@subbetween</code> .	404, 405
<code>\@glsstyle@mc colindexspannav</code>	397	<code>\@glsxtr@bookindex@subsep</code>	404, 405
<code>\@glsstyle@mc coltreegroup</code>	397	<code>\@glsxtr@bookindex@subsubatendgroup</code>	
<code>\@glsstyle@mc coltreehypergroup</code>	398	404–406
<code>\@glsstyle@mc coltreenamegroup</code>	398	<code>\@glsxtr@bookindex@subsubbetween</code> ..	
<code>\@glsstyle@mc coltreenamehypergroup</code>	399	404, 405
<code>\@glsstyle@mc coltreename spannav</code> ..	399	<code>\@glsxtr@bookindex@groupskip</code> ...	404, 406
<code>\@glsstyle@mc coltreesspannav</code>	398	<code>\@glsxtr@cat</code>	101, 113, 147, 148, 187
<code>\@glsstyle@tree</code>	380	<code>\@glsxtr@checkgroup</code>	141
<code>\@glsstyle@treegroup</code>	381	<code>\@glsxtr@counterrecordhook</code>	11
<code>\@glsstyle@treehypergroup</code>	382	<code>\@glsxtr@csname</code>	108–111
<code>\@glsstyle@treename</code>	382	<code>\@glsxtr@current@style</code>	57, 401
<code>\@glsstyle@treenamegroup</code>	383	<code>\@glsxtr@currentunitcount</code>	108, 110
<code>\@glsstyle@treenamehypergroup</code> ...	383	<code>\@glsxtr@currunitcount</code>	109, 112
<code>\@glstarget</code>	89, 122	<code>\@glsxtr@debugnr</code>	24
<code>\@glstext@</code>	90	<code>\@glsxtr@debugval</code>	24
<code>\@glunset</code>	99	<code>\@glsxtr@declareoption</code> ...	5, 16, 18, 21, 23
<code>\@glswidestname</code>	386, 394	<code>\@glsxtr@defaultnoglossarywarning</code> ..	21
<code>\@glsxtr</code>	55, 56	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glsxtr@do@wrglossary</code>	117	7, 9, 66, 68, 84, 179, 182

\@glsxtr@defpostpunc	16, 17, 25	\@glsxtr@glossdescfont	174, 175
\@glsxtr@deprecated@abbrstyle	243, 245, 246,	\@glsxtr@glossnamefont	175–181
248, 257, 259, 260, 262, 275, 278, 282, 284		\@glsxtr@gobbleto@endescspch	186
\@glsxtr@dialect	36, 37	\@glsxtr@groupheading	141, 143
\@glsxtr@disabledflycommand	56	\@glsxtr@idx@displaynumberlist	118
\@glsxtr@display@loc	127	\@glsxtr@idx@entrynumberlist	118
\@glsxtr@do@wrindex	85, 86	\@glsxtr@ifcsstart	54
\@glsxtr@do@gl:disablehyperinlist ..	83	\@glsxtr@ifpunctoken	193
\@glsxtr@do@inc@linkcount	153	\@glsxtr@ifunitcounter	107
\@glsxtr@do@record@wrglossary	8, 14	\@glsxtr@insert@dots	195
\@glsxtr@do@redef@for:gl:sentries	7	\@glsxtr@insert@dots@next	195
\@glsxtr@do@style	22, 330	\@glsxtr@insertdots	162, 197
\@glsxtr@do@titlecaps@warn	174–177, 181, 188	\@glsxtr@label	32, 51, 154, 173
\@glsxtr@doabbreviationsdef	18	\@glsxtr@loadstyles	366, 367
\@glsxtr@doaccsupp	21, 25	\@glsxtr@local@textformat	66, 67
\@glsxtr@docdefsetting	15, 52	\@glsxtr@locale	36, 37
\@glsxtr@docdefval	15, 51–53	\@glsxtr@longnewglossaryentry	39
\@glsxtr@docounterrecord	11	\@glsxtr@mark@wordseps	195
\@glsxtr@doglossary	140, 141	\@glsxtr@mark@wordseps@next	195
\@glsxtr@doiflabelinlist	142	\@glsxtr@markwordseps	196, 197
\@glsxtr@dolocktag	59, 60	\@glsxtr@mixed@assign@sortkey	119
\@glsxtr@dorecord	8, 10	\@glsxtr@noidx@displaynumberlist ..	118
\@glsxtr@dorecordnodefer	8, 10	\@glsxtr@noidx@do	143
\@glsxtr@dosee@alsoindex@glossary ..	14	\@glsxtr@noidx@entrynumberlist	118
\@glsxtr@doseeglossary	13, 25	\@glsxtr@noidx@numberlistloop	118
\@glsxtr@dostylewarn	214, 215	\@glsxtr@nomissingglstextnr	21
\@glsxtr@enabletagging	187	\@glsxtr@nomissingglstextval	21
\@glsxtr@end@	54	\@glsxtr@noop@recordcounter	11, 13
\@glsxtr@endescspch	183–186	\@glsxtr@nopostpunc	121
\@glsxtr@entrycount@org@localreset	102	\@glsxtr@nopostpunc@postdesc	121
\@glsxtr@entrycount@org@localunset	102	\@glsxtr@notfoundinlist	193
\@glsxtr@entrycount@org@reset	102	\@glsxtr@op@recordcounter	14
\@glsxtr@entrycount@org@unset	102	\@glsxtr@optlist	56
\@glsxtr@entryunitcount@org@localreset	110	\@glsxtr@org@@starttoc	311
\@glsxtr@entryunitcount@org@localunset	110	\@glsxtr@org@GLS@	63
\@glsxtr@entryunitcount@org@reset ..	110	\@glsxtr@org@GLSpl@	63
\@glsxtr@entryunitcount@org@unset ..	110	\@glsxtr@org@Gls@	63
\@glsxtr@err@undefaction	7, 13	\@glsxtr@org@Glspl@	63
\@glsxtr@field@linkdefs	62, 64	\@glsxtr@org@Glsxtrtitlefirst .	312, 313
\@glsxtr@format@overridefalse	181	\@glsxtr@org@Glsxtrtitlefirstplural	312, 314
\@glsxtr@format@override:tr:ue ..	181, 182	\@glsxtr@org@Glsxtrtitlefull ..	312, 314
\@glsxtr@foundinlist	193	\@glsxtr@org@Glsxtrtitlefullpl	312, 314
\@glsxtr@full	200	\@glsxtr@org@Glsxtrtitlelong ..	312, 314
\@glsxtr@fullpl	202	\@glsxtr@org@Glsxtrtitlelongpl	312, 314
\@glsxtr@gettype	119	\@glsxtr@org@Glsxtrtitlename ..	312, 313
		\@glsxtr@org@Glsxtrtitleplural	312, 313
		\@glsxtr@org@Glsxtrtitleshort .	312, 313
		\@glsxtr@org@Glsxtrtitleshortpl	312, 313

<code>\@glsxtr@org@Glsxtrtitletext</code> ..	312, 313	<code>\@glsxtr@pagetag</code>	59, 60
<code>\@glsxtr@org@MakeUppercase</code>	312, 313	<code>\@glsxtr@pagetag</code>	59, 60
<code>\@glsxtr@org@checkfirsthyper</code> ...	83, 115	<code>\@glsxtr@prevunitcount</code>	110
<code>\@glsxtr@org@currentfieldvalue</code>	36	<code>\@glsxtr@printglossnr</code>	122
<code>\@glsxtr@org@delimN</code>	60	<code>\@glsxtr@printglossopts</code>	56, 119, 122
<code>\@glsxtr@org@delimR</code>	60	<code>\@glsxtr@printglossval</code>	122
<code>\@glsxtr@org@doseeglossary</code>	25, 117	<code>\@glsxtr@printunsrtglossaryskipentry</code>	140, 141
<code>\@glsxtr@org@gloautosee</code>	26	<code>\@glsxtr@provide@addstoragekey</code>	29
<code>\@glsxtr@org@glolinkprefix</code>	66, 68	<code>\@glsxtr@provide@storagekey</code>	28
<code>\@glsxtr@org@gls@</code>	62	<code>\@glsxtr@record</code> ..	13, 14, 62–64, 68, 201–211
<code>\@glsxtr@org@glsdohypertarget</code>	122	<code>\@glsxtr@record@noglossarywarning</code> ..	14
<code>\@glsxtr@org@glsignore</code>	60	<code>\@glsxtr@record@setting</code>	8, 10, 13, 47, 52, 53, 116
<code>\@glsxtr@org@glspl@</code>	62, 63	<code>\@glsxtr@record@setting@alsoindex</code> ..	8, 10, 47, 116
<code>\@glsxtr@org@glsxtrtitlefirst</code> ..	312, 313	<code>\@glsxtr@record@setting@off</code>	52
<code>\@glsxtr@org@glsxtrtitlefirstplural</code>	312, 313	<code>\@glsxtr@record@setting@only</code>	116
<code>\@glsxtr@org@glsxtrtitlefull</code> ..	312, 314	<code>\@glsxtr@recordsee</code>	14, 25, 47
<code>\@glsxtr@org@glsxtrtitlefullpl</code> ..	312, 314	<code>\@glsxtr@redef@forglsentries</code> ..	7, 26, 27
<code>\@glsxtr@org@glsxtrtitlelong</code> ..	312, 314	<code>\@glsxtr@redefstyles</code>	22, 330
<code>\@glsxtr@org@glsxtrtitlelongpl</code> ..	312, 314	<code>\@glsxtr@reg@glosslist</code>	116–119, 123
<code>\@glsxtr@org@glsxtrtitlename</code> ..	312, 313	<code>\@glsxtr@restore@postpunc</code>	121
<code>\@glsxtr@org@glsxtrtitleorpdforheading</code>	312, 313	<code>\@glsxtr@rglstrigger@record</code> ...	149–151
<code>\@glsxtr@org@glsxtrtitleplural</code> ..	312, 313	<code>\@glsxtr@s@longnewglossaryentry</code>	39
<code>\@glsxtr@org@glsxtrtitleshort</code> ..	312, 313	<code>\@glsxtr@savepreloctag</code>	59–61
<code>\@glsxtr@org@glsxtrtitleshortpl</code> ..	312, 313	<code>\@glsxtr@setentrycountunsetattr</code> ...	101
<code>\@glsxtr@org@glsxtrtitletext</code> ..	312, 313	<code>\@glsxtr@setentryunitcountunsetattr</code> ..	113
<code>\@glsxtr@org@makeglossaries</code>	116	<code>\@glsxtr@setupshortcuts</code>	20, 21, 26
<code>\@glsxtr@org@markboth</code>	311	<code>\@glsxtr@shortcutsnr</code>	20
<code>\@glsxtr@org@markright</code>	310, 311	<code>\@glsxtr@shortcutsval</code>	20, 136
<code>\@glsxtr@org@newacronymstyle</code> ..	114, 115	<code>\@glsxtr@swaptwo</code>	193
<code>\@glsxtr@org@postdescription</code> ..	121, 188, 189	<code>\@glsxtr@tag</code>	187
<code>\@glsxtr@org@see@noindex</code>	134	<code>\@glsxtr@taggingcs</code>	187
<code>\@glsxtr@org@setacronymstyle</code> ..	114, 115	<code>\@glsxtr@textformat</code>	67
<code>\@glsxtr@org@theHvalue</code>	8, 9	<code>\@glsxtr@theHvalue</code> ...	8–10, 65, 66, 68, 148
<code>\@glsxtr@org@unset@buffer</code>	98, 99	<code>\@glsxtr@thevalue</code>	8–10, 65, 66, 68, 148
<code>\@glsxtr@org@prefix</code>	10, 11	<code>\@glsxtr@thisloctag</code>	60
<code>\@glsxtr@org@printglossary</code>	56, 122	<code>\@glsxtr@titlelabel</code>	125, 126, 142, 143
<code>\@glsxtr@org@warndep</code>	194	<code>\@glsxtr@tmp</code>	22, 65, 128
<code>\@glsxtr@p@acrlong@</code>	90	<code>\@glsxtr@type</code>	173
<code>\@glsxtr@p@acrlongpl@</code>	90	<code>\@glsxtr@unitcountlist</code>	107, 108
<code>\@glsxtr@p@acrshort@</code>	90	<code>\@glsxtr@unset</code>	98, 99
<code>\@glsxtr@p@acrshortpl@</code>	90	<code>\@glsxtr@unset@buffer</code>	98, 99
<code>\@glsxtr@p@long@</code>	89	<code>\@glsxtr@unsrt@getgroupptitle</code>	140
<code>\@glsxtr@p@longpl@</code>	90	<code>\@glsxtr@usesee</code>	46
<code>\@glsxtr@p@plural@</code>	89	<code>\@glsxtr@warn@onexistsordo</code>	7, 14
<code>\@glsxtr@p@short@</code>	89	<code>\@glsxtr@warn@undefaction</code>	7, 14
<code>\@glsxtr@p@shortpl@</code>	89		
<code>\@glsxtr@p@text@</code>	89		

