

glossaries-extra.sty v1.33: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-07-26

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	56
1.3.4	Entry Formatting, Hyperlinks and Indexing	60
1.3.5	Entry Counting	97
1.3.6	Acronym Modifications	112
1.3.7	Indexing and Displaying Glossaries	114
1.4	Link Counting	151
1.5	Integration with glossaries-accsupp	153
1.6	Categories	166
1.7	Abbreviations	192
1.7.1	Abbreviation Styles Setup	213
1.7.2	Predefined Styles (Default Font)	216
1.7.3	Predefined Styles (Small Capitals)	233
1.7.4	Predefined Styles (Fake Small Capitals)	247
1.7.5	Predefined Styles (Emphasized)	261
1.7.6	Predefined Styles (User Parentheses Hook)	283
1.7.7	Predefined Styles (Hyphen)	292
1.7.8	Predefined Styles (No Short on First Use)	306
1.8	Using Entries in Headings	309
1.9	Multi-Lingual Support	328
1.10	glossaries-extra-bib2gls.sty	329
2	Style Adjustments (glossaries-extra-stylemods.sty)	365
2.1	Package Initialisation	365
2.2	List-Like Styles	366
2.3	Longtable Styles	369
2.4	Long Ragged Styles	371
2.5	Supertabular Styles	373
2.6	Super Ragged Styles	375
2.7	Inline Style	377
2.8	Tree Styles	377
2.9	Multicolumn Styles	395

3 bookindex style (glossary-bookindex.sty)	401
3.1 Package Initialisation and Options	401
Glossary	407
Change History	408
Index	427

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/07/26 v1.33 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

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

```
6 {%
```

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

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

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

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

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

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

```
13 }%
```

Set the defaults.

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

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

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

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

```
18 {}%
```

```
19 {%
```

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

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

```
22 {}%
```

```
23 }%
```

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

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

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

```
27 }
```

```
28 }
```

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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

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

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

`\glsxtr@recordsee` Does nothing by default.

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

`\ultnumberformat`

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

`\ultNumberFormat`

```

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

```

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

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

```

83 \newcommand*{\@glsxtr@do@record@wrglossary}[1]{%
84   \begin{group}
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  \end{group}

```

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

```

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

```

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

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

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

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

Save the label in case it's needed.

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

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

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

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

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

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings.

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

Increment associated counter.

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

Check if noindex option has been used.

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

Check if thevalue has been set.

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

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

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

Save the entry counter.

```
147         \glsxtr@saveentrycounter
```

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

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

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

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

No need to escape special characters.

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

glslink@prekeys

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

lslink@postkeys

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

lossadd@prekeys

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

ossadd@postkeys

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

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

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

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

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

```

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

```

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

```

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

```

`r@recordcounter`

```

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

```

`p@recordcounter`

```

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

```

`op@recordcounter`

```

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

```

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

```

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

```

srtglossaryunit

```

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

```

tr@setup@record Initialise.

```

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

```

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

```

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

```

addloclistfield

```

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

```

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

```

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

```

Add a key to store the group heading.

```

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

```

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

```

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

```

alsoindex

```

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

```

only

```

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

```

off

```

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

```

Now define the record package option.

```

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

```

Don't record.

```

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

```

Only record (don't index).

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

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

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

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

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

Warn about using `\printglossary`:

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

Load glossaries-extra-bib2gls:

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

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

```
303 \def\glxtr@setup@record{%
304 \renewcommand*{\@do@seeglossary}{\glxtr@do@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*{\@glsextr@docdefval}{0}
```

Need to provide conditional commands that are backward compatible:

if@glsextrdocdef

```
318 \newcommand*{\if@glsextrdocdef}{\ifnum\@glsextr@docdefval>0 }
```

lsxtrdocdeftrue

```
319 \newcommand*{\@glsextrdocdeftrue}{\def\@glsextr@docdefval{1}}
```

sxtrdocdeffalse

```
320 \newcommand*{\@glsextrdocdeffalse}{\def\@glsextr@docdefval{0}}
```

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

```
321 \define@choicekey{glossaries-extra.sty}{docdef}
```

```
322 [\@glsextr@docdefsetting\@glsextr@docdefval]%
```

```
323 {false,true,restricted}[true]%
```

```
324 {%
```

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

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

```
327   \fi
```

```
328 }
```

ocdefrestricted

```
329 \newcommand*{\if@glsextrdocdefrestricted}{\ifnum\@glsextr@docdefval=2 }
```

oifexistsorwarn

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

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

indexcrossrefs

Automatically index cross references at the end of the document

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

```
332   \if@glsextrindexcrossrefs
```

```
333   \else
```

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

```
335   \fi
```

```
336 }
```

Switch off since this can increase the build time.

```
337 \@glsextrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
338 \newcommand*{\@glsextr@autoindexcrossrefs}{\@glsextrindexcrossrefstrue}
```

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

```

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

```

iesExtraWarning Allow users to suppress warnings.

```

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

```

raWarningNoLine Allow users to suppress warnings.

```

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

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

```

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

```

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

```

postdot Shortcut for `nopostdot=false`

```

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

```

nopostdot Needs to redefine `\@glxtr@defpostpunc`

```

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

```

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

```

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

```



```

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

```

`\glsxtrabbrvtype` Glossary type for abbreviations.

```

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

```

`\bbreviationsdef` Set by abbreviations option.

```

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

```

`\bbreviationsdef`

```

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

```

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

```

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

```

```

411 \fi
412 }%

```

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

```

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

```

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

```

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

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

```

Disable this command after it's been used.

```

442 \let\GlsXtrDefineAbbreviationShortcuts\relax
443 }

```

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

```

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

```

```

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

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

```

Disable this command after it's been used.

```

470 \let\GlsXtrDefineAcShortcuts\relax
471 }

```

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

```

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

```

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

@setupshortcuts Command used to set the shortcuts option.

```

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

```

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

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

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

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

```

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

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

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

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

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

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

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

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

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

```

```

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

```

glxtr@do@style

```

587 \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.

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

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

```

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

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

`ocationHyperlink`

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

The first two arguments are always control sequences.

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

`cationhyperlink`

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

`indexcounter`

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

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

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

Only increment if the current counter is `wrglossary`.

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

```

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

```

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

```

625 \newcommand*{\@glsxtrwrglossmark}{}

```

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

```

626 \newcommand*{\@@glsxtrwrglossmark}{}
627 \AtBeginDocument{\renewcommand*{\@glsxtrwrglossmark}{\@glsxtrwrglossmark}}

```

sxtrwrglossmark Does nothing by default.

```

628 \newcommand*{\glsxtrwrglossmark}{\ensuremath{\cdot}}

```

debug Provide extra debug options.

```

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

```

Pass all other options to glossaries.

```

648 \DeclareOptionX*{%
649   \expandafter\glsxtr@doption\expandafter{\CurrentOption}}

```

Process options.

```

650 \ProcessOptionsX

```

Load glossaries if not already loaded.

```

651 \RequirePackage{glossaries}

```

Load the glossaries-accsupp package if required.

```

652 \@glsxtr@doaccsupp

```


Redefine \glspostdescription if required.

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

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

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

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

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

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

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

oindex@glossary

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

Define abbreviations glossaries if required.

```

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

```

Setup shortcuts if required.

```

703 \@glxtr@setupshortcuts

```

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

```

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

```

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

```

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

```

Now define the user command:

```

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

```

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

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

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

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

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

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

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

`\ifglsused` In the event that undefined entries should trigger a warning rather than an error, `\ifglsused` needs to be modified to check for existence.
731 `\renewcommand*{\ifglsused}[3]{%`
732 `\glsdoifexists{#1}{\ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}}%`
733 `}`

1.2 Extra Utilities

`rifemptyglossary` `\glxtrifemptyglossary{<type>}{<true>}{<false>}`

Provide command to determine if any entries have been added to the glossary (where the glossary label is provided in the first argument). The entries are stored in the comma-separated list `\gloolist@<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.)

```

734 \newcommand{\glxtrifemptyglossary}[3]{%
735   \ifcsdef{glolist@#1}%
736   {%
737     \ifcsstring{glolist@#1}{,}{#2}{#3}%
738   }%
739   {%
740     \glxtrundefaction{Glossary type '#1' doesn't exist}{}%
741     #2%
742   }%
743 }

```

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

```

744 \newcommand*{\glxtrifkeydefined}[3]{%
745   \key@ifundefined{glossentry}{#1}{#3}{#2}%
746 }

```

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

```

747 \newcommand*{\glxtrprovidestoragekey}{%
748   \@ifstar\@sglsxtr@provide@storagekey\@glxtr@provide@storagekey
749 }

```

vide@storagekey Unstarred version.

```

750 \newcommand*{\@glxtr@provide@storagekey}[3]{%
751   \key@ifundefined{glossentry}{#1}%
752   {%
753     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
754     \appto\@gl s@keymap{,}{#1}{#1}}%
755     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
756     \appto\@newglossaryentryposthook{%
757       \letcs{\@glo@tmp}{@glo@#1}%
758       \gl s@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
759     }%

```

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

```

760   \ifblank{#3}
761   {%
762   {%
763     \newcommand*{#3}[1]{\@gl s@entry@field{##1}{#1}}%
764   }%
765   }%
766   {%

```

Provide the no-link command if not already defined.

```

767   \ifblank{#3}
768   {%
769   {%

```

```

770     \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
771   }%
772 }%
773 }

```

vide@storagekey Starred version.

```

774 \newcommand*{\s@glxtr@provide@storagekey}[1]{%
775   \key@ifundefined{glossentry}{#1}%
776   {%
777     \expandafter\newcommand\expandafter*\expandafter
778     {\csname gls@assign@#1@field\endcsname}[2]{%
779       \@gls@expand@field{##1}{#1}{##2}%
780     }%
781   }%
782   {}}%
783   \glxtr@provide@addstoragekey{#1}%
784 }

```

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

```

785 \newcommand{\GlsXtrFmtField}{useri}

```

tDefaultOptions

```

786 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```

`\@glxtrfmt` Unstarred form.

```

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

```

`\s@glxtrfmt` Starred form.

```

789 \newcommand*{\s@glxtrfmt}[3][\@glxtrfmt{#1}{#2}{#3}{}
790 \new@ifnextchar[\s@glxtrfmt{#1}{#2}{#3}{}%
791 {\@glxtrfmt{#1}{#2}{#3}{}%
792 }

```

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

```

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

```

`\@glxtrfmt` Actual inner working.

```

794 \newcommand*{\@glxtrfmt}[4]{%

```

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

```

795 \begingroup
796   \def\glslabel{#2}%
797   \glsdoifexistsordo{#2}%
798   {%
799     \ifglshasfield{\GlsXtrFmtField}{#2}%
800     {%
801       \let\do@gl@link@checkfirsthyper\relax
802       \expandafter\@gl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
803       {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
804     }%
805     {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
806   }%
807   {%

```

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

```

808     \begingroup
809       \@gl@setdefault@gl@link@opts
810       \setkeys{gl@link}{\GlsXtrFmtDefaultOptions,#1}%
811       \ifKV@gl@link@noindex\else\glsadd{#2}\fi
812     \endgroup
813     \glstrfmtdisplay{@firstofone}{#3}{#4}%
814   }%
815 \endgroup
816 }

```

`\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).

```

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

```

`\glxtrentryfmt` No link or indexing.

```

818 \ifdef\texorpdfstring
819 {
820   \newcommand*{\glxtrentryfmt}[2]{%
821     \texorpdfstring{\@glxtrentryfmt{#1}{#2}}{#2}%
822   }
823 }
824 {
825   \newcommand*{\glxtrentryfmt}{\@glxtrentryfmt}
826 }

```

`@glxtrentryfmt`

```

827 \newrobustcmd*{\@glxtrentryfmt}[2]{%
828   \glsdoifexistsordo{#1}%
829   {%
830     \ifglshasfield{\GlsXtrFmtField}{#1}%

```

```

831   {%
832     \csuse{\glscurrentfieldvalue}{#2}%
833   }%
834   {#2}%
835 }%
836 {#2}%
837 }

```

`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.

```

838 \newcommand*\glxtrfieldlistadd}[3]{%
839   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
840 }

```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```

841 \newcommand*\glxtrfieldlistgadd}[3]{%
842   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
843 }

```

`trfieldlistead` Similarly but uses `\listcseadd`.

```

844 \newcommand*\glxtrfieldlistead}[3]{%
845   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
846 }

```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```

847 \newcommand*\glxtrfieldlistxadd}[3]{%
848   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
849 }

```

Now provide commands to iterate over these lists.

`fielddolistloop`

```

850 \newcommand*\glxtrfielddolistloop}[2]{%
851   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
852 }

```

`fieldforlistloop`

```

853 \newcommand*\glxtrfieldforlistloop}[3]{%
854   \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
855 }

```

List element tests:

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

```

856 \newcommand*\glxtrfieldifinlist}[5]{%
857   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
858 }

```

rfielddxifinlist Expands item.

```
859 \newcommand*\glstrfieldxifinlist}[5]{%
860   \xifinlistcs{#3}{glo@\glsetoklabel{#1}@#2}{#4}{#5}%
861 }
```

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

```
862 \newcommand*\glsxtrforcsvfield}[3]{%
863   \@glxtrifhasfield{#2}{#1}%
864   {%
865     \let\glxtrendfor\endfortrue
866     \@for\@glxtr@label:=\glscurrentfieldvalue\do
867       {\expandafter#3\expandafter{\@glxtr@label}}}%
868   }%
869 }
```

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.

```
870 \newrobustcmd{\glxtrifhasfield}{%
871   \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
872 }
```

lsxtrifhasfield Unstarred version adds grouping.

```
873 \newcommand{\@glxtrifhasfield}[4]{%
874   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
875 }
```

lsxtrifhasfield Starred version omits grouping.

```
876 \newcommand{\s@glxtrifhasfield}[4]{%
877   \letcs{\glscurrentfieldvalue}{glo@\glsetoklabel{#2}@#1}%
878   \ifundef\glscurrentfieldvalue
879     {#4}%
880     {%
881       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
882     }%
883 }
```

rIfFieldNonZero Designed for numeric fields.

```
884 \newcommand{\GlsXtrIfFieldNonZero}[4]{%
885   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
886 }
```

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

Designed for numeric fields.

```
887 \newcommand{\GlsXtrIfFieldEqNum}[5]{%
888   \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
889 }
```

XtrIfFieldCmpNum

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

Designed for numeric fields.

```
890 \newcommand{\GlsXtrIfFieldCmpNum}[6]{%
891   {%
892     \letcs{\glscurrentfieldvalue}{glo@glsetoklabel{#2}@#1}%
893     \ifundef\glscurrentfieldvalue
894     {\def\glscurrentfieldvalue{0}}%
895     {%
896       \ifdefempty\glscurrentfieldvalue
897       {\def\glscurrentfieldvalue{0}}%
898       {}}%
899     }%
900     \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
901   }%
902 }
```

sXtrIfFieldUndef

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

Just uses \ifcsundef.

```
903 \newcommand{\GlsXtrIfFieldUndef}[2]{%
904   \ifcsundef{glo@glsetoklabel{#2}@#1}%
905 }
```

\glxtrusefield

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

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

\Glsxtrusefield

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

```
909 \newcommand*{\Glsxtrusefield}[2]{%
910   \@Gls@entry@field{#1}{#2}%
911 }
```

\glxtrdeffield

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

```
912 \newcommand*{\glxtrdeffield}[2]{\csdef{glo@glsetoklabel{#1}@#2}}
```

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

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

`etfieldifexists`

```
914 \newcommand*{\glsxtrsetfieldifexists}[3]{\glsdoifexists{#1}{#3}}
```

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

```
915 \newrobustcmd*{\GlsXtrSetField}[3]{%
916   \glsxtrsetfieldifexists{#1}{#2}%
917   {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
918 }
```

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

```
919 \newrobustcmd*{\GlsXtrLetField}[3]{%
920   \glsxtrsetfieldifexists{#1}{#2}%
921   {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
922 }
```

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

```
923 \newrobustcmd*{\csGlsXtrLetField}[3]{%
924   \glsxtrsetfieldifexists{#1}{#2}%
925   {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
926 }
```

`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.