\@glxstr@wrglossary@locationhyperlink	\@mfu@nocaplist	188
..... 24	\@one	103, 112, 145
\@glxstr@wrglossnr	\@newglossaryentry@defcounters	101, 109
\@glxstr@wrglossval	\@newglossaryentryposthook
\@glxstrbuffer@nodup@unset 12, 13, 28, 48, 81	
\@glxstrbuffer@unset	\@newglossaryentryprehook
\@glxstrdialecthook 12, 13, 28, 39, 40, 48, 81	
\@glxstrdocdeffalse	\@enil	128, 129, 143
\@glxstrentryfmt	\@ennil	183, 184, 186, 193, 195
\@glxstrfmt	\@no@glxstrindexaliased	84, 85
\@glxstrglossentry	\@no@makeglossaries	133
\@glxstrglossentryother	\@nocounterr	154
\@glxstrhypernameprefix	\@nopostdesc	121
\@glxstrifhasfield	\@onelevel@sanitize	12, 47, 56, 125, 126, 143
\@glxstrifhyphenstart	\@onlypreamble	57,
\@glxstrindexaliased	60, 111, 135, 137, 154, 182, 184, 185, 187	
\@glxstrindexcrossrefsfalse	\@org@glossaryentrynumbers	120, 121
\@glxstrindexcrossrefstrue	\@org@newglossaryentryprehook	39, 40
\@glxstrinmark	\@print@unsrt@glossary	139, 140
\@glxstrlong	\@printgloss@setsort	119, 120
\@glxstrlongpl	\@printglossary	56, 139, 140
\@glxstrnewgls	\@printunsrt@glossary@handler	141
\@glxstrnewgls@inner	\@printunsrtglossary	139
\@glxstrnewgls@innercsname	\@rGLS	150
\@glxstrnotinmark	\@rGLS@	151
\@glxstrp	\@rGLSpl	151
\@glxstrp@opt	\@rGLSpl@	151
\@glxstrpl	\@rGls	150
\@glxstrpostloctag	\@rGls@	150
\@glxstrpreloctag	\@rGlspl	150
\@glxstrsetaliasnoindex	\@rGlspl@	150
\@glxstrshort	\@rgls	149
\@glxstrshortpl	\@rgls@	149
\@glxstrundeftag	\@rglspl	149
\@glxstrwrglossmark	\@rglspl@	149
\@gobble .. 7, 13, 16, 69, 141, 142, 195, 404–406	\@sGlsXtrEnableOnTheFly	54
\@gobbletwo	\@secondofthree	70–
\@ifnextchar	72, 77–80, 82, 201, 203, 205, 207, 208, 210	
\@ifpackageloaded	\@secondoftwo	63, 69,
154, 173, 175, 178, 179, 181, 330, 334, 365	72–80, 83, 89, 115, 122, 180, 193, 201,	
\@ifstar	202, 204–211, 225, 247, 261, 283, 312, 313	
29, 32, 39, 40, 42, 54, 86, 98, 99, 139, 187	\@sglsxtr@provide@storagekey	28
\@ifundefined	\@starttoc	311
\@ignored@glossaries	\@thirdofthree	70–72,
\@input	77–80, 82, 202, 204, 205, 207, 209, 211, 313	
\@input@	\@thirdoftwo	72–76
\@istfilename	\@this@key	180
\@makeglossary	\@tracklang@lang	37
\@mfu@domakefirstuc	\@warn@nomakeglossaries	130
187, 188		

\@xdy@main@language	130	\acl	19
\@xdycrossrefhook	47	\ACLP	19
\@xdy@language	130	\Aclp	19
\@xdy@locationclassorder	47	\aclp	19
\\	128	\ACP	19
		\Acp	19
		\acp	19
_	53, 131, 132	\ACRfullfmt	114
		\Acrfullfmt	114
		\acrfullfmt	114
		\ACRfullplfmt	114
		\Acrfullplfmt	114
		\acrfullplfmt	114
		\acronymentry	114
		\acronymfont	77–80, 92, 115
		\acronymname	17
		\acronymsort	114
		\acronymtype	18, 113–115
		\acrpluralsuffix	114, 136
		\ACS	19
		\Acs	19
		\acs	19
		\ACSP	19
		\Acsp	19
		\acsp	19
		\actualchar	185
		\addtolength	395
		\advance	103, 112, 135, 145
		\AF	18
		\Af	18
		\af	18
		\AFP	18
		\Afp	18
		\afp	18
		\AL	18
		\Al	18
		\al	18
		\ALP	18
		\Alp	18
		\alp	18
		amsmath package	23
		\AnyTrackedLanguages	330, 365
		\appto	12, 13, 22, 28, 45, 47–49, 81, 85, 86, 101, 109, 141, 142, 181, 182, 192, 195, 366
		\arabic	23, 406
		\AS	18
		\As	18
		\as	18
		\ASP	18
A			
\AA	348		
\aa	348		
\AB	18		
\Ab	18		
\ab	18		
abbreviation styles:			
long-hyphen-postshort-hyphen	299, 301		
long-hyphen-short-hyphen	296, 300		
long-postshort-user	288		
long-short-user	286		
nolong-short	230		
short	228		
short-hyphen-long-hyphen	304, 305		
short-hyphen-postlong-hyphen	304, 305, 307		
short-long-user	288		
short-nolong	228, 230		
short-nolong-desc	229		
short-postlong-user	290		
\abbreviationsname	17		
\abbrvpluralsuffix			
	136, 163, 197, 218, 220, 223, 225, 226, 228, 231, 235, 237, 238, 240, 242, 243, 245, 247, 249, 251, 252, 254, 256, 257, 259, 261, 263, 265, 267, 268, 270, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 292, 295, 297, 300, 303, 305, 308		
\ABP	18		
\Abp	18		
\abp	18		
\AC	19		
\Ac	19		
\ac	19		
\ACF	19		
\Acf	19		
\acf	19		
\ACFP	19		
\Acfp	19		
\acfp	19		
\ACL	19		
\Acl	19		

<code>\Asp</code>	18	<code>\cgl</code> s	18, 19, 101, 112
<code>\asp</code>	18	<code>\cGLSformat</code>	105
<code>\AtBeginDocument</code>	24, 27, 58, 136	<code>\cGlsformat</code>	104
<code>\AtEndDocument</code>	50, 103, 111, 129, 130	<code>\cglformat</code>	104, 106
B			
babel package	182, 184, 192	<code>\cGLSpl</code>	18, 19, 101, 112
<code>\begin</code>	126, 131, 132, 140, 368, 370–377, 380, 382, 397–401, 404	<code>\cGlspl</code>	18, 19, 101, 112
<code>\begingroup</code>	8, 9, 30, 84, 138–140, 153	<code>\cglspl</code>	18, 19, 101, 112
<code>\bgroup</code>	39, 40, 120	<code>\cGLSplformat</code>	105
bib2gls ...	23, 30, 143, 148, 149, 330, 331, 333	<code>\cGlsplformat</code>	105
C			
<code>\c@wrglossary</code>	23	<code>\cglsplformat</code>	104, 106
<code>\catcode</code>	51, 134	<code>\changes</code>	316, 380, 382
category attributes:		<code>\char</code>	125
accessinsertdots	162	<code>\columnwidth</code>	58
apospplural	197	<code>\count@</code>	103, 112
discardperiod	191	<code>\csappto</code>	38
entrycount	98, 101, 103, 112, 113	<code>\csdef</code>	28, 34, 38, 39, 81, 82, 102, 107, 108, 111, 168, 179, 189, 190, 215, 216, 224, 247, 261, 282, 286, 288–290, 300, 302, 305, 307, 367, 385
firstshortaccess	164	<code>\cseappto</code>	43
firsttuc	177	<code>\csedef</code>	109
glossdesc	173	<code>\csgdef</code>	35, 41– 43, 53, 59, 102, 103, 108, 110, 111, 385, 407
glossdescfont	174	<code>\cslet</code>	34, 40, 120
glossname	175	<code>\csletcs</code>	34, 216
glossnamefont	175, 178	<code>\csname</code>	6, 29, 42, 47, 52, 57, 61, 63, 66, 68, 77–82, 84, 93, 108, 109, 117, 126, 129, 130, 133, 134, 140, 145, 147, 148, 153, 173, 194, 201–211, 217, 394
headuc	314	<code>\cspreto</code>	39
indexname	182	<code>\csuse</code>	9, 30, 31, 42, 50, 60, 81, 82, 95, 96, 107–111, 120, 125–127, 137, 139, 142–144, 168, 179, 189, 190, 215, 216, 385, 386, 388, 389
indexonlyfirst	85	<code>\csxdef</code>	45, 48, 108, 111
insertdots	196, 197	<code>\currentglossary</code>	120, 138, 139, 406
linkcount	152	<code>\CurrentOption</code>	24, 366, 367
linkcountmaster	153	<code>\CurrentTrackedLanguage</code>	364
markshortwords	196	<code>\CurrentTrackedLanguageTag</code>	136
markwords	196, 197, 293, 295, 303	<code>\CurrentTrackedScript</code>	364, 365
nameshortaccess	163	<code>\CurrentTrackedTag</code>	330, 365
nohyper	83	<code>\CustomAbbreviationFields</code>	198, 218–222, 224, 226, 228, 231, 233, 234, 236–239, 241, 245, 246, 249, 250, 252, 254, 256, 259, 261, 263–271, 273, 275, 278, 280, 282, 285, 286, 288– 293, 295–298, 300, 301, 303–305, 307–309
nohyperfirst	70–72	D	
noshortplural	197	<code>\DeclareAcronymList</code>	113
regular	61, 106, 217–221, 223, 224, 227–234, 236, 238–240, 243–245, 247, 250–254, 257–259, 261, 264–266, 268, 269, 271, 272, 274, 276–278, 280–282, 285, 291–293, 295–298, 303, 304, 308, 310		
textformat	67		
textshortaccess	163		
<code>\cdot</code>	24		
<code>\centering</code>	403		
<code>\cGLS</code>	18, 19, 101, 112		
<code>\cGls</code>	18, 19, 101, 112		

<code>\DeclareOption</code>	5, 366	214, 218–221, 223–227, 229–233, 235,
<code>\DeclareOptionX</code>	5, 24	237–251, 253–265, 267, 269–284, 286–
<code>\def</code>	9, 10, 12–15, 25,	290, 292–294, 296, 299–302, 305–309,
	27, 30, 33, 40, 42, 46–48, 51, 54–56, 59,	368, 370–381, 383, 394, 395, 401, 403, 405
	61–66, 68–80, 86, 88–93, 99, 104–106,	<code>\emph</code>
	113, 117, 119–123, 125–129, 140, 143,	262, 263
	145, 148–151, 161–163, 183–188, 192–	<code>\empty</code>
	197, 201–211, 214, 294, 296, 299, 302,	127, 128
	304, 305, 365, 378, 379, 394–396, 400, 401	<code>\encapchar</code>
<code>\defglentryfmt</code>	41–43	185
<code>\define@boolkey</code>	15, 16, 65, 83	<code>\end</code>
<code>\define@choicekey</code>		122, 127,
	7, 13, 15, 16, 20, 21, 24, 64, 122	131, 132, 141, 368, 370–377, 397–401, 404
<code>\define@key</code>	12, 13,	<code>\end@glstr@display@loc</code>
	16, 22, 28, 48, 65, 68, 81, 122, 161, 162, 194	127
<code>\DefineAcronymSynonyms</code>	20	<code>\endcsname</code>
<code>\delimN</code>	60	6, 29, 42, 47, 52, 57,
<code>\delimR</code>	60	61, 63, 66, 68, 77–82, 84, 93, 108, 109,
<code>\detokenize</code>	54	117, 126, 129, 130, 133, 134, 140, 145,
<code>\dimen@</code>	115, 385–393	147, 148, 153, 173, 194, 201–211, 217, 394
<code>\dimen@i</code>	388, 389	<code>\endgroup</code>
<code>\dimen@ii</code>	385, 386, 388, 389	8, 10, 30, 84, 138–140, 153
<code>\dimexpr</code>	58, 384	<code>\ensuremath</code>
<code>\disable@keys</code>	17, 27, 53, 134	24
<code>\do</code>	6, 22, 32, 51, 69, 100, 101, 113,	entry categories:
	116, 119, 126, 140, 147, 154, 173, 179, 187	abbreviation
<code>\do@gl@link@checkfirsthyper</code>		211
	30, 62–64, 66, 77–80, 201–211	general
<code>\do@gl@disablehyperinlist</code>	66, 84	168, 170
doc package	185	index
<code>\dolistcsloop</code>	31	171
<code>\DTLifinlist</code>	117, 118, 123	<code>\epreto</code>
<code>\DTLifint</code>	125	182
		<code>\equal</code>
		133, 190
		etoolbox package
		5
		<code>\expandafter</code>
		24, 29, 30, 32,
		36, 45, 46, 50, 51, 54–56, 65, 81, 82, 86,
		93, 99, 106, 107, 117–119, 123, 126, 128,
		140, 141, 143, 145, 152, 163, 173, 176,
		177, 180–182, 185, 193, 196–198, 293, 404
		<code>\expandonce</code>
		114, 143, 162–164, 183, 184, 219, 297
		<code>\ExtraCustomAbbreviationFields</code>
		162–164, 196, 198
		F
		<code>\fi</code> ..
		7–12, 14–16, 18, 20, 21, 24–26, 30, 33,
		38, 45, 47, 48, 50, 52–54, 57–59, 61, 64,
		67, 84, 85, 103, 104, 111, 112, 115, 119–
		122, 125, 127–131, 133–135, 137, 148,
		149, 182–184, 186, 193, 195, 197, 204–
		211, 213, 214, 218–221, 223–233, 235,
		237–251, 253–265, 267, 269–284, 286–
		290, 292–294, 296, 299–302, 305–309,
		368, 371–381, 383, 385–395, 401, 403, 406
		first use
		408
		flag
		408
		text
		408
		<code>\firstacronymfont</code>
		115
		fontspec package
		136
		<code>\footnote</code>
		222
		<code>\forall glossaries</code>
		50, 140, 171, 173, 387–393