```
927 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
928   \glsxtrsetfieldifexists{#1}{#2}%
929   {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
930 }
```

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

```
931 \newrobustcmd*{\gGlsXtrSetField}[3]{%
932   \glsxtrsetfieldifexists{#1}{#2}%
933   {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
934 }
```

`xGlsXtrSetField`

```
935 \newrobustcmd*{\xGlsXtrSetField}[3]{%
936   \glsxtrsetfieldifexists{#1}{#2}%
937   {\protected@csxdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
938 }
```

`eGlsXtrSetField`

```
939 \newrobustcmd*{\eGlsXtrSetField}[3]{%
```

```

940 \glsxtrsetfieldifexists{#1}{#2}%
941 {\protected@csedef{glo@\glsdetoklabel{#1}@\#2}{#3}}%
942 }

```

XtrIfFieldEqStr

```

943 \newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
944 \glsxtrifhasfield{#1}{#2}%
945 {%
946 \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
947 }%
948 {#5}%
949 }

```

rIfFieldEqXpStr Like the above but first expands the string.

```

950 \newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
951 \glsxtrifhasfield{#1}{#2}%
952 {%
953 \protected@edef\@gls@tmp{#3}%
954 \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
955 }%
956 {#5}%
957 }

```

fXpFieldEqXpStr Like the above but also expands the field value.

```

958 \newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
959 \glsxtrifhasfield{#1}{#2}%
960 {%
961 \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
962 \let\glscurrentfieldvalue\@gls@tmp
963 \protected@edef\@gls@tmp{#3}%
964 \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
965 }%
966 {#5}%
967 }

```

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`.

```

968 \ifdef\foreignlanguage
969 {
970 \ifdef\GetTrackedDialectFromLanguageTag
971 {
972 \newcommand{\GlsXtrForeignText}[2]{%

```

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

```

973 \let\@glxtr@org@currentfieldvalue\glscurrentfieldvalue
974 \glxtrifhasfield{\GlsXtrForeignTextField}{#1}%
975 {%
976   \expandafter\GetTrackedDialectFromLanguageTag\expandafter
977     {\glscurrentfieldvalue}{\@glxtr@dialect}%
978   \let\@glxtr@locale\glscurrentfieldvalue
979   \let\glscurrentfieldvalue\@glxtr@org@currentfieldvalue
980   \ifdefempty\@glxtr@dialect
981     {%

```

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

```

982   \ifundef\TrackedDialectClosestSubMatch
983     {%
984       \GlossariesExtraWarning{Can't obtain dialect label
985         (tracklang v1.3.6+ required)}%
986     }%
987     {\let\@glxtr@dialect\TrackedDialectClosestSubMatch}%
988   }%
989   {}%
990   \ifdefempty\@glxtr@dialect
991     {%

```

No tracked dialect found for the root language.

```

992   }%
993   {%

```

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

```

994   \ifcsundef{captions\@glxtr@dialect}{}%
995   {%

```

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

```

996   \IfTrackedDialectHasMapping{\@glxtr@dialect}%
997   {%
998     \edef\@glxtr@dialect{%
999       \GetTrackedDialectToMapping{\@glxtr@dialect}}%

```

Does a caption hook exist for this?

```

1000   \ifcsundef{captions\@glxtr@dialect}{}%
1001   {%

```

No mapping. Try root language label instead.

```

1002   \ifcsundef{captions\@tracklang@lang}{}%
1003   {%
1004     \let\@glxtr@dialect\@tracklang@lang
1005   }%
1006   }%
1007   }%
1008   {%

```

No mapping. Try root language label instead.

```

1009         \ifcsundef{captions\@tracklang@lang}{}%
1010         {%
1011             \let\@glstr@dialect\@tracklang@lang
1012         }%
1013     }%
1014 }%
1015 }%
1016 \ifdefempty\@glstr@dialect
1017 {%
1018     \GlsXtrUnknownDialectWarning{\@glstr@locale}{\@tracklang@lang}%
1019     #2%
1020 }%
1021 {\foreignlanguage{\@glstr@dialect}{#2}}%
1022 }%
1023 {#2}% key not set
1024 }
1025 }
1026 {
1027     \newcommand{\GlsXtrForeignText}[2]{%
1028         \GlossariesExtraWarning{Can't encapsulate foreign text:
1029             tracklang v1.3.6+ required}%
1030         #2%
1031     }
1032 }
1033 }
1034 {
    \foreignlanguage isn't defined so just do <text>.
1035     \newcommand{\GlsXtrForeignText}[2]{#2}
1036 }

```

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

```

1037 \newcommand*{\GlsXtrForeignTextField}{userii}

```

nDialectWarning

```

1038 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
1039     \GlossariesExtraWarning{Can't determine valid dialect label
1040         for locale '#1' (root language: #2)}%
1041 }

```

\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.

```

1042 \ifdef\GlsEntryCounterLabelPrefix
1043 {%
1044     \newcommand*{\glstrpageref}[1]{%
1045         \ifglsentrycounter
1046             \pageref{\GlsEntryCounterLabelPrefix\glstetoklabel{#1}}%

```

```

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

```

lossarypreamble

```

1069 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1070 \ifcsdef{glolist@#1}%
1071 {%
1072 \ifcsundef{@glossarypreamble@#1}%
1073 {\csdef{@glossarypreamble@#1}{}}%
1074 {}%
1075 \csappto{@glossarypreamble@#1}{#2}%
1076 }%
1077 {%
1078 \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1079 }%
1080 }

```

lossarypreamble

```

1081 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1082 \ifcsdef{glolist@#1}%
1083 {%
1084 \ifcsundef{@glossarypreamble@#1}%
1085 {\csdef{@glossarypreamble@#1}{}}%
1086 {}%
1087 \cspreto{@glossarypreamble@#1}{#2}%
1088 }%
1089 {%
1090 \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1091 }%

```

1092 }

1.3 Modifications to Commands Provided by glossaries

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

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

`ewglossaryentry`

```
1093 \renewcommand*{\longnewglossaryentry}{%
1094 \ifstar\@glstr@s@longnewglossaryentry\@glstr@longnewglossaryentry
1095 }
```

`ewglossaryentry` Starred version.

```
1096 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1097 \glsdoifnoexists{#1}%
1098 {%
1099 \bgroup
1100 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1101 \long\def\@newglossaryentryprehook{%
1102 \long\def\@glo@desc{#3}%
1103 \@org@newglossaryentryprehook
1104 }%
1105 \renewcommand*{\gls@assign@desc}[1]{%
1106 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
1107 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1108 }
1109 \gls@defglossaryentry{#1}{#2}%
1110 \egroup
1111 }%
1112 }
```

`ewglossaryentry` Unstarred version.

```
1113 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1114 \glsdoifnoexists{#1}%
1115 {%
1116 \bgroup
1117 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1118 \long\def\@newglossaryentryprehook{%
1119 \long\def\@glo@desc{#3\glstrpostlongdescription}%
1120 \@org@newglossaryentryprehook
1121 }%
1122 \renewcommand*{\gls@assign@desc}[1]{%
1123 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

The following is different from the base glossaries.sty:

```

1124      \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
1125      }
1126      \gls@defglossaryentry{#1}{#2}%
1127  \egroup
1128  }%
1129 }
```

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

```

1130 \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.

```

1131 \renewcommand{\newignoredglossary}{%
1132   \@ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
1133 }
```

ignoredglossary The original definition is patched to check for existence.

```

1134 \newcommand*{\glsxtr@org@newignoredglossary}[1]{%
1135   \ifcsdef{glolist@#1}
1136   {%
1137     \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
1138   }%
1139   {%
1140     \ifdefempty\@ignored@glossaries
1141     {%
1142       \edef\@ignored@glossaries{#1}%
1143     }%
1144     {%
1145       \eappto\@ignored@glossaries{,#1}%
1146     }%
1147     \csgdef{glolist@#1}{,}%
1148     \ifcsundef{gls@#1@entryfmt}%
1149     {%
1150       \defglsentryfmt[#1]{\glsentryfmt}%
1151     }%
1152     {}%
1153     \ifdefempty\@gls@nohyperlist
1154     {%
1155       \renewcommand*{\@gls@nohyperlist}{#1}%
1156     }%
1157     {%
1158       \eappto\@gls@nohyperlist{,#1}%
1159     }%
1160   }%
1161 }
```


ignoredglossary Starred form.

```

1162 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
1163   \ifcsdef{glolist@#1}
1164   {%
1165     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1166   }%
1167   {%
1168     \ifdefempty\@ignored@glossaries
1169     {%
1170       \edef\@ignored@glossaries{#1}%
1171     }%
1172     {%
1173       \eappto\@ignored@glossaries{,#1}%
1174     }%
1175     \csgdef{glolist@#1}{,}%
1176     \ifcsundef{gls@#1@entryfmt}%
1177     {%
1178       \defglsentryfmt[#1]{\glsentryfmt}%
1179     }%
1180     {}%
1181   }%
1182 }

```

\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.

```

1183 \glsifusetranslator
1184 {%
1185   \renewcommand*{\glsettoctitle}[1]{%
1186     \ifcsdef{gls@tr@set@#1@toctitle}%
1187     {%
1188       \csuse{gls@tr@set@#1@toctitle}%
1189     }%
1190     {%
1191       \ifcsdef{@glotype@#1@title}%
1192       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1193       {\def\glossarytoctitle{\glossarytitle}}%
1194     }%
1195   }%
1196 }
1197 {
1198   \renewcommand*{\glsettoctitle}[1]{%
1199     \ifcsdef{@glotype@#1@title}%
1200     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1201     {\def\glossarytoctitle{\glossarytitle}}%
1202   }
1203 }

```

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

```

1204 \newcommand{\provideignoredglossary}{%

```

```

1205 \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1206 }

```

ignoredglossary Unstarred version.

```

1207 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1208   \ifcsdef{glolist@#1}
1209   {}%
1210   {%
1211     \ifdefempty\@ignored@glossaries
1212     {%
1213       \edef\@ignored@glossaries{#1}%
1214     }%
1215     {%
1216       \eappto\@ignored@glossaries{,#1}%
1217     }%
1218     \csgdef{glolist@#1}{,}%
1219     \ifcsundef{gls@#1@entryfmt}%
1220     {%
1221       \defglsentryfmt[#1]{\glsentryfmt}%
1222     }%
1223     {}%
1224     \ifdefempty\@gls@nohyperlist
1225     {%
1226       \renewcommand*{\@gls@nohyperlist}{#1}%
1227     }%
1228     {%
1229       \eappto\@gls@nohyperlist{,#1}%
1230     }%
1231   }%
1232 }

```

ignoredglossary Starred form.

```

1233 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1234   \ifcsdef{glolist@#1}
1235   {}%
1236   {%
1237     \ifdefempty\@ignored@glossaries
1238     {%
1239       \edef\@ignored@glossaries{#1}%
1240     }%
1241     {%
1242       \eappto\@ignored@glossaries{,#1}%
1243     }%
1244     \csgdef{glolist@#1}{,}%
1245     \ifcsundef{gls@#1@entryfmt}%
1246     {%
1247       \defglsentryfmt[#1]{\glsentryfmt}%
1248     }%
1249     {}%

```

```

1250 }%
1251 }

```

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

```

1252 \newcommand*{\glsxtrcopytoglossary}[2]{%
1253   \glsdoifexists{#1}%
1254   {%
1255     \ifcsdef{glolist@#2}
1256     {%
1257       \cseappto{glolist@#2}{#1,}%
1258     }%
1259   }%
1260   \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1261 }%
1262 }%
1263 }

```

1.3.1 Existence Checks

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

```

1264 \renewcommand{\glsdoifexists}[2]{%
1265   \ifglstryexists{#1}{#2}%
1266   {%

```

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

```

1267   \edef\glslabel{\glsdetoklabel{#1}}%
1268   \glsxtrundefaction{Glossary entry ‘\glslabel’
1269     has not been defined}{You need to define a glossary entry before
1270     you can reference it.}%
1271   }%
1272 }

```

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

```

1273 \renewcommand{\glsdoifnoexists}[2]{%
1274   \ifglstryexists{#1}{%
1275     \glsxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1276       has already been defined}{}}{#2}%
1277 }

```

`\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.

```

1278 \ifdef\glsdoifexistsordo
1279 {%
1280   \renewcommand{\glsdoifexistsordo}[3]{%
1281     \ifglstryexists{#1}{#2}%
1282     {%

```

```

1283     \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1284     has not been defined}{You need to define a glossary entry
1285     before you can use it.}%
1286     #3%
1287   }%
1288 }%
1289 }
1290 {%
1291   \glstr@warnonexistsordo\glsdoifexistsordo
1292   \newcommand{\glsdoifexistsordo}[3]{%
1293     \ifglstryexists{#1}{#2}%
1294     {%
1295       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1296       has not been defined}{You need to define a glossary entry
1297       before you can use it.}%
1298       #3%
1299     }%
1300   }%
1301 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

1302 \ifdef\doifglossarynoexistsordo
1303 {%
1304   \renewcommand{\doifglossarynoexistsordo}[3]{%
1305     \ifglossaryexists{#1}%
1306     {%
1307       \glstrundefaction{Glossary type '#1' already exists}{}%
1308       #3%
1309     }%
1310     {#2}%
1311   }%
1312 }
1313 {%
1314   \glstr@warnonexistsordo\doifglossarynoexistsordo
1315   \newcommand{\doifglossarynoexistsordo}[3]{%
1316     \ifglossaryexists{#1}%
1317     {%
1318       \glstrundefaction{Glossary type '#1' already exists}{}%
1319       #3%
1320     }%
1321     {#2}%
1322   }%
1323 }
1324

```

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.

```

1325 \appto\@newglossaryentryposthook{%
1326   \ifdefvoid\@glo@see
1327     {\csxdef{glo@\@glo@label @see}{}}%
1328     {%
1329       \csxdef{glo@\@glo@label @see}{\@glo@see}%
1330       \if@glxtr@autoseeindex
1331         \@glxtr@autoindexcrossrefs
1332       \fi
1333     }%
1334 }
1335 \appto\@gls@keymap{,{see}{see}}
```

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

```

1336 \newcommand*{\glxtrusesee}[1]{%
1337   \glsdoifexists{#1}%
1338   {%
1339     \letcs{\@glo@see}{glo\@glsdetoklabel{#1}@see}%
1340     \ifdefempty\@glo@see
1341       {}%
1342     {%
1343       \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1344     }%
1345   }%
1346 }
```

`\glxtr@usesee`

```

1347 \newcommand*{\glxtr@usesee}[1][\@seenname]{%
1348   \@glxtr@usesee[#1]%
1349 }
```

`\@glxtr@usesee`

```

1350 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1351   \glxtruseseeformat{#1}{#2}%
1352 }
```

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

```

1353 \newcommand*{\glxtruseseeformat}[2]{%
1354   \glsseeformat[#1]{#2}{}%
1355 }
```

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

```

1356 \renewcommand*{\glsseeitemformat}[1]{%
```

```

1357 \ifglshashshort{\glslabel}{\glsaccesstext{#1}}{\glsaccessname{#1}}%
1358 }

```

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