E

<code>\eappto</code>	11, 22, 41–43, 141, 143, 162–164, 182, 366
<code>\edef</code>	6, 8–11, 36, 41–44, 47,
	49–52, 66–68, 83, 84, 87, 88, 107, 108,
	110, 116–118, 123, 125, 127–130, 134,
	138, 139, 148, 153, 174, 175, 177–180,
	183, 184, 186, 194, 364, 388, 389, 404, 405
<code>\eglssetwidest</code>	387–393
<code>\egroup</code>	40, 121
<code>\else</code>	8–12, 15, 16, 18, 20,
	21, 25, 26, 30, 33, 38, 51, 53, 54, 59, 61,
	63, 67, 84, 85, 104, 115, 116, 119–122,
	125, 127–129, 131, 133, 135, 148, 149,
	181–183, 186, 193, 195, 197, 204–211,

<code>\forallforallglsentries</code>	52, 103, 112	<code>listdottedstyle</code>	368
<code>\forallforeachtrackeddialect</code>	330, 365	<code>listgroup</code>	369
<code>\forallforeignlanguage</code>	36, 37	<code>listhypergroup</code>	369
<code>\forallforglsentries</code>	6, 50, 171, 173, 387–393	<code>mcolalmtree</code>	399
<code>\forallforlistcsloop</code>	32, 112, 127	<code>mcolalmtreegroup</code>	400
<code>\forallforlistloop</code>	99, 123, 124, 188	<code>mcolalmtreehypergroup</code>	400
<code>\forallfuturelet</code>	193	<code>mcolalmtreespannav</code>	401
G			
<code>\gdef</code>	60, 184, 185	<code>mcolindexgroup</code>	396
<code>\Genacrfullformat</code>	114	<code>mcolindexhypergroup</code>	396
<code>\genacrfullformat</code>	114	<code>mcolindexspannav</code>	397
<code>\GenericAcronymFields</code>	114	<code>mcoltreegroup</code>	397
<code>\Genplacrfullformat</code>	114	<code>mcoltreehypergroup</code>	398
<code>\genplacrfullformat</code>	114	<code>mcoltreenonamegroup</code>	398
<code>\GetTrackedDialectFromLanguageTag</code> ..	36	<code>mcoltreenonamehypergroup</code>	399
<code>\GetTrackedDialectToMapping</code>	36	<code>mcoltreenonamespannav</code>	399
<code>\glo@grabfirst</code>	143	<code>mcoltreespannav</code>	398
<code>\glo@name</code>	176, 177, 181	<code>sublistdotted</code>	368
<code>\gloaliaslabel</code>	88	<code>tree</code>	380
<code>\global</code>	10, 40, 52, 121, 144	<code>treegroup</code>	381
<code>\glolinkprefix</code>	65–68, 88, 122, 136	<code>treehypergroup</code>	382
<code>glossaries package</code>		<code>treenoname</code>	382
.....	14, 25, 26, 38, 45, 47–49, 119, 367	<code>treenonamegroup</code>	383
<code>glossaries-accsupp package</code>	21, 25, 154, 198	<code>treenonamehypergroup</code>	383
<code>glossaries-extra package</code>	2, 365	<code>glossary-bookindex package</code>	367
<code>glossaries-extra-bib2gls package</code> ..	14, 27, 330, 365	<code>glossary-hypernav package</code>	87
<code>glossaries-extra-stylemods package</code> ..	21, 189, 330	<code>glossary-long package</code>	372
<code>glossaries-stylemods package</code>	402	<code>glossary-longbooktabs package</code>	372
<code>glossaries.sty package</code>	40	<code>glossary-tree package</code>	378, 379
<code>\GlossariesExtraWarning</code> ..	6, 16, 21, 36–	<code>\glossaryentrynumbers</code> ...	61, 120, 121, 144
.....	39, 54, 56, 67, 115, 118, 128, 131, 134,	<code>\glossaryheader</code>	127,
.....	140, 174, 175, 177, 178, 180, 187, 188, 217	140, 368–377, 379–383, 394, 396–400, 404
<code>\GlossariesExtraWarningNoLine</code> ..	16, 103, 112	<code>\glossaryname</code>	120
<code>\GlossariesWarning</code> ..	59, 118, 120, 123, 124, 214	<code>\glossarypostamble</code>	127, 141, 142
<code>\GlossariesWarningNoLine</code>	117, 130	<code>\glossarypreamble</code>	126, 140
<code>glossary styles:</code>		<code>\glossarysection</code> ..	126, 127, 133, 134, 140, 142
<code>altlist</code>	368	<code>\glossarytitle</code> ..	42, 120, 126, 127, 133, 134, 140
<code>altlistgroup</code>	369	<code>\glossarytoctitle</code>	
<code>altlisthypergroup</code>	370	42, 120, 126, 127, 133, 134, 140
<code>almtree</code>	384, 385, 394	<code>\glossentry</code>	
<code>almtreegroup</code>	395	120, 144, 367–377, 379, 381, 383, 394, 404
<code>almtreehypergroup</code>	396	<code>\glossentrydesc</code>	
<code>index</code>	379	367, 368, 370–377, 380–382, 384
<code>indexgroup</code>	380	<code>\glossentryname</code>	
<code>indexhypergroup</code>	380	138, 367–377, 379, 381, 383, 394, 395, 402
<code>inline</code>	378	<code>\glossentrynameother</code>	139
<code>list</code>	368	<code>\glossentrysymbol</code> ..	371–375, 377, 381, 382, 384
<code>listdotted</code>	367	<code>\glossxtrsetpopts</code>	189
		<code>\glostyle</code>	380, 382
		<code>\GLS</code>	101, 112, 147

<code>\Gls</code>	55, 101, 112, 147	<code>\glsaccessfirst</code>	70
<code>\gls</code>	38, 55, 57, 101, 112, 118, 131, 147	<code>\GLSaccessfirstplural</code>	72
<code>\gls@assign@desc</code>	40	<code>\Glsaccessfirstplural</code>	72
<code>\gls@assign@field</code>	12, 13, 28, 81	<code>\glsaccessfirstplural</code>	71
<code>\gls@checkseeallowed</code>	52–54, 117	<code>\Glsaccesslong</code>	
<code>\gls@codepage</code>	130	.. 79, 199, 207, 219, 227, 232, 235, 241,	
<code>\gls@defdocnewglossaryentry</code> ..	51, 102, 109	242, 244, 250, 255–258, 264, 265, 273–	
<code>\gls@defglossaryentry</code>	40, 52, 55, 56	279, 286, 287, 295, 297, 298, 301, 303, 309	
<code>\gls@dotocitle</code>	120	<code>\glsaccesslong</code>	79, 199, 206, 207,
<code>\gls@glossary</code>	47	218, 221, 223–227, 229–232, 235, 237–	
<code>\gls@grplabel</code>	87	246, 248, 249, 251, 253–260, 262, 263,	
<code>\gls@ifnotmeasuring</code>	100	265, 267, 269, 270, 272–284, 286, 287,	
<code>\gls@level</code>	144	290, 292, 295, 297, 298, 301, 303, 308, 309	
<code>\gls@noidxglossary</code>	117	<code>\Glsaccesslongpl</code>	80,
<code>\gls@org@glossaryentryfield</code>	120	199, 210, 219, 227, 232, 235, 241–244,	
<code>\gls@org@glossarysubentryfield</code>	120	250, 255–258, 264, 265, 273, 274, 276–	
<code>\gls@orgTrackLangRequireDialectPrefix</code>		279, 286–288, 295, 297, 298, 301, 303, 309	
.....	364, 365	<code>\glsaccesslongpl</code>	
<code>\gls@save@numberlist</code>	59, 61	80, 199, 210, 211, 218, 221, 223,
<code>\gls@set@xr@key</code>	48	224, 226, 227, 229, 230, 232, 235, 237,	
<code>\gls@tmplen</code>	387–393, 395	239–246, 248, 249, 251, 253–260, 262,	
<code>\gls@type</code>	117	263, 265, 267, 269–284, 286, 287, 290,	
<code>\glsabbrvdefaultfont</code> ...	200, 218, 220,	292, 293, 295, 297, 298, 301, 303, 308, 309	
223, 225, 226, 228, 231, 284, 294, 297, 307		<code>\GLSaccessname</code>	72
<code>\glsabbrvmfont</code>		<code>\Glsaccessname</code>	72
.....	262–271, 273, 275, 277, 279, 281–283	<code>\glsaccessname</code>	46, 72
<code>\glsabbrvfont</code>		<code>\GLSaccessplural</code>	71
..	90, 91, 115, 200, 204, 205, 208, 209,	<code>\Glsaccessplural</code>	71
211, 213, 214, 218–226, 228, 231, 233,		<code>\glsaccessplural</code>	71
235, 237, 238, 240, 242, 243, 245, 247,		<code>\Glsaccessshort</code>	77, 205, 214, 221,
249, 251, 252, 254, 256, 257, 259, 261,		223–226, 229, 230, 237, 239, 240, 246,	
263, 265, 267, 268, 270, 271, 273, 275,		248, 251, 253, 254, 260, 262, 267, 269,	
277, 279, 281, 283, 285, 287, 289, 291–		270, 272, 281–283, 289, 290, 292, 301, 306	
293, 295, 297, 300, 301, 303, 305, 306, 308		<code>\glsaccessshort</code> ..	77, 199, 204, 205, 213,
<code>\glsabbrvhyphenfont</code>		218–220, 223, 225, 227–230, 232, 235,	
.....	294–296, 300, 302–305, 307	237–251, 253–258, 260–265, 267, 269–	
<code>\glsabbrvonlyfont</code>	307, 308, 310	274, 276–279, 281–283, 286, 287, 289,	
<code>\glsabbrvscfont</code> ..	234–240, 242, 243, 245, 247	290, 292, 295, 297, 298, 300, 303, 306, 309	
<code>\glsabbrvsmfont</code>		<code>\Glsaccessshortpl</code> ..	78, 209, 214, 221, 223–
.....	248–252, 254, 256, 257, 259, 261	226, 229, 230, 237, 239, 240, 246, 248,	
<code>\glsabbrvuserfont</code>	284–290, 292	251, 253, 254, 260, 262, 267, 269, 270,	
<code>\GLSaccessdesc</code>	73	272, 281–284, 289, 290, 293, 301, 306, 309	
<code>\Glsaccessdesc</code>	73, 174, 186	<code>\glsaccessshortpl</code>	78, 199,
<code>\glsaccessdesc</code>	73, 174, 191	208, 209, 214, 218, 219, 221, 223–225,	
<code>\GLSaccessdescplural</code>	73	227, 229, 230, 232, 235, 237–244, 246–	
<code>\Glsaccessdescplural</code>	73	251, 253–255, 257, 258, 260, 262, 264–	
<code>\glsaccessdescplural</code>	73	267, 269–274, 276, 278, 279, 281–283,	
<code>\GLSaccessfirst</code>	71	286–290, 292, 295, 298, 300, 303, 306, 309	
<code>\Glsaccessfirst</code>	70	<code>\GLSaccesssymbol</code>	74

<code>\Glsaccesssymbol</code>	74, 186	<code>\glsdoifexistsorwarn</code>	15, 173, 174, 186
<code>\glsaccesssymbol</code>	74, 186, 191	<code>\glsdoifnoexists</code>	39, 40
<code>\GLSaccesssymbolplural</code>	74	<code>\glsdonohyperlink</code>	67, 89
<code>\Glsaccesssymbolplural</code>	74	<code>\glsdosanitizesort</code>	118
<code>\glsaccesssymbolplural</code>	74	<code>\glsenableentrycount</code>	101, 103, 111
<code>\GLSaccessstext</code>	70	<code>\glsenableentryunitcount</code>	103, 112
<code>\Glsaccessstext</code>	70	<code>\glsentrycounter</code>	24, 128
<code>\glsaccessstext</code>	46, 69	<code>\GlsEntryCounterLabelPrefix</code>	38
<code>\glsacrshortcutstrue</code>	20, 21	<code>\glsentrycurrcount</code>	102, 103, 109
<code>\glsacspacemax</code>	115	<code>\Glsentrydesc</code>	158, 166, 175
<code>\glsadd</code>	30, 51, 69, 131	<code>\glsentrydesc</code>	158, 159, 166, 175
<code>\glsadd options</code>		<code>\Glsentrydescplural</code>	159, 166
<code>theHvalue</code>	9	<code>\glsentrydescplural</code>	159, 166
<code>thevalue</code>	9, 10	<code>\Glsentryfirst</code>	107, 152, 156, 165
<code>\glsaddpostsetkeys</code>	10, 68	<code>\glsentryfirst</code>	106, 151, 156, 165, 326
<code>\glsaddpresetkeys</code>	10, 68	<code>\Glsentryfirstplural</code> ...	107, 152, 157, 165
<code>\glsaddstoragekey</code>	49, 168, 332, 333	<code>\glsentryfirstplural</code>	
<code>\glsbackslash</code>	54	107, 151, 156, 157, 165, 327
<code>\glscapscase</code>	63, 69–80, 82, 201–213	<code>\glsentryfmt</code>	41–43
<code>\glscategory</code>		<code>\Glsentryfull</code>	114
..	61, 69, 83, 90, 91, 169, 170, 173–175,	<code>\glsentryfull</code>	114
	177–180, 186, 189, 190, 201–205, 208, 209	<code>\Glsentryfullpl</code>	114
<code>\glscategorylabel</code>		<code>\glsentryfullpl</code>	114
..	83, 162–164, 194, 196, 197, 224, 247,	<code>\glsentryitem</code>	
	261, 282, 286, 288–290, 300, 302, 305, 307	138, 139, 367–377, 379, 381, 383, 394, 405
<code>\glsclosebrace</code>	47, 132, 133	<code>\Glsentrylong</code>	91, 92, 107, 152, 161, 167
<code>\glscounter</code>	9	<code>\glsentrylong</code> .	91, 92, 106, 151, 160, 161,
<code>\glscurrententrylabel</code>	167, 224, 247, 261, 282, 289, 291, 305, 327
.....	59, 60, 120, 129, 138–141, 188, 189	<code>\Glsentrylongpl</code>	92, 107, 161, 167
<code>\glscurrentfieldvalue</code>		<code>\glsentrylongpl</code> ...	92, 93, 107, 161, 167, 328
.....	30–33, 35, 36, 284, 331, 332	<code>\Glsentrylongplural</code>	152
<code>\glscustomtext</code> ..	62–64, 77–80, 201–211, 213	<code>\glsentrylongplural</code>	151
<code>\glsdefaulttype</code>	6,	<code>\Glsentryname</code>	154, 164, 176–179
	17, 38, 39, 119, 120, 131, 139, 140, 142	<code>\glsentryname</code>	
<code>\glsdescriptionaccessdisplay</code>	158, 159, 174	138, 154, 164, 183, 324, 325, 387–393, 407
<code>\glsdescriptionpluralaccessdisplay</code>	159	<code>\glsentrynumberlist</code>	118, 125, 391–393
<code>\glsdescwidth</code>	370–377	<code>\Glsentryplural</code>	155, 165
<code>\glsdetoklabel</code>		<code>\glsentryplural</code>	155, 156, 165, 325, 326
.....	8, 9, 28, 31–35, 38–40, 44–46,	<code>\glsentryprevcount</code>	102, 104, 110
	50, 52–54, 66, 68, 84, 88, 102, 103, 108–	<code>\glsentryprevmaxcount</code>	110
	112, 117, 120, 123, 124, 138, 139, 143,	<code>\glsentryprevtotalcount</code>	110
	144, 147, 148, 173, 176, 177, 181, 388, 389	<code>\Glsentryshort</code>	91, 92, 160, 167
<code>\glsdisplaynumberlist</code>	118, 123	<code>\glsentryshort</code>	90–92, 115,
<code>\glsdohyperlink</code>	87, 89	159, 160, 166, 167, 286, 288, 299, 323, 324
<code>\glsdohypertarget</code>	89, 122	<code>\Glsentryshortpl</code>	91, 92, 160, 167
<code>\glsdoifexists</code>	15,	<code>\glsentryshortpl</code> ..	91, 92, 160, 167, 323, 324
	25, 34, 39, 43, 45–47, 62, 63, 68, 77–80,	<code>\Glsentrysymbol</code>	157, 166
	88, 100, 117, 123, 124, 138, 139, 201–211	<code>\glsentrysymbol</code>	157, 165, 166, 390–392
<code>\glsdoifexistsordo</code>	30, 31, 64	<code>\Glsentrysymbolplural</code>	158, 166