```

1359 \newcommand*{\lsxtruseealso}[1]{%
1360   \glsdoifexists{#1}%
1361   {%
1362     \letcs{\@glo@see}{glo\glsdetoklabel{#1}@seealso}%
1363     \ifdefempty\@glo@see
1364     {}%
1365     {%
1366       \expandafter\lsxtruseealsoformat\expandafter{\@glo@see}%
1367     }%
1368   }%
1369 }

```

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

```

1370 \newcommand*{\lsxtruseealsoformat}[1]{%
1371   \glsseeformat[\seealso]{#1}%
1372 }

```

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

```

1373 \newrobustcmd{\glxtrseelist}[1]{%
1374   \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1375 }

```

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

```

1376 \providecommand{\seealso}{see also}

```

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

```

1377 \ifdef\@xdycrossrefhook
1378 {
1379   Add the cross-reference class definition to the hook.
1380   \appto\@xdycrossrefhook{%
1381     \write\glswrite{(define-crossref-class \string"seealso\string"
1382       :unverified )}%
1383     \write\glswrite{(markup-crossref-list
1384       :class \string"seealso\string"^^J\space\space\space
1385       :open \string"\string\glxtruseealsoformat\glsopenbrace\string"
1386       :close \string"\glsclosebrace\string")}%
1387   }

```

Append to class list.

```
1387 \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

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

```
1388 \newrobustcmd*{\glxtrindexseealso}[2]{%
1389   \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1390     \@glxtr@recordsee{#1}{#2}%
1391   \fi
1392   \glsdoifexists{#1}%
1393   {%
1394     \@@glxtrwrglossmark
1395     \def\@gls@xref{#2}%
1396     \@onelevel@sanitize\@gls@xref
1397     \@gls@checkmkidxchars\@gls@xref
1398     \gls@glossary{\csname glo@#1@type\endcsname}{%
1399       (indexentry
1400         :tkey (\csname glo@#1@index\endcsname)
1401         :xref (\string"\@gls@xref\string")
1402         :attr \string"seealso\string"
1403       )
1404     }%
1405   }%
1406 }
1407 }
1408 {
```

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

```
1409 \newrobustcmd*{\glxtrindexseealso}{\glssee[\seealsoname]}
1410 }
```

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`.

```
1411 \ifdef\gls@set@xr@key
1412 {
```

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

```
1413 \define@key{glossentry}{alias}{%
1414   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1415 }
1416 \define@key{glossentry}{seealso}{%
1417   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1418 }
```

Add to the key mappings.

```
1419 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}
```

Set the default value.

```
1420 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%
```

Assign the field values.

```
1421 \appto\@newglossaryentryposthook{%
1422 \ifdefvoid\@glo@seealso
1423 {\csxdef{glo@\@glo@label @seealso}{}}%
1424 {%
1425 \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1426 \if@glxtr@autoindex
1427 \@glxtr@autoindexcrossrefs
1428 \fi
1429 }%
```

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

```
1430 \ifdefvoid\@glo@alias
1431 {\csxdef{glo@\@glo@label @alias}{}}%
1432 {%
1433 \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1434 }%
1435 }
```

Provide user-level commands to access the values.

`\glxtralias`

```
1436 \newcommand*{\glxtralias}[1]{\@gls@entry@field{#1}{alias}}
```

`trseealsolabels`

```
1437 \newcommand*{\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}
```

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

```
1438 \appto\@glo@autoindexhook{%
1439 \ifdefvoid\@glo@alias
1440 {%
1441 \ifdefvoid\@glo@seealso
1442 {}%
1443 {%
1444 \edef\@do@glsssee{\noexpand\glxtrindexseealso
1445 {\@glo@label}{\@glo@seealso}}%
1446 \@do@glsssee
1447 }%
1448 }%
1449 }
```

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

```
1450 \ifdefvoid\@glo@see
1451 {%
1452 \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1453 \@do@glsssee
1454 }%
```



```

1455     {}%
1456   }%
1457 }%
1458 }
1459 {

```

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

`\glsxtralias`

```

1460 \glsaddstoragekey*{alias}{}{\glsxtralias}

```

`trseealsolabels`

```

1461 \glsaddstoragekey*{seealso}{}{\glsxtrseealsolabels}

```

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.

```

1462 \appto\@newglossaryentryposthook{%
1463   \ifcsvoid{glo@\@glo@label @alias}%
1464   {%
1465     \ifcsvoid{glo@\@glo@label @seealso}%
1466     {}%
1467     {%
1468       \edef\@do@glsssee{\noexpand\glsxtrindexseealso
1469         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1470       \@do@glsssee
1471     }%
1472   }%
1473   {%

```

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

```

1474   \ifdefvoid\@glo@see
1475   {%
1476     \edef\@do@glsssee{\noexpand\glsssee
1477       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1478     \@do@glsssee
1479   }%
1480   {}%
1481   }%
1482 }

```

```

1483 }

```

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

```

1484 \AtEndDocument{\if@glsxtrindexcrossrefs\glsxtraddallcrossrefs\fi}

```

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

```

1485 \newcommand*{\glsxtraddallcrossrefs}{%

```

```

1486 \forall glossaries{\@glo@type}%
1487 {%
1488   \forall sentries[\@glo@type]{\@glo@label}%
1489   {%
1490     \ifglused{\@glo@label}%
1491     {\expandafter\@glxtr@addunusedxrefs\expandafter{\@glo@label}}{}%
1492   }%
1493 }%
1494 }

```

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

```

1495 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
1496   \letcs{\@glo@see}{glo@glstetoklabel{#1}@see}%
1497   \ifdefvoid\@glo@see
1498   {%
1499     {%
1500       \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1501     }%
1502     \letcs{\@glo@see}{glo@glstetoklabel{#1}@seealso}%
1503     \ifdefvoid\@glo@see
1504     {%
1505       {%
1506         \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1507       }%
1508     }

```

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

```

1509 \newcommand*{\glxtr@addunused}[1][]{%
1510   \@glxtr@addunused
1511 }

```

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

```

1512 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
1513   \@for\@glxtr@label:=#1\do
1514   {%
1515     \ifglused{\@glxtr@label}{}%
1516     {%
1517       \gladd[format=glxtrunusedformat]{\@glxtr@label}%
1518       \glunset{\@glxtr@label}%
1519       \expandafter\@glxtr@addunusedxrefs\expandafter{\@glxtr@label}%
1520     }%
1521   }%
1522 }

```

xtrunusedformat

```

1523 \newcommand*{\glxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

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

```

1524 \ifdef\glstr@begindocdefs
1525 {%
1526   \renewcommand*\glstr@begindocdefs{%
1527     \ifnum\@glstr@docdefval=1\relax
1528       \glstr@enablesavenonumberlist
1529       \edef\@glstr@restreat{%
1530         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1531       \makeatletter
1532       \InputIfFileExists{\jobname.glsdefs}{\relax}%
1533       \glstr@restreat
1534       \undef\@glstr@restreat
1535       \glstr@defdocnewglossaryentry
1536     \fi
1537   }
1538 }
1539 {}

```

`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.