<code>\glsentrysymbolplural</code>	158, 166	256, 258, 259, 261, 263, 265, 267, 269–
<code>\Glsentrytext</code>	155, 165	271, 273, 275, 277, 279, 281, 283, 286,
<code>\glsentrytext</code>	88, 155, 165, 325	287, 289, 292, 295, 297, 300, 303, 306, 308
<code>\glsentrytype</code>	138	<code>\glsfirstlongfootnotefont</code>
<code>\Glsentryuseri</code>	75 222–226, 245–248, 259–262, 280–284
<code>\glsentryuseri</code>	75	<code>\glsfirstlonghyphenfont</code>
<code>\Glsentryuserii</code>	75	294–306
<code>\glsentryuserii</code>	75	<code>\glsfirstlongonlyfont</code>
<code>\Glsentryuseriii</code>	75	308–310
<code>\glsentryuseriii</code>	75	<code>\glsfirstlonguserfont</code>
<code>\Glsentryuseriv</code>	76	285–293
<code>\glsentryuseriv</code>	76	<code>\GLSfirstplural</code>
<code>\Glsentryuserv</code>	76	319
<code>\glsentryuserv</code>	76	<code>\Glsfirstplural</code>
<code>\Glsentryuservi</code>	76	319
<code>\glsentryuservi</code>	76	<code>\glsfirstpluralaccessdisplay</code> ..
<code>\glsextrapostnamehook</code>	179	156, 157
<code>\glsfieldfetch</code>	88	<code>\glsforeachincategory</code>
<code>\glsfieldxdef</code>	173	214
<code>\glsfindwidesttoplevelname</code>	386	<code>\glsgenentryfmt</code>
<code>\GLSfirst</code>	318	61
<code>\Glsfirst</code>	319	<code>\glsgetattribute</code> 67, 87, 88, 104, 108–110,
<code>\glsfirst</code>	318	129, 148, 153, 174, 175, 177, 178, 180, 182
<code>\glsfirstabbrvdefaultfont</code>	200, 218, 220, 223, 225, 226, 228, 231, 297	<code>\glsgetcategoryattribute</code>
<code>\glsfirstabbrvmfont</code>	263–284	169
<code>\glsfirstabbrvfont</code>	115, 199,	<code>\glsgetgrouptitle</code>
218–232, 235, 237, 238, 240, 242, 243,	 369, 370, 380, 382–384, 395–401
245, 247, 249, 251, 252, 254, 256, 257,		<code>\glsgetwidestname</code>
259, 261, 263, 265, 267, 268, 270, 271,		385
273, 275, 277, 279, 281, 283, 286, 287,		<code>\glsgroupheading</code>
289, 292, 295, 297, 298, 300, 303, 305, 308	 143, 368–377, 379–384, 394–401, 406
<code>\glsfirstabbrvhyphenfont</code>		<code>\glsgroupskip</code>
..... 294–296, 299, 300, 302–307		143, 368, 370–378, 380, 381, 383, 395, 406
<code>\glsfirstabbrvonlyfont</code>	308, 309	<code>\glshasattribute</code>
<code>\glsfirstabbrvscfont</code>	235–248	67, 87, 103, 104,
<code>\glsfirstabbrvsmfont</code>	249–262	108, 110, 112, 129, 148, 153, 174, 175,
<code>\glsfirstabbrvuserfont</code>	285–293	177, 178, 180, 182, 218–220, 222, 223,
<code>\glsfirstaccessdisplay</code>	156	225, 228, 229, 231, 233–236, 238, 245,
<code>\glsfirstlongdefaultfont</code>		247, 249–252, 259, 261, 263–269, 277,
218, 220, 226, 228, 231, 234–245, 249–		280, 281, 283, 285, 286, 288, 289, 291–
259, 263, 264, 266–268, 270–275, 277, 278		293, 295–298, 300, 302–305, 307, 308, 310
<code>\glsfirstlongemfont</code>		<code>\glshascategoryattribute</code>
..... 265, 266, 268, 269, 275, 276, 278–280		169
<code>\glsfirstlongfont</code>	199, 218–	<code>\glshex</code> ..
221, 223, 225–233, 235, 237, 238, 240,		335–341, 343–349, 351–354, 356–364
242, 243, 245, 247, 249, 251, 252, 254,		<code>\glshyperlink</code>
		88, 332
		<code>\glshypernavsep</code>
		126
		<code>\glshypernumber</code>
		129, 181, 182
		<code>\glsifattribute</code>
	 64, 65, 70, 83, 85, 94, 152, 171,
		174–177, 180, 181, 188, 191, 192, 314–323
		<code>\glsifcategory</code>
		171
		<code>\glsifcategoryattribute</code>
	 83, 162–164, 169, 170, 196, 197
		<code>\glsifnotregular</code>
		69
		<code>\glsifnotregularcategory</code>
		170
		<code>\glsifplural</code>
		.. 63, 69, 71–74, 77–80, 191, 192, 201–212
		<code>\glsifregular</code>
		61, 69, 106, 107, 151, 152
		<code>\glsifregularcategory</code>
		170
		<code>\glsifusetranslator</code>
		42

<code>\glsignore</code>	60	243, 245, 247, 249, 251, 253, 254, 256,
<code>\glsinlinedescformat</code>	378	258, 259, 261, 263, 265, 267, 269–271,
<code>\glsinlinesubdescformat</code>	378	274, 275, 277, 279, 281, 283, 286, 287,
<code>\glsinsert</code>	63,	289, 292, 295, 297, 300, 303, 306, 308, 309
	69, 77–80, 201–213, 293, 300, 302, 305, 307	
<code>\glskeylisttok</code>	113, 114, 196, 198	<code>\glslongfootnotefont</code>
<code>\glslabel</code>	8, 9, 30, 44, 46,	222, 223, 225, 245, 247, 259, 261, 281, 283
	61, 64–67, 83, 84, 87, 88, 115, 148, 152,	<code>\glslonghyphenfont</code>
	153, 189–191, 212, 213, 224, 247, 261, 294, 295, 297, 298, 300, 303, 305, 306
	282, 286, 288, 289, 291, 300, 302, 305, 307	<code>\glslongonlyfont</code>
<code>\glslabeltok</code> .	113, 114, 196, 198, 218–223,	308
	225, 226, 228–231, 233–238, 240, 242,	<code>\glslongpltok</code>
	245, 247, 249–252, 254, 256, 259, 261,	. 198, 218–222, 231, 233, 235–237, 242,
	263–271, 273, 275, 277, 279–281, 283,	245, 249–252, 256, 259, 263–266, 268,
	285–293, 295–300, 302–305, 307, 308, 310	269, 273, 275, 278, 279, 281, 285, 286,
<code>\glsletentryfield</code>	183	288–293, 295–298, 300, 302–304, 308, 310
<code>\glslink</code>	114	<code>\glslongpluralaccessdisplay</code>
<code>\glslink options</code>		161
counter	10	<code>\glslongtok</code>
format	181 113, 114, 196, 198, 218–222, 224,
hyper	310	226, 228, 231, 233–238, 240, 242, 245,
hyperoutside	65	247, 249–252, 254, 256, 259, 261, 263–
noindex	8, 9, 83, 310	271, 273, 275, 278, 280, 282, 285, 286,
textformat	67	288–293, 295–298, 300, 302–305, 308–310
theHvalue	66	<code>\glslonguserfont</code>
thevalue	66, 145	285–290, 292
wrgloss	8, 64	<code>\glsmcols</code>
<code>\glslinkcheckfirsthyperhook</code>	83	397–401
<code>\glslinkpostsetkeys</code>	10, 66, 148	<code>\GLSname</code>
<code>\glslinkpresetkeys</code>	10, 66, 148	316
<code>\glslinkvar</code>	86	<code>\Glsname</code>
<code>\glslistchildpostlocation</code>	368	316
<code>\glslistchildprelocation</code>	368, 369	<code>\glsnameaccessdisplay</code>
<code>\glslistdesc</code>	368, 369	154, 176, 178
<code>\glslistdottedwidth</code>	367	<code>\glsnamefont</code>
<code>\glslistgroupheaderfmt</code>	369, 370	175–180
<code>\glslistnavigationitem</code>	369, 370	<code>\glsnavhyperlink</code>
<code>\glslistprelocation</code>	368, 369	126
<code>\glslocalunset</code>	63, 149	<code>\glsnavhyperlinkname</code>
<code>\glslongaccessdisplay</code>	160, 161	87
<code>\glslongdefaultfont</code> .	200, 218, 220, 222,	<code>\glsnavhypertarget</code>
	226, 228, 231, 235, 237, 238, 240, 242– 369, 370, 380, 382, 384, 396–401
	244, 249, 251, 253, 254, 256, 258, 263,	<code>\glsnavigation</code>
	267, 270, 271, 273, 274, 277, 285, 294, 307 369, 370, 380, 382, 384, 396–401
<code>\glslongemfont</code>		<code>\glsnextpages</code>
 263, 265, 268, 269, 275, 276, 278, 279	120
<code>\glslongfont</code>	91, 92, 200, 206,	<code>\glsnoidxdisplayloc</code>
	207, 210, 211, 218–221, 223, 225, 226,	124
	228, 231–233, 235, 237, 238, 240, 242,	<code>\glsnoidxdisplayloclisthandler</code>
		123
		<code>\glsnoidxloclist</code>
		124, 144
		<code>\glsnoidxnumberlistloophandler</code>
		124
		<code>\glsnonextpages</code>
		120
		<code>\glsnonumberlistfalse</code>
		59
		<code>\glsnonumberlisttrue</code>
		59
		<code>\glsnopostdotfalse</code>
		121
		<code>\glsnopostdottrue</code>
		121
		<code>\glsnumberlistloop</code>
		118
		<code>\glsnumlistlastsep</code>
		123
		<code>\glsnumlistsep</code>
		123
		<code>\glsopenbrace</code>
		47, 132, 133
		<code>\glsorder</code>
		116