```

1540 \let\glstr@orgmakenoidxglossaries\makenoidxglossaries
1541 \renewcommand*\makenoidxglossaries{%
1542   \ifdefequal\@glstr@record@setting\@glstr@record@setting@off
1543   {%
1544     \glstr@orgmakenoidxglossaries

```

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

```

1545   \renewcommand*\@do@seeglossary}[2]{%
1546     \@glstr@wrglossmark
1547     \edef\@glstr@label{\glstr@detoklabel{##1}}%
1548     \protected@write\@auxout{\relax}{%
1549       \string\@glstr@reference
1550       {\csname glo@\@glstr@label @type\endcsname}%
1551       {\@glstr@label}%
1552       {%
1553         \string\glstr@seeformat##2}%
1554       }%
1555     }%
1556   }%

```

Check for `docdef=restricted`:

```

1557   \if@glstr@docdefrestricted

```

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

```

1558     \renewcommand*{\@gls@reference}[3]{%
1559     \ifcsundef{@glsref###1}{\csgdef{@glsref###1}{}}{}%
1560     \ifinlistcs{##2}{@glsref###1}%
1561     {}%
1562     {\listcsgadd{@glsref###1}{##2}}%
1563     \ifcsundef{glo@glstdetoklabel{##2}@loclist}%
1564     {\csgdef{glo@glstdetoklabel{##2}@loclist}{}}%
1565     {}%
1566     \listcsgadd{glo@glstdetoklabel{##2}@loclist}{##3}%
1567     }%
1568     \else
        Disable document definitions.
1569     \@glxtrdocdeffalse
1570     \fi
1571     \disable@keys{glossaries-extra}{docdef}%
1572 }%
1573 {%
1574     \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1575     not permitted\MessageBreak
1576     with record=\@glxtr@record@setting\space package option}%
1577     {You may only use \string\makenoidxglossaries\space with the
1578     record=off option}%
1579 }%
1580 }

```

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

```

1581 \renewcommand*{\gls@defdocnewglossaryentry}{%
1582     \ifcase\@glxtr@docdefval
        docdef=false:
1583     \renewcommand*{\newglossaryentry}[2]{%
1584     \PackageError{glossaries-extra}{Glossary entries must
1585     be \MessageBreak defined in the preamble with \MessageBreak
1586     package option 'docdef=false'\MessageBreak(consider using
1587     'docdef=restricted')}{Move your glossary definitions to
1588     the preamble. You can also put them in a \MessageBreak separate file
1589     and load them with \string\loadglsentries.}%
1590     }%
1591     \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.

```

1592     \let\gls@checkseeallowed\relax
1593     \let\newglossaryentry\newglossaryentry
1594     \or

```

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

```

1595     \let\gls@checkseeallowed\relax
1596     \fi
1597 }%

```

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`

```
1598 \newcommand*{\GlsXtrEnableOnTheFly}{%
1599   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1600 }
```

`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.)

```
1601 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1602   \renewcommand*{\glsdetoklabel}[1]{%
1603     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1604     {%
1605       \expandafter\detokenize\expandafter{##1}%
1606     }%
1607     {\detokenize{##1}}}%
1608   }%
1609   \@GlsXtrEnableOnTheFly
1610 }
1611 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1612   \expandafter\if\glsbackslash#1%
1613     #3%
1614   \else
1615     #4%
1616   \fi
1617 }
```

`sxtrstarflywarn`

```
1618 \newcommand*{\glsxtrstarflywarn}{%
1619   \GlossariesExtraWarning{Experimental starred version of
1620   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1621   read the warnings in the glossaries-extra user manual)}}%
1622 }
```

`rEnableOnTheFly`

```
1623 \newcommand*{\@GlsXtrEnableOnTheFly}{%
```

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

These definitions are all assigned the category given by:

`\glsxtrcat`

```
1624 \newcommand*{\glsxtrcat}{general}
```

```

\glsxtr
1625 \newcommand*\glsxtr[1] [] {%
1626 \def\glsxtr@keylist{##1}%
1627 \@glsxtr
1628 }

\@glsxtr
1629 \newcommand*\@glsxtr[2] [] {%
1630 \ifglsentryexists{##2}%
1631 {%
1632 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1633 }%
1634 {%
1635 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1636 description={\nopostdesc},##1}%
1637 }%
1638 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1639 }

\Glsxtr
1640 \newcommand*\Glsxtr[1] [] {%
1641 \def\glsxtr@keylist{##1}%
1642 \@Glsxtr
1643 }

\@Glsxtr
1644 \newcommand*\@Glsxtr[2] [] {%
1645 \ifglsentryexists{##2}%
1646 {%
1647 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1648 }%
1649 {%
1650 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1651 description={\nopostdesc},##1}%
1652 }%
1653 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
1654 }

\glsxtrpl
1655 \newcommand*\glsxtrpl[1] [] {%
1656 \def\glsxtr@keylist{##1}%
1657 \@glsxtrpl
1658 }

\@glsxtrpl
1659 \newcommand*\@glsxtrpl[2] [] {%
1660 \ifglsentryexists{##2}%
1661 {%

```

```

1662     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1663 }%
1664 {%
1665     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1666     description={\nopostdesc},##1}%
1667 }%
1668 \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1669 }

```

\Glsxtrpl

```

1670 \newcommand*\Glsxtrpl[1][1]{%
1671   \def\glstr@keylist{##1}%
1672   \@Glsxtrpl
1673 }

```

\@Glsxtrpl

```

1674 \newcommand*\@Glsxtrpl[2][1]{%
1675   \ifglstryexists{##2}
1676   {%
1677     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1678   }%
1679   {%
1680     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1681     description={\nopostdesc},##1}%
1682   }%
1683   \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1684 }

```

\GlsXtrWarning

```

1685 \newcommand*\GlsXtrWarning[2]{%
1686   \def\glstr@optlist{##1}%
1687   \@onelevel@sanitize\glstr@optlist
1688   \GlossariesExtraWarning{The options '\glstr@optlist' have
1689   been ignored for entry '##2' as it has already been defined}%
1690 }

```

Disable commands after the glossary:

```

1691 \renewcommand\@printglossary[2]{%
1692   \def\glstr@printglossopts{##1}%
1693   \@glstr@orgprintglossary{##1}{##2}%
1694   \def\glstr{\glstr@disabledflycommand\glstr}%
1695   \def\glstrpl{\glstr@disabledflycommand\glstrpl}%
1696   \def\Glsxtr{\glstr@disabledflycommand\Glsxtr}%
1697   \def\Glsxtrpl{\glstr@disabledflycommand\Glsxtrpl}%
1698 }

```

abledflycommand

```

1699 \newcommand*\@glstr@disabledflycommand[1]{%
1700   \PackageError{glossaries-extra}%

```

```

1701   {\string##1\space can't be used after any of the \MessageBreak
1702     glossaries have been displayed}%
1703   {The on-the-fly commands enabled by
1704     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1705     before the glossaries. If you want to use any entries \MessageBreak
1706     after any of the glossaries, you must use the standard \MessageBreak
1707     method of first defining the entry and then using the \MessageBreak
1708     entry with commands like \string\gls}%
1709     @@glxtr@disabledflycommand
1710   }%
1711   \newcommand*{@@glxtr@disabledflycommand}[2][\{##2}

```

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

```

1712   \let\GlsXtrEnableOnTheFly\relax
1713 }
1714 \@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.

```

1715 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

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

etglossarystyle

```

1716 \renewcommand*{\setglossarystyle}[1]{%
1717   \ifcsundef{@glsstyle@#1}%
1718   {%
1719     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1720   }%
1721   {%
1722     \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

1723     \protected@edef\@glxtr@current@style{#1}%
1724   }%
1725   \ifx\@glossary@default@style\relax
1726     \protected@edef\@glossary@default@style{#1}%
1727   \fi
1728 }

```

In case we have an old version of glossaries:

```

1729 \ifdef\@glossary@default@style
1730 {}
1731 {%
1732   \let\@glossary@default@style\relax
1733 }

```


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

```
1734 \ifdef\glslistdottedwidth
1735 {%
1736   \ifdim\glslistdottedwidth=.5\hsize
1737     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1738   \AtBeginDocument{%
1739     \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1740       \setlength{\glslistdottedwidth}{.5\columnwidth}%
1741     \fi
1742   }%
1743 \fi
1744 }
1745 {}%
```

Similarly for \glsdescwidth:

\glsdescwidth

```
1746 \ifdef\glsdescwidth
1747 {%
1748   \ifdim\glsdescwidth=.6\hsize
1749     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1750   \AtBeginDocument{%
1751     \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1752       \setlength{\glsdescwidth}{.6\columnwidth}%
1753     \fi
1754   }%
1755 \fi
1756 }
1757 {}%
```

and for \glspagelistwidth:

lspagelistwidth

```
1758 \ifdef\glspagelistwidth
1759 {%
1760   \ifdim\glspagelistwidth=.1\hsize
1761     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1762   \AtBeginDocument{%
1763     \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1764       \setlength{\glspagelistwidth}{.1\columnwidth}%
1765     \fi
1766   }%
1767 \fi
1768 }
1769 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```
1770 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
```

```

1771 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1772   \glsnonumberlistfalse
1773   \renewcommand*{\glossaryentrynumbers}[1]{%
1774     \ifglsentryexists{\glscurrententrylabel}%
1775     {%
1776       \@glsxtrpreloctag
1777       \GlsXtrFormatLocationList{#1}%
1778       \@glsxtrpostloctag
1779       \gls@save@numberlist{#1}%
1780     }{}%
1781   }%
1782 \else
1783   \glsnonumberlisttrue
1784   \renewcommand*{\glossaryentrynumbers}[1]{%
1785     \ifglsentryexists{\glscurrententrylabel}%
1786     {%
1787       \gls@save@numberlist{#1}%
1788     }{}%
1789   }%
1790 \fi

```

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

```

1791 \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

```

1792 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1793   \let\@glsxtrpreloctag\@glsxtrpreloctag
1794   \let\@glsxtrpostloctag\@glsxtrpostloctag
1795   \renewcommand*{\@glsxtr@pagetag}{#1}%
1796   \renewcommand*{\@glsxtr@pagetag}{#2}%
1797   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
1798     \csgdef{\@glsxtr@preloctag@##1}{##2}%
1799   }%
1800   \renewcommand*{\@glsxtr@doloctag}{%
1801     \ifcsundef{\@glsxtr@preloctag@\glscurrententrylabel}%
1802     {%
1803       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1804         Rerun required}%
1805     }%
1806     {%
1807       \csuse{\@glsxtr@preloctag@\glscurrententrylabel}%
1808     }%

```

```

1809 }%
1810 }
1811 \@onlypreamble\GlsXtrEnablePreLocationTag

```

glsxtrpreloctag

```

1812 \newcommand*{\@glsxtrpreloctag}{%
1813   \let\@glsxtr@org@delimN\delimN
1814   \let\@glsxtr@org@delimR\delimR
1815   \let\@glsxtr@org@glsignore\glsignore
   \gdef is required as the delimiters may occur inside a scope.
1816   \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1817   \renewcommand*{\delimN}{%
1818     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1819     \@glsxtr@org@delimN}%
1820   \renewcommand*{\delimR}{%
1821     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1822     \@glsxtr@org@delimR}%
1823   \renewcommand*{\glsignore}[1]{%
1824     \gdef\@glsxtr@thisloctag{\relax}%
1825     \@glsxtr@org@glsignore{##1}}%
1826   \@glsxtr@doloctag
1827 }

```

glsxtrpreloctag

```

1828 \newcommand*{\@glsxtrpreloctag}{}

```

@glsxtr@pagetag

```

1829 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

1830 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

1831 \newcommand*{\@glsxtrpostloctag}{%
1832   \let\delimN\@glsxtr@org@delimN
1833   \let\delimR\@glsxtr@org@delimR
1834   \let\glsignore\@glsxtr@org@glsignore
1835   \protected@write\@auxout{%
1836     {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
1837 }

```

lsxtrpostloctag

```

1838 \newcommand*{\@glsxtrpostloctag}{}

```

lsxtr@preloctag

```

1839 \newcommand*{\@glsxtr@savepreloctag}[2]{}
1840 \protected@write\@auxout{%
1841   \string\providecommand\string\@glsxtr@savepreloctag[2]{}

```

glsxtr@doloctag

```
1842 \newcommand*{\@glsxtr@doloctag}{}
```

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

```
1843 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
1844   \XKV@plfalse
1845   \XKV@sttrue
1846   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1847   {%
1848     \csname glsnonumberlist\XKV@resa\endcsname
1849     \ifglsnonumberlist
1850       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1851     \else
1852       \def\glossaryentrynumbers##1{%
1853         \@glsxtrpreloctag
1854         \GlsXtrFormatLocationList{##1}%
1855         \@glsxtrpostloctag
1856         \gls@save@numberlist{##1}}%
1857     \fi
1858   }%
1859 }
```

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.

```
1860 \renewcommand*{\glsentryfmt}{%
1861   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}%
1862   \glsifregular{\glslabel}%
1863   {\glsxtrregularfont{\glsgenentryfmt}}%
1864   {%
1865     \ifglshasshort{\glslabel}%
1866     {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
1867     {\glsxtrregularfont{\glsgenentryfmt}}%
1868   }%
1869 }
```

sxtrregularfont Font used for regular entries.

```
1870 \newcommand*{\glsxtrregularfont}[1]{#1}
```

bbreviationfont Font used for abbreviation entries.

```
1871 \newcommand*{\glsxtrabbreviationfont}[1]{#1}
```

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

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

```
1872 \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).

```
1873 \@glxtr@record{#2}{#3}{glslink}%
1874 \glsdoifexists{#3}%
1875 {%
```

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).

```
1876 \let\glxtrorg@ifKV@glslink@hyper@ifKV@glslink@hyper
1877 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1878 \def\glscustomtext{#4}%
1879 \@glxtr@field@linkdefs
1880 #1%
1881 \@gls@link[#2]{#3}{#4}%
1882 \let@ifKV@glslink@hyper\glxtrorg@ifKV@glslink@hyper
1883 }%
1884 \glspostlinkhook
1885 }
```

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

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

```
1886 \let\@glxtr@org@gls@\@gls@
1887 \def\@gls@#1#2{%
1888 \@glxtr@record{#1}{#2}{glslink}%
1889 \@glxtr@org@gls@{#1}{#2}%
1890 }%
```

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

```
1891 \let\@glxtr@org@glspl@\@glspl@
1892 \def\@glspl@#1#2{%
1893 \@glxtr@record{#1}{#2}{glslink}%
1894 \@glxtr@org@glspl@{#1}{#2}%
1895 }%
```

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

```
1896 \let\@glxtr@org@Gls@\@Gls@
1897 \def\@Gls@#1#2{%
```

```

1898 \@glstr@record{#1}{#2}{glslink}%
1899 \@glstr@org@Gls@{#1}{#2}%
1900 }%

```

\@Glspl@ Save the original definition and redefine.

```

1901 \let\@glstr@org@Glspl@\@Glspl@
1902 \def\@Glspl@#1#2{%
1903 \@glstr@record{#1}{#2}{glslink}%
1904 \@glstr@org@Glspl@{#1}{#2}%
1905 }%

```

\@GLS@ Save the original definition and redefine.

```

1906 \let\@glstr@org@GLS@\@GLS@
1907 \def\@GLS@#1#2{%
1908 \@glstr@record{#1}{#2}{glslink}%
1909 \@glstr@org@GLS@{#1}{#2}%
1910 }%

```

\@GLSpl@ Save the original definition and redefine.

```

1911 \let\@glstr@org@GLSpl@\@GLSpl@
1912 \def\@GLSpl@#1#2{%
1913 \@glstr@record{#1}{#2}{glslink}%
1914 \@glstr@org@GLSpl@{#1}{#2}%
1915 }%

```

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

```

1916 \renewcommand*{\@glsdisp}[3][{}]{%
1917 \@glstr@record{#1}{#2}{glslink}%
1918 \glsdoifexists{#2}{%
1919 \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1920 \let\glsifplural\@secondoftwo
1921 \let\glscapscase\@firstofthree
1922 \def\glscustomtext{#3}%
1923 \def\glsinsert{}%
1924 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1925 \@gl@link[#1]{#2}{\@glo@text}%
1926 \ifKV@glslink@local
1927 \glslocalunset{#2}%
1928 \else
1929 \glsunset{#2}%
1930 \fi
1931 }%
1932 \glspostlinkhook
1933 }