<code>\glspagelistwidth</code>	371, 373, 375, 377	<code>\GLStext</code>	316, 317
<code>\glspar</code>	142	<code>\Glstext</code>	317
<code>\GLSpl</code>	101, 112, 147	<code>\glstext</code>	316, 317
<code>\Glspl</code>	56, 101, 112, 147	<code>\glstextaccessdisplay</code>	155
<code>\glspl</code>	56, 101, 112, 147	<code>\glstextformat</code>	64, 67
<code>\GLSplural</code>	317, 318	<code>\glstextup</code>	234
<code>\Glsplural</code>	318	<code>\glstreechilddesc</code>	379, 381
<code>\glsplural</code>	317	<code>\glstreechildpredesc</code>	381
<code>\glspluralaccessdisplay</code>	155, 156	<code>\glstreechildprelocation</code> ...	379, 381, 383
<code>\glspluralsuffix</code>	136, 196, 200	<code>\glstreechildsymbol</code>	379, 381
<code>\glspostdescription</code>	16, 17, 121, 188, 367, 368, 370–377, 380–382, 384	<code>\glstreedefaultnamefmt</code>	378, 379
<code>\glspostinline</code>	378	<code>\glstreedesc</code>	379–381
<code>\glspostlinkhook</code> .	62, 64, 77–81, 93, 201–211	<code>\glstreegroupheaderfmt</code>	380, 382–384, 395–401, 403
<code>\glsprestandardsort</code>	118	<code>\glstreeindent</code>	381, 383, 393–395
<code>\glsresetentrylist</code>	127, 140	<code>\glstreeitem</code>	379, 397, 405
<code>\glsee</code>	48–50	<code>\glstreenamebox</code>	394, 395
<code>\glseeformat</code>	46, 52, 117, 124	<code>\glstreenamefmt</code>	378, 379, 381, 383, 385, 387–395
<code>\glseeelist</code>	47	<code>\glstreenavigationfmt</code>	380, 382, 384, 396–401
<code>\glsssetabbrvfmt</code> .	61, 69, 90, 91, 173–175, 177, 178, 180, 186, 201–205, 208, 209, 211	<code>\glstreenonamechilddesc</code>	383
<code>\glsssetattribute</code>	218–220, 222, 223, 225, 226, 228, 230, 231, 233–236, 238, 240, 242, 245, 247, 249–252, 254, 256, 259, 261, 263–271, 273, 275, 277, 279–281, 283, 285, 287–289, 291–293, 295–297, 299, 300, 302–305, 307, 308, 310	<code>\glstreenonamedesc</code>	382, 383
<code>\glsssetcategoryattribute</code>	101, 113, 115, 148, 154, 169, 170, 172, 187	<code>\glstreenonamesymbol</code>	383
<code>\glsssetnoexpandfield</code>	12, 13	<code>\glstreepredesc</code>	380, 382
<code>\glsssettoctitle</code>	120	<code>\glstreeprelocation</code> ...	379, 381, 383, 384
<code>\glssshortaccessdisplay</code>	159, 160	<code>\glstreesubitem</code>	379, 405
<code>\glssshortpltok</code>	198, 218–224, 226, 228, 235–240, 245, 247, 249–252, 254, 259, 261, 263– 266, 268–271, 281, 282, 285, 286, 288– 293, 295, 296, 300, 302–305, 307, 308, 310	<code>\glstreesubsubitem</code>	379, 406
<code>\glssshortpluralaccessdisplay</code>	160	<code>\glstreesymbol</code>	379, 381
<code>\glssshorttok</code> .	113, 114, 196–198, 218–222, 224, 226, 228, 233–239, 242, 245, 247, 249–252, 254, 256, 259, 261, 263–271, 273, 275, 280, 282, 285, 286, 288–293, 295, 296, 298, 300, 302–305, 307, 308, 310	<code>\GLstrLetField</code>	34
<code>\glsssubentryitem</code>	138, 367–377, 379, 381, 383, 395, 405	<code>\glstype</code>	63, 66, 77–81, 148, 201–211
<code>\glssymbolaccessdisplay</code>	157	<code>\glsunset</code>	51, 63, 104, 105, 149
<code>\glssymbolpluralaccessdisplay</code>	158	<code>\glssuserdescription</code> ...	285, 286, 289, 292
<code>\glstarget</code>	138, 139, 367–377, 379, 381, 383, 394, 395, 405, 406	<code>\glswrite</code>	47, 116
		<code>\glswriteentry</code>	8, 9
		<code>\Glsxtr</code>	56
		<code>\glxtr</code>	56
		<code>\glxtr@@do@wrglossary</code>	8, 10, 12, 13
		<code>\glxtr@addloclistfield</code>	14
		<code>\glxtr@addunused</code>	50, 51
		<code>\glxtr@applyabbrvfmt</code>	211
		<code>\glxtr@applyabbrvstyle</code> ...	194, 196, 215
		<code>\glxtr@counterrecord</code>	137
		<code>\glxtr@do@alsoindex@wrglossary</code> ...	14
		<code>\glxtr@doooption</code>	5, 16, 17, 23, 24, 26
		<code>\glxtr@fields</code>	135, 136
		<code>\glxtr@headentry@p</code>	94, 95
		<code>\glxtr@hyperoutsidefalse</code>	65
		<code>\glxtr@hyperoutsidettrue</code>	65
		<code>\glxtr@ifnextpunc</code>	193

<code>\glxtr@ifpunctoken</code>	193	<code>\glxtrAltTreeIndent</code>	384, 385
<code>\glxtr@inc@linkcount</code>	66, 153	<code>\glxtralttreeInit</code>	394, 400, 401
<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 23, 27	<code>\glxtrAltTreePar</code>	384
<code>\glxtr@indexonly@saveentrycounter</code>	13, 14, 27	<code>\glxtrAltTreeSetHangIndent</code>	384, 394
<code>\glxtr@keylist</code>	55, 56	<code>\glxtrAltTreeSetSubHangIndent</code>	395
<code>\glxtr@label</code>	407	<code>\glxtralttreeSubSymbolDescLocation</code>	395
<code>\glxtr@langtag</code>	136	<code>\glxtralttreeSymbolDescLocation</code>	384, 395
<code>\glxtr@linkprefix</code>	136	<code>\glxtrassignfieldfont</code>	69–76
<code>\glxtr@loaddialect</code>	330, 365	<code>\glxtrautoindex</code>	182
<code>\glxtr@makeglossaries</code>	116	<code>\glxtrautoindexassignsort</code>	182
<code>\glxtr@newabbreviation</code>	115, 196	<code>\glxtrautoindexentry</code>	182
<code>\glxtr@next</code>	193	<code>\glxtrbibaddress</code>	332
<code>\glxtr@org@odo@wrglossary</code>	27	<code>\glxtrbibauthor</code>	332
<code>\glxtr@org@dohyperlink</code>	87	<code>\glxtrbibbooktitle</code>	332
<code>\glxtr@org@getgroupitle</code>	125	<code>\glxtrbibchapter</code>	332
<code>\glxtr@org@newignoredglossary</code>	40	<code>\glxtrbibedition</code>	332
<code>\glxtr@org@makenoidxglossaries</code>	52	<code>\glxtrbibhowpublished</code>	332
<code>\glxtr@pluralsuffixes</code>	135, 136	<code>\glxtrbibinstitution</code>	332
<code>\glxtr@process</code>	140, 141	<code>\glxtrbibjournal</code>	332
<code>\glxtr@provideignoredglossary</code>	42	<code>\glxtrbibmonth</code>	332
<code>\glxtr@punctlist</code>	192, 193	<code>\glxtrbibnote</code>	332
<code>\glxtr@record</code>	11, 136	<code>\glxtrbibnumber</code>	332
<code>\glxtr@record@nr</code>	13	<code>\glxtrbiborganization</code>	332
<code>\glxtr@recordsee</code>	12	<code>\glxtrbibpages</code>	332
<code>\glxtr@resource</code>	134, 135	<code>\glxtrbibpublisher</code>	332
<code>\glxtr@s@newignoredglossary</code>	40	<code>\glxtrbibschool</code>	332
<code>\glxtr@s@provideignoredglossary</code>	42	<code>\glxtrbibseries</code>	332
<code>\glxtr@saveentrycounter</code>	8, 9, 12, 84	<code>\glxtrbibtitle</code>	332
<code>\glxtr@setaccessdisplay</code>	180	<code>\glxtrbibtype</code>	333
<code>\glxtr@setbookindexmark</code>	407	<code>\glxtrbibvolume</code>	333
<code>\glxtr@setup@record</code>	13, 14, 26, 27	<code>\glxtrbookindexatendgroup</code>	405
<code>\glxtr@shortcutsval</code>	136	<code>\glxtrbookindexatsubendgroup</code>	405
<code>\glxtr@texencoding</code>	136	<code>\glxtrbookindexatsubsubendgroup</code>	405
<code>\glxtr@undefaction@nr</code>	7	<code>\glxtrbookindexbetween</code>	405
<code>\glxtr@undefaction@val</code>	7	<code>\glxtrbookindexbookmark</code>	406
<code>\glxtr@usesee</code>	45	<code>\glxtrbookindexcols</code>	404
<code>\glxtr@warnonexistsordo</code>	7, 13, 14, 44, 45	<code>\glxtrbookindexcolspread</code>	404
<code>\glxtr@writefields</code>	134	<code>\glxtrbookindexfirstmarkfmt</code>	407
<code>\glxtrabbreviationfont</code>	61	<code>\glxtrbookindexformatheader</code>	406
<code>\glxtrabbrvfootnote</code>	222–224, 245–247, 259–261, 280–282	<code>\glxtrbookindexgroupskip</code>	406
<code>\glxtrabbrvpluralsuffix</code>	136, 200, 218, 220, 223, 225, 226, 228, 231, 234, 249, 263, 285, 295, 297, 300, 305, 308	<code>\glxtrbookindexlastmarkfmt</code>	407
<code>\glxtrabbrvtype</code>	17, 18, 198	<code>\glxtrbookindexmulticolenv</code>	404
<code>\glxtractivatenopost</code>	120	<code>\glxtrbookindexname</code>	402, 405
<code>\glxtraddallcrossrefs</code>	50	<code>\glxtrbookindexparentchildsep</code>	402, 404
<code>\glxtralias</code>	84	<code>\glxtrbookindexparentsubchildsep</code>	404, 405
		<code>\glxtrbookindexprelocation</code>	402, 405
		<code>\glxtrbookindexsubbetween</code>	405

<code>\glstrbookindexsubname</code>	406	<code>\GlsXtrFormatLocationList</code>	59, 61, 391–393
<code>\glstrbookindexsubprelocation</code>	406	<code>\GLSxtrfull</code>	18, 19, 321, 322
<code>\glstrbookindexsubsubbetween</code>	405	<code>\Glsxtrfull</code>	18, 19, 322
<code>\glstrbookindexthepage</code>	407	<code>\glstrfull</code>	18, 19, 321, 322
<code>\glstrcat</code>	55, 56	<code>\Glsxtrfullformat</code>	
<code>\glstrchecknohyperfirst</code>	70–72	. 199, 213, 215, 216, 218, 221, 223, 225,	
<code>\glstrcombingdiacriticIIrules</code> .	336	227, 229, 232, 235, 237, 239, 240, 243,	
<code>\glstrcombingdiacriticIIrules</code> ..	336	244, 246, 248, 250, 251, 253, 255, 257,	
<code>\glstrcombingdiacriticIrules</code> ...	336	259, 260, 262, 264, 265, 267, 269, 271,	
<code>\glstrcombingdiacriticIVrules</code> ..	336	272, 275, 276, 278, 280, 281, 283, 286,	
<code>\glstrComputeTreeIndent</code>	394, 395	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\glstrComputeTreeSubIndent</code>	395	<code>\glstrfullformat</code>	
<code>\glstrcounterprefix</code>	128 199, 213, 215, 216, 218, 220,	
<code>\glstrcurrencyrules</code>	339	223, 225, 227, 229, 232, 235, 237, 239,	
<code>\Glsxtrdefaultsubsequentfmt</code> ...	214, 216	240, 243–245, 247, 249, 251, 253, 254,	
<code>\glstrdefaultsubsequentfmt</code> ...	214, 215	257–259, 261, 263, 265, 267, 269, 271,	
<code>\Glsxtrdefaultsubsequentplfmt</code> .	214, 216	272, 274, 276, 278, 280, 281, 283, 286,	
<code>\glstrdefaultsubsequentplfmt</code> .	214, 215	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\GlsXtrDefineAbbreviationShortcuts</code> .	20	<code>\GLSxtrfullpl</code>	18, 19, 322, 323
<code>\GlsXtrDefineAcShortcuts</code>	20, 21	<code>\Glsxtrfullpl</code>	18, 19, 323
<code>\GlsXtrDefineOtherShortcuts</code>	20, 21	<code>\glstrfullpl</code>	18, 19, 322
<code>\glstrdetoklocation</code>	147	<code>\Glsxtrfullplformat</code>	
<code>\glstrdiscardperiod</code>	190	. 199, 213, 215, 216, 219, 221, 223, 225,	
<code>\glstrdisplayendloc</code>	127	227, 229, 232, 235, 237, 239, 240, 243,	
<code>\glstrdisplayendloohook</code>	128	245, 246, 248, 250, 251, 253, 255, 257,	
<code>\glstrdisplaysingleloc</code>	127, 128	259, 260, 262, 264, 265, 267, 269, 271,	
<code>\glstrdisplaystartloc</code>	127	272, 275, 276, 278, 280, 281, 283, 286,	
<code>\glstrdoautoindexname</code>	85, 86, 179	287, 289, 293, 295, 298, 301, 303, 306, 309	
<code>\glstrdopostpunc</code>	224, 247, 261, 282	<code>\glstrfullplformat</code>	
<code>\glstrdowrglossaryhook</code>	85, 86 213, 215, 216, 218, 221, 223, 225,	
<code>\GlsXtrDualField</code>	332	227, 229, 232, 235, 237, 239, 240, 243,	
<code>\glstremsuffix</code>	263, 265, 267,	244, 246, 247, 249, 251, 253, 255, 257,	
268, 270, 271, 273, 275, 277, 279, 281, 283		258, 260, 262, 263, 265, 267, 269, 271,	
<code>\GlsXtrEnableEntryCounting</code>	113	272, 274, 276, 278, 280, 281, 283, 286,	
<code>\GlsXtrEnableEntryUnitCounting</code>	101	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\GlsXtrEnableOnTheFly</code>	54, 57	<code>\glstrfullsep</code>	
<code>\glstrendfor</code>	32	. 199, 218–221, 224–227, 229, 230, 232,	
<code>\glstrfieldlistgadd</code>	137	235–244, 246, 248–258, 260, 262–274,	
<code>\glstrfieldtitlecase</code>	174–177, 181	276–279, 282–284, 294–299, 302–305, 309	
<code>\glstrfieldtitlecasecs</code>	173	<code>\glstrgenabbrvfmt</code>	61
<code>\glstrfieldxifinlist</code>	142	<code>\glstrgeneralpuncIIrules</code>	339
<code>\glstrfirstscfont</code>	234	<code>\glstrgeneralpuncIrules</code>	339
<code>\glstrfirstsmfont</code>	249	<code>\glstrgetgrouptitle</code>	126, 406
<code>\GlsXtrFmtDefaultOptions</code>	30	<code>\glstrgroupfield</code>	143
<code>\glstrfmtdisplay</code>	30	<code>\Glsxtrheadfirst</code>	313
<code>\GlsXtrFmtField</code>	30, 31	<code>\glstrheadfirst</code>	313
<code>\glstrfootnotename</code>		<code>\Glsxtrheadfirstplural</code>	313
.... 222, 224, 245, 247, 259, 261, 280, 282		<code>\glstrheadfirstplural</code>	313
<code>\GlsXtrForeignTextField</code>	36	<code>\Glsxtrheadfull</code>	313

<code>\glstrheadfull</code>	313	<code>\glstrinlinefullformat</code>	
<code>\Glsxtrheadfullpl</code>	313	199, 201, 202, 215,
<code>\glstrheadfullpl</code>	313		216, 223, 225, 227, 228, 230, 232, 238,
<code>\Glsxtrheadlong</code>	313		240–242, 244, 246, 248, 253–256, 258,
<code>\glstrheadlong</code>	313		260, 262, 270, 272–274, 276, 277, 279,
<code>\Glsxtrheadlongpl</code>	313		282, 283, 287, 290, 297, 301, 306, 309, 328
<code>\glstrheadlongpl</code>	313	<code>\Glsxtrinlinefullplformat</code>	
<code>\Glsxtrheadname</code>	313	200, 203, 215, 216,
<code>\glstrheadname</code>	138, 313		224, 226, 227, 229, 230, 232, 239–241,
<code>\Glsxtrheadplural</code>	313		243, 244, 246, 248, 253–255, 257, 258,
<code>\glstrheadplural</code>	313		260, 262, 270, 272–274, 276, 278, 279,
<code>\Glsxtrheadshort</code>	313		282, 284, 287, 290, 298, 301, 306, 309, 329
<code>\glstrheadshort</code>	313	<code>\glstrinlinefullplformat</code>	
<code>\Glsxtrheadshortpl</code>	313	199, 200, 202, 204, 215, 216,
<code>\glstrheadshortpl</code>	313		224, 225, 227, 229, 230, 232, 238, 240–
<code>\Glsxtrheadtext</code>	313		242, 244, 246, 248, 253–255, 257, 258,
<code>\glstrheadtext</code>	313		260, 262, 270, 272–274, 276, 278, 279,
<code>\glstrhyperlink</code>	23, 24, 88		282, 283, 287, 290, 298, 301, 306, 309, 329
<code>\glstrhyphensuffix</code>	295, 303	<code>\glstrinsertinsidefalse</code>	217
<code>\glstrtrifcounttrigger</code>	104, 105	<code>\GlsXtrInternalLocationHyperlink</code>	24, 129
<code>\glstrtrifcustomdiscardperiod</code>	190	<code>\glstrLatinA</code>	341–346
<code>\glstrtrifemptyglossary</code>	127, 133, 140	<code>\glstrLatinAELigature</code>	343, 345, 346
<code>\GlsXtrIfFieldCmpNum</code>	33	<code>\glstrLatinE</code>	341–346
<code>\GlsXtrIfFieldEqNum</code>	138	<code>\glstrLatinEszettSs</code>	342–344, 346
<code>\GlsXtrIfFieldNonZero</code>	331	<code>\glstrLatinEszettSz</code>	343, 345
<code>\glstrtrifhasfield</code> ..	35, 36, 84, 331, 332, 402	<code>\glstrLatinEth</code>	342–345
<code>\glstrtrifhyphenstart</code>		<code>\glstrLatinH</code>	341–346
.....	294, 296, 299, 302, 304, 305	<code>\glstrLatinI</code>	341–346
<code>\glstrtrifindexing</code>	85	<code>\glstrLatinInsularG</code>	345
<code>\glstrtrifinmark</code>	68, 95–97, 312, 313	<code>\glstrLatinK</code>	341–346
<code>\glstrtrifnextpunc</code>	193	<code>\glstrLatinL</code>	341–346
<code>\glstrtrifperiod</code>	190, 192	<code>\glstrLatinM</code>	341–346
<code>\glstrtrifrecordtrigger</code>	149–151	<code>\glstrLatinN</code>	341–346
<code>\glstrtrifwasfirstuse</code>		<code>\glstrLatinO</code>	341–346
.....	69–72, 77–80, 83, 115,	<code>\glstrLatinOELigature</code>	343, 345, 346
	191, 201, 204–211, 224, 225, 247, 261,	<code>\glstrLatinP</code>	341–346
	282, 283, 286, 288–290, 300, 302, 305, 307	<code>\glstrLatinS</code>	341–346
<code>\glstrinclinckcounter</code>	153	<code>\glstrLatinT</code>	341–346
<code>\glstrindexaliased</code>	84, 85	<code>\glstrLatinThorn</code>	345
<code>\glstrindexseealso</code>	49, 50	<code>\glstrLatinX</code>	341–346
<code>\glstrtrinithyperoutside</code>	66	<code>\glstrlocationhyperlink</code>	128
<code>\glstrtrinitwrgloss</code>	66, 148	<code>\glstrlocrangefmt</code>	127, 128
<code>\glstrtrinitwrglossbeforefalse</code>	64	<code>\GLSxtrlong</code>	18, 19, 320
<code>\glstrtrinitwrglossbeforetrue</code>	64	<code>\Glsxtrlong</code>	18, 19, 321
<code>\Glsxtrinlinefullformat</code>	200, 201, 215,	<code>\glstrlong</code>	18, 19, 320
	216, 224, 226, 227, 229, 230, 232, 239–	<code>\glstrlonghyphen</code>	301
	242, 244, 246, 248, 253–255, 257, 258,	<code>\glstrlonghyphennoshort</code>	297, 298
	260, 262, 270, 272–274, 276, 278, 279,	<code>\glstrlonghyphenshort</code>	295
	282, 283, 287, 290, 298, 301, 306, 309, 329	<code>\glstrlongnoshortdescname</code>	231, 278, 297

<code>\glxstrlongnoshortname</code>	<code>\GlsXtrNoGlsWarningNoOut</code>	134
..... 233, 241, 256, 273, 275, 298	<code>\GlsXtrNoGlsWarningTail</code>	134
<code>\GLSxtrlongpl</code>	<code>\glxstrnopostpunc</code>	121
..... 18, 19, 320, 321	<code>\glxstronlydescname</code>	309
<code>\Glsxtrlongpl</code>	<code>\glxstronlydescsort</code>	309
..... 18, 19, 320	<code>\glxstronlyname</code>	308
<code>\glxstrlongshortdescname</code>	<code>\glxstronlysuffix</code>	308
..... 219, 236, 250, 264, 266, 296, 301	<code>\glxstrorg@ifKV@glslink@hyper</code>	62
<code>\glxstrlongshortdescsort</code>	<code>\glxstrorglong</code>	196, 219, 297
..... 219, 236, 250, 264, 266, 291, 296, 302	<code>\glxstrorgshort</code>	196, 219
<code>\glxstrlongshortname</code>	<code>\GLSxtrp</code>	94
..... 218, 234, 249, 263, 265, 285, 286, 295, 300	<code>\Glsxtrp</code>	94
<code>\glxstrlongshortuserdescname</code> ..	<code>\glxstrp</code>	93, 95, 96
..... 288, 291	<code>\glxstrparen</code>	191,
<code>\glxstrmarkhook</code> 199, 218–221, 224–227, 229, 230, 232,	
..... 311 235–244, 246, 248–258, 260, 262–274,	
<code>\glxstrMathItalicAlpha</code> 276–279, 282–284, 294–299, 302–305, 309	
..... 350, 351, 354, 355	<code>\Glsxtrpl</code>	56
<code>\glxstrMathItalicBeta</code> ..	<code>\glxtrpl</code>	56
..... 350, 351, 354, 355	<code>\glxtrpostdescription</code> .	121, 172, 189, 378
<code>\glxstrMathItalicChi</code>	<code>\glxtrposthyphenlong</code>	305, 307
..... 351, 355, 356	<code>\glxtrposthyphenshort</code>	300, 302
<code>\glxstrMathItalicDelta</code> .	<code>\glxtrposthyphensubsequent</code>	
..... 350, 351, 354, 355 300, 302, 305, 307	
<code>\glxstrMathItalicEpsilon</code> 350, 351, 354, 355	<code>\glxtrpostlink</code>	190
<code>\glxstrMathItalicEta</code> ...	<code>\glxtrpostlinkendsentence</code>	190
..... 350, 351, 354, 355	<code>\glxtrpostlinkhook</code>	189
<code>\glxstrMathItalicGamma</code> .	<code>\glxtrpostlocalreset</code>	100, 102, 110
..... 350, 351, 354, 355	<code>\glxtrpostlocalunset</code>	100, 102, 110
<code>\glxstrMathItalicIota</code> ..	<code>\glxtrpostlongdescription</code>	40
..... 350, 351, 354, 355	<code>\glxtrpostnamehook</code>	176–179, 181
<code>\glxstrMathItalicKappa</code> .	<code>\GlsXtrPostNewAbbreviation</code>	
..... 350, 351, 354, 355 198, 215, 216, 218–	
<code>\glxstrMathItalicLambda</code> 350, 351, 354, 355 220, 222–224, 226, 228, 229, 231, 233–	
<code>\glxstrMathItalicMu</code> 236, 238, 240, 242, 245, 247, 249–252,	
..... 350, 351, 354, 355 254, 256, 259, 261, 263–266, 268–271,	
<code>\glxstrMathItalicNu</code> 273, 275, 277, 279–282, 285, 286, 288–	
..... 350, 351, 354, 356 293, 295–298, 300, 302–305, 307, 308, 310	
<code>\glxstrMathItalicOmega</code>	<code>\glxtrpostreset</code>	100, 102, 110
..... 351, 355, 356	<code>\glxtrpostunset</code>	98, 102, 110
<code>\glxstrMathItalicOmicron</code> 350, 351, 355, 356	<code>\glxtrprelocation</code>	
<code>\glxstrMathItalicPhi</code> 368, 370, 372, 374, 376, 379, 402	
..... 351, 355, 356	<code>\glxtrprotectlinks</code>	87–89
<code>\glxstrMathItalicPi</code>	<code>\GlsXtrRecordCounter</code>	11
..... 350, 351, 355, 356	<code>\glxtrrecordtriggervalue</code>	148
<code>\glxstrMathItalicPsi</code>	<code>\GlsXtrRecordWarning</code>	134
..... 351, 355, 356	<code>\glxtrregularfont</code>	61, 69
<code>\glxstrMathItalicRho</code> ...	<code>\glxtrresourcecount</code>	135
..... 350, 351, 355, 356	<code>\glxtrresourcefile</code>	135
<code>\glxstrMathItalicSigma</code>		
..... 351, 355, 356		
<code>\glxstrMathItalicTau</code>		
..... 351, 355, 356		
<code>\glxstrMathItalicTheta</code> .		
..... 350, 351, 354, 355		
<code>\glxstrMathItalicUpsilon</code> ...		
..... 351, 355, 356		
<code>\glxstrMathItalicXi</code>		
..... 350, 351, 355, 356		
<code>\glxstrMathItalicZeta</code> ..		
..... 350, 351, 354, 355		
<code>\glxstrnewabbrevpresetkeyhook</code>		197
<code>\glxstrnewnumber</code>		19
<code>\glxstrnewsymbol</code>		19
<code>\glxstrNoGlossaryWarning</code>		14, 21, 129
<code>\GlsXtrNoGlsWarningAutoMake</code>		133
<code>\GlsXtrNoGlsWarningBuildInfo</code>		133
<code>\GlsXtrNoGlsWarningCheckFile</code>		133
<code>\GlsXtrNoGlsWarningEmptyMain</code>		133
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...		133
<code>\GlsXtrNoGlsWarningEmptyStart</code>		133
<code>\GlsXtrNoGlsWarningHead</code>		133
<code>\GlsXtrNoGlsWarningMisMatch</code>		133

\glxtrresourceinit	134	\glxtrsupplocationurl	129
\glxtrrestoremarkhook	311	\glxtrtagfont	188
\glxtrrestorepostpunc	121	\Glsxtrtitlefirst	312, 313, 326
\glxtrscfont	234	\glxtrtitlefirst	312, 313, 326
\glxtrscsuffix		\Glsxtrtitlefirstplural	312–314, 327
.... 235, 237, 238, 240, 242, 243, 245, 247		\glxtrtitlefirstplural	312, 313, 327
\GlsXtrSetActualChar	185	\Glsxtrtitlefull	312–314, 329
\glxtrsetaliasnoindex	13, 14, 84, 85	\glxtrtitlefull	312–314, 328
\GlsXtrSetEncapChar	185	\Glsxtrtitlefullpl	312–314, 329
\GlsXtrSetEscChar	185	\glxtrtitlefullpl	312–314, 329
\glxtrsetfieldifexists	34, 35	\Glsxtrtitlelong	312–314, 327, 328
\GlsXtrSetLevelChar	185	\glxtrtitlelong	312–314, 327
\glxtrsetpopts	93	\Glsxtrtitlelongpl	312–314, 328
\glxtrsetupfulldefs		\glxtrtitlelongpl	312–314, 328
..... 201–204, 225, 247, 261, 283		\Glsxtrtitlename	312, 313, 325
\GLSxtrshort	18, 19, 97, 314, 315	\glxtrtitlename	312, 313, 324
\Glsxtrshort	18, 19, 315	\glxtrtitleorpdforheading	
\glxtrshort	18, 19, 314 25, 138, 139, 312, 313	
\glxtrshortdescname ...	228, 239, 254, 271	\Glsxtrtitleplural	312, 313, 326
\glxtrshorthyphen	306	\glxtrtitleplural	312, 313, 325, 326
\glxtrshorthyphenlong	303	\Glsxtrtitleshort	312, 313, 324
\glxtrshortlongdescname		\glxtrtitleshort	312, 313, 323
..... 221, 237, 252, 267, 269, 304, 307		\Glsxtrtitleshortpl	312, 313, 324
\glxtrshortlongdescsort		\glxtrtitleshortpl	312, 313, 323, 324
.... 221, 237, 252, 267, 269, 293, 304, 307		\Glsxtrtitletext	312, 313, 325
\glxtrshortlongname		\glxtrtitletext	312, 313, 325
..... 220, 236, 250, 266, 268, 289, 292, 303, 305		\GlsXtrTotalRecordCount	147
\glxtrshortlonguserdescname ..	290, 293	\glxtrtreetopindent	385, 393
\glxtrshortnolongname .	226, 238, 252, 270	\glxtrundefaction ..	7, 13, 14, 28, 41, 43–45
\GLSxtrshortpl	18, 19, 314, 315	\glxtrundeftag	27, 123, 124
\Glsxtrshortpl	18, 19, 315	\GlsXtrUnknownDialectWarning	37
\glxtrshortpl	18, 19, 314, 315	\glxtrunsrtdo	141, 142
\glxtrsmfont	248	\glxtrUpAlpha	349, 350, 354, 355
\glxtrsmsuffix		\glxtrUpBeta	349, 350, 354, 355
.... 249, 251, 252, 254, 256, 257, 259, 261		\glxtrUpChi	349, 350, 355, 356
\GlsXtrStandaloneGlossaryType .	138, 139	\glxtrUpDelta	349, 350, 354, 355
\GlsXtrStandaloneSubEntryItem .	138, 139	\glxtrUpDigamma	349, 350, 354
\Glsxtrsubsequentfmt		\glxtrUpEpsilon	349, 350, 354, 355
..... 212, 216, 232, 242, 244,		\glxtrUpEta	349, 350, 354, 355
256, 258, 274, 275, 277, 279, 297, 301, 306		\glxtrUpGamma	349, 350, 354, 355
\glxtrsubsequentfmt		\glxtrUpIota	349, 350, 354, 355
..... 212, 215, 231, 242, 243,		\glxtrUpKappa	349, 350, 354, 355
256, 258, 274, 275, 277, 279, 297, 300, 306		\glxtrUpLambda	349, 350, 354, 355
\Glsxtrsubsequentplfmt		\glxtrUpMu	349, 350, 354, 356
..... 212, 216, 232, 242, 244,		\glxtrUpNu	349, 350, 354, 356
256, 258, 274, 275, 277, 279, 297, 301, 306		\glxtrUpOmega	349, 350, 355, 356
\glxtrsubsequentplfmt		\glxtrUpOmicron	349, 350, 355, 356
..... 212, 215, 231, 242, 244,		\glxtrUpPhi	349, 350, 355, 356
256, 258, 274, 275, 277, 279, 297, 300, 306		\glxtrUpPi	349, 350, 355, 356

<code>\glxtrUpPsi</code>	349, 350, 355, 356	125, 126, 138, 143, 145, 147, 152, 174,
<code>\glxtrUpRho</code>	349, 350, 355, 356	175, 177–180, 190, 194, 211, 215, 370–377
<code>\glxtrUpSigma</code>	349, 350, 355, 356	<code>\ifcsstring</code>
<code>\glxtrUpTau</code>	349, 350, 355, 356	28, 169, 214
<code>\glxtrUpTheta</code>	349, 350, 354, 355	<code>\ifcsundef</code>
<code>\glxtrUpUpsilon</code>	349, 350, 355, 356	.. 33, 36–39, 41–43, 53, 57, 59, 89, 102,
<code>\glxtrUpXi</code>	349, 350, 355, 356	107–111, 125, 126, 129, 142, 153, 154,
<code>\glxtrUpZeta</code>	349, 350, 354, 355	169, 214, 216, 217, 367, 385, 386, 394, 407
<code>\GlsXtrUseAbbrStyleFmts</code>		<code>\ifcsvoid</code>
.....	220, 222, 228, 230, 231,	49, 168
	233, 234, 236, 238, 241, 250, 252, 255,	<code>\ifdef</code>
	264, 266, 268, 270, 273, 277, 280, 288,	14, 19, 26, 30, 36, 38, 44,
	291, 293, 296, 297, 299, 302, 304, 307, 310	47, 48, 51, 57, 58, 83, 87, 88, 94, 96, 97,
<code>\GlsXtrUseAbbrStyleSetup</code>	228–230, 233,	119, 123, 124, 136, 145, 171, 172, 185,
	234, 241, 243, 255, 257, 272, 276, 277, 280	188, 284, 312, 323–329, 331, 364, 365,
<code>\glxtruserfield</code>	284	367–370, 378–384, 395–401, 403, 406, 407
<code>\glxtruserparen</code>	285–293	<code>\ifdefempty</code>
<code>\glxtrusersuffix</code>	285, 287, 289, 292	7–9, 33, 36, 37, 41–43, 45,
<code>\glxtruseseealsoformat</code>	46, 47	46, 66, 68, 101, 113, 116, 119, 128, 140,
<code>\glxtruseseeformat</code>	46	143, 147, 148, 162–164, 187, 195, 211, 404
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	194, 216	<code>\ifdefequal</code>
<code>\GlsXtrWarning</code>	55, 56	35, 52, 133, 143, 180
<code>\glxtrword</code>	195	<code>\ifdefstring</code>
<code>\glxtrwordsep</code>	195, 294, 296, 299, 302, 304, 305	6, 23, 24, 35, 182, 187, 403
<code>\glxtrwrglossmark</code>	24	<code>\ifdefvoid</code> ..
		45, 48–50, 88, 107, 125, 129, 144
		<code>\ifdim</code>
		58, 115, 385–393
		<code>\IfFileExists</code>
		22, 129, 133, 134, 136, 365, 366
		<code>\ifglossaryexists</code>
		45
		<code>\ifglssacronym</code>
		18, 133
		<code>\ifglssacrshortcuts</code>
		20
		<code>\ifglssautomake</code>
		119, 133, 136
		<code>\ifglssentrycounter</code>
		38
		<code>\ifglssentryexists</code>
		9,
		27, 43, 44, 55, 56, 59, 69, 144, 169, 189
		<code>\ifglssfieldeq</code>
		168
		<code>\ifglsshasfield</code>
		30, 31, 284
		<code>\ifglsshaslong</code>
		106, 107, 151, 152
		<code>\ifglsshasparent</code> .
		138, 139, 141, 144, 387–389
		<code>\ifglsshasshort</code>
		46, 61, 69
		<code>\ifglsshasymbol</code>
		191, 381, 382, 384
		<code>\ifglssindexonlyfirst</code>
		85
		<code>\ifglssnogroupskip</code>
	
		368, 370–378, 380, 381, 383, 395, 403
		<code>\ifglssnonnumberlist</code>
		61
		<code>\ifglssnopostdot</code>
		16, 121
		<code>\ifglssanitizesort</code>
		119
		<code>\ifglssubentrycounter</code>
		38
		<code>\ifglssused</code>
		50, 51, 83, 85,
		103, 112, 115, 212, 387, 388, 390, 391, 393
		<code>\ifglssxindy</code>
		129, 131
		<code>\ifglsxtr@hyperoutside</code>
		67
		<code>\ifglsxtr@trinitwrglossbefore</code>
		64, 67, 148, 149
		<code>\ifglsxtr@insertinside</code>
		. 204–211, 213, 214, 218–221, 223–233,

H

<code>\hangindent</code> .	381, 383, 384, 394–396, 400, 401
<code>\hbox</code>	367
<code>\hfill</code>	367
<code>\href</code>	88
<code>\hsize</code>	58
<code>\hss</code>	367
<code>\hyperlink</code>	88
<code>\hyperpage</code>	182
<code>\hyperref</code>	87, 129, 331
<code>hyperref package</code>	89, 181, 310, 323

I

<code>\if</code>	54
<code>\if@glxtr@autoseeindex</code>	26, 45, 48
<code>\if@glxtr@format@override</code>	182
<code>\if@glxtr@docdefrestricted</code>	52
<code>\if@glxtr@indexcrossrefs</code>	15, 50
<code>\ifblank</code>	29, 55, 56, 116
<code>\ifbool</code>	28, 39
<code>\ifcase</code> ..	7, 13, 20, 21, 24, 53, 64, 122, 379, 405
<code>\ifcsdef</code>	28,
	38–43, 65, 67, 81, 82, 93–97, 107, 120,

235, 237–251, 253–265, 267, 269–284, 286–290, 292–294, 296, 299–302, 305–309	101–103, 110, 111, 113–117, 119–126, 134, 136, 137, 140, 141, 143, 148, 153, 163, 174–184, 187, 188, 193–197, 201– 211, 214–216, 225, 247, 261, 283, 310– 314, 364, 365, 379, 384, 386, 397, 404–407
\ifHy@hyperindex 181	\letabbreviationstyle .. 224, 226, 227, 229, 233, 234, 239, 241, 254, 255, 271, 272
\ifinlist 99	\letcs 28, 32, 33, 45, 46, 50, 65, 67, 81, 123–125, 143, 144, 174–181, 407
\ifinlistcs 32, 53	\levelchar 185
\ifinner 25	\listadd 107
\ifKV@glslink@hyper 62, 66–68	\listbreak 187
\ifKV@glslink@local 63, 149	\listcsadd 31
\ifKV@glslink@noindex .. 8, 9, 12, 30, 84, 85	\listcseadd 31, 108
\ifmmode 25	\listcsgadd 31, 53
\ifnum 15, 33, 51, 52, 103, 104, 111, 112, 125, 135, 148, 381, 383, 394, 395	\listcsxadd 31, 108
\ifstrempy 138, 145, 154	\listxadd 99
\ifstrequal 17, 22	\loadglsentries 53, 131
\ifthenelse 133, 190	\long 40
\IfTrackedDialectHasMapping 36	
\IfTrackedLanguageFileExists 330	
\ifundef 23, 32, 33, 36, 116, 187, 188, 364	
\ifx 8–11, 47, 57, 59, 67, 116, 120, 122, 127, 128, 137, 182, 183, 186, 193, 195, 197, 293, 294, 401	
\immediate ... 52, 103, 111, 112, 129, 130, 136	
\index 182	
\indexspace . 368, 380–384, 395–401, 403, 406	
\input 330	
\inputencodingname 136	
\InputIfFileExists 51	
\istfilename 116	
\item 131, 132, 367–370, 379, 380, 396, 397	
J	
\jobname 51, 52, 129, 131–136	
K	
\key@ifundefined . 12, 13, 28, 29, 81, 140, 143	
\KV@glslink@hyperfalse 70, 83, 89	
\KV@glslink@hypertrue 89	
\KV@glslink@noindexfalse 83, 84	
\KV@glslink@noindextrue 84, 89	
L	
\L 346, 349	
\l 349	
\label 23	
\LaTeX 131, 132	
\leaders 367	
\leavevmode 40, 66	
\let 5, 7–14, 16, 18, 19, 25– 27, 30, 32, 35–37, 39, 40, 52–54, 57–60, 62–64, 66–80, 82–87, 89, 90, 93, 98, 99,	
	M
	\MakeAcronymsAbbreviations 115
	\makeatletter 51, 129, 134, 184
	\makeatother 184
	\makebox 367, 394, 395
	\makefirstuc 188
	makeglossaries 122
	\makeglossaries 116, 130, 132, 133, 137
	\makeglossary 117
	makeindex 408
	makeindex 14, 116
	\makenoidxglossaries 132
	\MakeTextUppercase 313
	\MakeUppercase 312, 313
	\marginpar 25
	\markboth 311
	\markright 311
	\mathit 334
	\mathrm 333–335
	\maxdimen 58
	\mbox 369, 370, 394
	\medskip 133, 142
	\MessageBreak 53, 57, 103, 112, 116, 119, 120, 134, 214, 215
	mfistuc package 187
	\mfistucMakeUppercase 70–80, 82, 91–94, 97, 106, 114, 152, 154–161, 164–167, 176, 177, 181, 202, 204, 205, 207, 209, 211–213

<code>\mfu@checkword@arg</code>	187, 188	<code>name</code>	46, 154, 164, 182, 315, 316, 324
<code>\mfu@checkword@do</code>	188	<code>plural</code>	155, 156, 165, 217, 317, 318, 325
N			
<code>\NeedsTeXFormat</code>	5, 331, 366, 402	<code>see</code>	15, 16, 26, 45, 48, 50, 53, 117
<code>\new@atom@glossaryentry</code>	52	<code>seealso</code>	16, 45, 46, 48–50, 419
<code>\new@glossaryentry</code>	53, 119	<code>short</code>	160, 167, 194
<code>\new@ifnextchar</code>		<code>shortaccess</code>	162–164
	30, 81, 82, 106, 145, 149–151, 192, 200–211	<code>shortaccessplural</code>	163
<code>\newabbr</code>	18, 19	<code>shortplural</code>	160, 167, 194
<code>\newabbreviation</code>	18, 19	<code>symbol</code>	157, 165, 166
<code>\newabbreviationhook</code>	198	<code>symbolplural</code>	157, 158, 166
<code>\newabbreviationstyle</code>	218–222,	<code>text</code>	87, 155, 165, 217, 219, 316, 317, 325
	224, 226, 228–231, 233, 234, 236–239,	<code>textaccess</code>	163
	241, 243, 245, 246, 249, 250, 252, 254–	<code>user2</code>	37
	257, 259, 261, 263, 264, 266–273, 275–	<code>\newglossarystyle</code>	404
	278, 280, 282, 285, 286, 288, 290, 291,	<code>\newif</code>	64, 181, 217
	293, 295–298, 300, 301, 303–305, 307–309	<code>\newlength</code>	384, 385
<code>\newacronym</code>	113, 114	<code>\newnum</code>	19
<code>\newacronymhook</code>	113	<code>\newrobustcmd</code> 29, 31, 32, 34, 35, 47, 48, 52,	
<code>\newacronymstyle</code>	114, 115		65, 69, 81, 82, 93, 94, 106, 121, 125, 138,
<code>\newcommand</code>	5–		139, 142, 145, 146, 149–151, 180, 187,
	13, 15–34, 36–46, 49–51, 54–57, 59–62,		188, 200–211, 293, 312, 314–323, 386–393
	64, 65, 68–70, 81, 82, 84–86, 88, 89, 93–	<code>\newsym</code>	19
	103, 106–116, 120–135, 137–145, 147–	<code>\newterm</code>	171
	162, 164–173, 179–196, 198–211, 213–	<code>\newtoks</code>	194
	217, 219–222, 226, 228, 231, 233, 234,	<code>\newwrite</code>	52, 116
	248, 249, 262, 263, 284, 285, 288, 290,	<code>\nobreak</code>	369, 370, 380, 382–384, 396–399
	294–296, 299, 300, 302, 304, 305, 307–	<code>\NoCaseChange</code>	95–97, 139, 314–323
	309, 311–332, 335–364, 366, 368, 378–	<code>\noexpand</code> 10, 11, 22, 47, 49–51, 114, 129, 130,	
	382, 384–386, 393, 394, 402, 403, 406, 407		134, 141–143, 153, 183, 184, 198, 366, 405
<code>\newcount</code>	135, 145	<code>\nofiles</code>	133
<code>\newcounter</code>	23, 152	<code>\noindent</code>	133, 382–384, 397–399, 401
<code>\newentry</code>	19	<code>\nopagebreak</code>	380, 382–384, 395–402, 406
<code>\newglossary</code>	17, 117	<code>\nopostdesc</code>	40, 55, 56, 121, 172
<code>\newglossaryentry</code>		<code>\ns@GLSxtrfull</code>	202
	... 19, 52, 53, 102, 109, 114, 171, 172, 198	<code>\ns@Glsxtrfull</code>	201
<code>\newglossaryentry options</code>		<code>\ns@glxtrfull</code>	200
<code>access</code>	163	<code>\ns@GLSxtrfullpl</code>	203
<code>alias</code>	16, 45, 48–50	<code>\ns@Glsxtrfullpl</code>	203
<code>desc</code>	158, 166	<code>\ns@glxtrfullpl</code>	202
<code>descplural</code>	159, 166	<code>\ns@GLSxtrlong</code>	207
<code>first</code>	87, 156, 165, 217, 318, 319, 326, 408	<code>\ns@Glsxtrlong</code>	206
<code>firstaccess</code>	164	<code>\ns@glxtrlong</code>	206
<code>firstplural</code> ..	156, 157, 165, 217, 319, 326, 408	<code>\ns@GLSxtrlongpl</code>	211
<code>group</code>	143	<code>\ns@Glsxtrlongpl</code>	210
<code>loclist</code>	31	<code>\ns@glxtrlongpl</code>	209
<code>long</code>	161, 167, 327	<code>\ns@GLSxtrshort</code>	205
<code>longplural</code>	161, 167, 328	<code>\ns@Glsxtrshort</code>	204
		<code>\ns@glxtrshort</code>	204
		<code>\ns@GLSxtrshortpl</code>	209

<code>\ns@Glsxtrshortpl</code>	208	<code>true</code>	20
<code>\ns@glxtrshortpl</code>	207, 208	<code>sort</code>	
<code>\null</code>	21	<code>use</code>	68
<code>\number</code>	51, 108–111, 134, 145	<code>style</code>	22
<code>\numexpr</code>	108, 109, 111	<code>stylemods</code>	22
O			
<code>\O</code>	346, 349	<code>symbols</code>	19, 172
<code>\o</code>	349	<code>undefaction</code>	43, 44
<code>\openout</code>	52	<code>error</code>	6
<code>\or</code>	7, 13, 14, 20, 21, 24, 53, 64, 379, 405	<code>warn</code>	6
<code>\org@glossaryentrynumbers</code>	59, 120	<code>xindy</code>	47, 48
<code>\org@glossarytitle</code>	120	<code>\PackageError</code>	6, 11, 22, 26, 52, 53, 57, 65, 81, 82, 85, 93, 94, 101–103, 109, 111, 113, 115–119, 126, 137, 141, 142, 145, 190, 214–217, 366, 367
<code>\org@ifKV@glslink@hyper</code>	66, 68	<code>\PackageWarning</code>	16
P			
<code>\p@gl@s@hyp@opt</code>	86	<code>\PackageWarningNoLine</code>	16
package options:		<code>\pageref</code>	23, 38
<code>abbreviations</code>	17, 18	<code>\par</code>	132– 134, 369, 370, 379, 381–384, 394–401, 403
<code>accsupp</code>	21, 154	<code>\parindent</code> ..	379, 381, 383–385, 394–401, 404
<code>acronym</code>	17	<code>\parskip</code>	379, 381, 383, 397–399, 404
<code>automake</code>	119, 131, 136	<code>\PassOptionsToPackage</code>	5, 22
<code>true</code>	136	<code>\pdfbookmark</code>	403
<code>autoseeindex</code>	26	<code>\preglossarypreamble</code>	39
<code>false</code>	26	<code>\preto</code>	84
<code>counter</code>		<code>\print@noop@unsrtglossaryunit</code> ...	12, 13
<code>wrglossary</code>	23	<code>\print@op@unsrtglossaryunit</code>	14
<code>debug</code>		<code>\printabbreviations</code>	17
<code>showtargets</code>	88	<code>\printglossaries</code>	117, 132
<code>docdef</code>	14, 15, 52, 53, 102, 109	<code>\printglossary</code>	17, 117, 132, 134
<code>atom</code>	51	<code>\printglossary options</code>	
<code>false</code>	52, 53	<code>nonumberlist</code>	61
<code>restricted</code>	15	<code>type</code>	119
<code>true</code>	51, 53	<code>\printnoidxglossaries</code>	132
<code>docdefs</code>		<code>\printnoidxglossary</code>	118, 132
<code>restricted</code>	52	<code>\printnumbers</code>	19, 172
<code>indexcounter</code>	331	<code>\printsymbols</code>	19, 172
<code>nonumberlist</code>	59	<code>\printunsrtglossary</code>	132, 134, 140
<code>nopostdot</code>	16	<code>\printunsrtglossaryentryprocesshook</code>	140, 141
<code>false</code>	16	<code>\printunsrtglossaryhandler</code>	141, 142
<code>numbers</code>	19	<code>\printunsrtglossarypredoglossary</code> ..	141
<code>postdot</code>	16	<code>\printunsrtglossaryskipentry</code>	140
<code>record</code> ..	7, 13, 52, 62, 116, 134, 201–211, 417	<code>\printunsrtglossaryunit</code>	13, 14, 142
<code>alsoindex</code>	8, 10	<code>\printunsrtglossaryunitsetup</code>	142
<code>only</code>	8, 132	<code>\ProcessOptions</code>	367
<code>shortcuts</code>	20	<code>\ProcessOptionsX</code>	24
<code>ac</code>	20	<code>\protect</code>	95–97, 164–167, 195, 199, 218–224, 226–228, 230–247, 249–261,
<code>all</code>	20		
<code>false</code>	20		
<code>none</code>	20		

263–271, 273–283, 285, 286, 288–293, 295–298, 300, 302–305, 307–310, 314–323	\rglsformat 149, 152
\protected@csedef 34, 35, 126, 385, 386	\rGLSpl 147
\protected@csxdef 35, 126, 385, 386	\rGlspl 147
\protected@edef 35, 57, 65, 87, 114, 125, 126, 142, 143, 182, 198	\rglspl 147
\protected@write 11, 12, 52, 60, 61, 116, 117, 134–137, 407	\rGLSplformat 151
\providecommand 17–19, 29, 47, 61, 82, 84, 103, 112, 116, 130, 135, 136, 331, 333–335, 367, 402	\rGlsplformat 150
\ProvidesFile 330	\rglsplformat 149, 152
\ProvidesPackage 5, 331, 366, 402	\romannumeral 385, 386, 394
Q	
\quotechar 185	
R	
\raggedright 372, 373, 376, 377, 404	\s@glxtrfmt 30
\refstepcounter 23	\s@glshyp@opt 86
\relax 7, 13–15, 18–21, 24, 26, 30, 33, 51–54, 57, 58, 60, 64, 66, 67, 86, 93, 103, 104, 111, 112, 117, 120, 121, 124, 125, 127, 134–137, 144, 148, 183, 184, 186, 188, 190, 195, 197, 293, 294, 379, 381, 383, 386–396, 400, 401, 404–407	\s@glxtr@enabtagging 187
relsize package 248	\s@glxtrfmt 29
\renewcommand 6, 7, 13–18, 20–24, 26, 39–44, 46, 51–54, 56, 57, 59–64, 81, 83–85, 87, 89, 93, 100–103, 106, 107, 109–122, 125–130, 142, 147, 171, 173–176, 178, 186–189, 199, 200, 215, 216, 218–293, 295–298, 300–311, 364, 367–384, 394–401, 404–406	\s@glxtrifhasfield 32
\renewenvironment 368, 370– 377, 379, 381, 383, 394, 397–399, 401, 404	\s@GlsXtrStartUnsetBuffering 98
\renewglossarystyle 367–377, 379–383, 394–401	\s@GlsXtrStopUnsetBuffering 99
\renewrobustcmd 68, 88	\s@printunsrtdglossary 139, 142
\RequireGlossariesExtraLang ... 330, 365	\seealsoname 46, 48
\RequirePackage ... 5, 14, 21, 22, 25, 366, 402	\seename 46
\reserved@a 193	\setabbreviationstyle 115, 219, 227
\reserved@b 193	\setacronymstyle 114, 115
\reserved@d 193	\setentrycounter 127
\RestoreAcronyms 115	\SetGenericNewAcronym 115
\rGLS 147	\setglossarystyle 23, 120, 367, 369, 370, 380, 382, 383, 395–401, 404
\rGls 147	\setkeys 9, 22, 26, 30, 66, 68, 85, 113, 120, 148, 196, 197
\rgls 147	\setlength 58, 379, 381, 383, 384, 395, 397–399, 404
\rGLSformat 151	\settowidth 115, 385–394
\rGlsformat 150	\setupglossaries 5, 26
	\sfcode 16, 17, 190, 378
	\small 25
	soul package 99
	\space 6, 11, 47, 53, 54, 57, 85, 101–103, 109, 111–113, 115–118, 120, 130, 133, 134, 137, 141, 142, 145, 191, 195, 199, 219, 367, 368, 378, 381, 382, 384, 385, 394, 402
	\spacefactor 16, 17, 190, 197, 378
	\stepcounter 153
	\string . 6, 11, 12, 47, 52–54, 57, 60, 61, 67, 81, 82, 85, 93, 94, 101–103, 109, 111– 113, 115–120, 130–137, 141, 142, 145, 175, 177–180, 183, 190, 331, 335–364, 407
	\strut 367–377, 383
	\subglossentry 120, 144, 367–377, 379, 381, 383, 395, 405
	\subitem 379
	\subsubitem 379

T		U	
<code>\tablehead</code>	374–377	<code>\u</code>	331
<code>\tabletail</code>	374–377	<code>\undef</code>	13, 14, 51, 187
<code>\tabularnewline</code>	370–378	<code>\underline</code>	188
<code>\TeX</code>	131	<code>\unskip</code>	40, 51, 367
<code>\texorpdfstring</code>	30, 31, 94–97, 312, 323–329	upgreek package	334
<code>\textbf</code>	378	<code>\usepackage</code>	132, 133
textcase package	310	W	
<code>\textsc</code>	234	<code>\warn@nomakeglossaries</code>	117
<code>\textsmaller</code>	248	<code>\warn@noprintglossary</code>	117, 121
<code>\texttt</code>	25, 130–133	wrglossary (counter)	23
<code>\the</code>	114, 128, 135, 186, 198, 218–226, 228–231, 233–240, 242, 245, 247, 249–252, 254, 256, 259, 261, 263–271, 273, 275, 277– 283, 285–293, 295–300, 302–305, 307–310	<code>\write</code>	47, 103, 111, 112, 116, 129, 130, 136
<code>\theHglentrycounter</code>	8, 10, 11, 66, 68, 148	X	
<code>\theHglentrycounter</code>	8–11, 66, 68, 148	<code>\x</code>	128, 153
<code>\theindex</code>	181, 182	<code>\xcapitalisewords</code>	173
<code>\thewrglossary</code>	23	<code>\xdef</code>	120, 141
<code>\this@dialect</code>	330, 365	<code>\xifinlist</code>	108
<code>\thisgrptitle</code>	406	<code>\xifinlistcs</code>	32
<code>\toks@</code>	128, 185, 186	xindy	408
<code>\TrackedDialectClosestSubMatch</code>	36	xindy	14, 116
tracklang package	36, 136, 335	xkeyval package	5
<code>\TrackLangGetDefaultScript</code>	364	<code>\XKV@checkchoice</code>	61
<code>\TrackLangIfHasDefaultScript</code>	364	<code>\XKV@plfalse</code>	61
<code>\TrackLangRequireDialectPrefix</code>	364, 365	<code>\XKV@resa</code>	61
		<code>\XKV@sttrue</code>	61