```

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

```

1934 \renewcommand*{\@gls@link}[3][{}]{%

```

```

1935 \@glstr@record{#1}{#2}{glslink}%
1936 \glsdoifexistsordo{#2}%
1937 {%
1938   \let\do@gl@link@checkfirsthyper\relax
1939   \@gl@link[#1]{#2}{#3}%
1940 }%
1941 {%
1942   \glstextformat{#3}%
1943 }%
1944 \glspostlinkhook
1945 }

```

sxtrinitwrgloss Set the default if the wrgloss is omitted.

```

1946 \newcommand*{\glstrinitwrgloss}{%
1947   \glsifattribute{\glslabel}{wrgloss}{after}%
1948   {%
1949     \glstrinitwrglossbeforefalse
1950   }%
1951   {%
1952     \glstrinitwrglossbeforetrue
1953   }%
1954 }

```

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

```

1955 \newif\ifglstrinitwrglossbefore
1956 \glstrinitwrglossbeforetrue

```

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

```

1957 \define@choicekey{glslink}{wrgloss}%
1958 [\@glstr@wrglossval\@glstr@wrglossnr]%
1959 {before,after}%
1960 {%
1961   \ifcase\@glstr@wrglossnr\relax
1962     \glstrinitwrglossbeforetrue
1963   \or
1964     \glstrinitwrglossbeforefalse
1965   \fi
1966 }

1967 \define@key{glslink}{thevalue}{\def\@glstr@thevalue{#1}}

1968 \define@key{glslink}{theHvalue}{\def\@glstr@theHvalue{#1}}

```

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

```

1969 \define@boolkey{glslink}[glstr@]{hyperoutside}[true]{}
1970 \glstr@hyperoutsidettrue

```

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

```

1971 \define@key{glslink}{textformat}{%
1972   \ifcsdef{#1}
1973   {%
1974     \letcs{\@glsxtr@local@textformat}{#1}%
1975   }%
1976   {%
1977     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
1978   }%
1979 }

1980 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

```

nithyperoutside Set the default if the hyperoutside is omitted.

```

1981 \newcommand*{\glsxtrinithyperoutside}{%
1982   \glsifattribute{\glslabel}{hyperoutside}{false}%
1983   {%
1984     \glsxtr@hyperoutsidefalse
1985   }%
1986   {%
1987     \glsxtr@hyperoutsidettrue
1988   }%
1989 }

```

r@inc@linkcount Does nothing by default.

```

1990 \newcommand*{\glsxtr@inc@linkcount}{}

```

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

```

1991 \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.

```

1992 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
1993   \protected@edef\@glsxtr@tmp{#2}%
1994   \expandafter#1\expandafter{\@glsxtr@tmp}%
1995 }

```

\@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.

```

1996 \def\@gls@link[#1]#2#3{%
1997   \leavevmode
1998   \edef\glslabel{\glsdetoklabel{#2}}%
1999   \def\@gls@link@opts{#1}%
2000   \let\@gls@link@label\glslabel
2001   \let\@glsnumberformat\@glsxtr@defaultnumberformat
2002   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2003   \edef\gls@type{\csname glo@\glslabel @type\endcsname}%
2004   \let\org@ifKV@glslink@hyper@ifKV@glslink@hyper

```


Save current value of `\glolinkprefix`:

```
2005 \let\@glxtr@org@glolinkprefix\glolinkprefix
```

Initialise `\@glxtr@local@textformat`

```
2006 \let\@glxtr@local@textformat\relax
```

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

```
2007 \def\@glxtr@thevalue{}%
2008 \def\@glxtr@theHvalue{\@glxtr@thevalue}%
```

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

```
2009 \glxtrinitwrgloss
```

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

```
2010 \glxtrinithyperoutside
```

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

```
2011 \@gls@setdefault@glslink@opts
```

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

```
2012 \glxtr@inc@linkcount
```

As the original definition.

```
2013 \do@glstdisablehyperinlist
2014 \do@gls@link@checkfirsthyper
```

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

```
2015 \glslinkpresetkeys
```

Set options.

```
2016 \setkeys{glslink}{#1}%
```

User hook after options are set:

```
2017 \glslinkpostsetkeys
```

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

```
2018 \ifdefempty{\@glxtr@thevalue}%
2019 {%
2020   \@gls@saveentrycounter
2021 }%
2022 {%
2023   \let\theglssentrycounter\@glxtr@thevalue
2024   \def\theHglssentrycounter{\@glxtr@theHvalue}%
2025 }%
2026 \@gls@setsort{glslabel}%
```

Check if the `textformat` key has been used.

```
2027 \ifx\@glxtr@local@textformat\relax
```

Check `textformat` attribute (new to v1.21).

```
2028 \glshasattribute{glslabel}{textformat}%
2029 {%
2030   \edef\@glxtr@attrval{\glsetattribute{glslabel}{textformat}}%
2031   \ifcsdef{\@glxtr@attrval}%
```

```

2032     {%
2033     \letcs{\@glxtr@textformat}{\@glxtr@attrval}%
2034     }%
2035     {%
2036     \GlossariesExtraWarning{Unknown control sequence name
2037     '@@glxtr@attrval' supplied in textformat attribute
2038     for entry '@glslabel'. Reverting to default \string\glstextformat}%
2039     \let\@glxtr@textformat\glstextformat
2040     }%
2041     }%
2042     {%
2043     \let\@glxtr@textformat\glstextformat
2044     }%
2045     \else
2046     \let\@glxtr@textformat\@glxtr@local@textformat
2047     \fi

```

Do write if it should occur before the link text:

```

2048     \ifglxtrinitwrglossbefore
2049     \@do@wrglossary{#2}%
2050     \fi

```

Do the link text:

```

2051     \ifKV@glslink@hyper
2052     \ifglxtr@hyperoutside
2053     \@glslink{\glolinkprefix\glslabel}{\@glxtr@textformat{#3}}%
2054     \else
2055     \@glxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2056     \fi
2057     \else
2058     \ifglxtr@hyperoutside
2059     \glndonohyperlink{\glolinkprefix\glslabel}{\@glxtr@textformat{#3}}%
2060     \else
2061     \@glxtr@textformat{\glndonohyperlink{\glolinkprefix\glslabel}{#3}}%
2062     \fi
2063     \fi

```

Do write if it should occur after the link text:

```

2064     \ifglxtrinitwrglossbefore
2065     \else
2066     \@do@wrglossary{#2}%
2067     \fi

```

Restore original value of \glolinkprefix:

```

2068     \let\glolinkprefix\@glxtr@org@glolinkprefix

```

As the original definition:

```

2069     \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2070 }

```

```

2071 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}

```

```

2072 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}

\lsaddpresetkeys
2073 \newcommand*{\glspresetkeys}{%

\lsaddpostsetkeys
2074 \newcommand*{\glspostsetkeys}{%

\glspresetkeys  Redefine to include \@glxtr@record and suppress in headings
2075 \renewrobustcmd*{\glspresetkeys}[2][\%
2076   \glspresetkeys
2077   {%
2078   {%
2079     \@glspresetmode
2080     \@glspresetrecord{#1}{#2}{\glspreset}%
2081     \glspresetexists{#2}%
2082     {%
2083       \let\@glspresetformat\@glspresetdefaultformat
2084       \edef\@glspresetcounter{\csname glo@\glspresetlabel{#2}\@glspreset\endcsname}%
2085       \def\@glspresetthevalue{%
2086       \def\@glspresettheHvalue{\@glspresetthevalue}%

Implement any default settings (before options are set)
2087     \glspresetkeys
2088     \setkeys{glossadd}{#1}%

Implement any default settings (after options are set)
2089     \glspostsetkeys
2090     \ifempty{\@glspresetthevalue}%
2091     {%
2092       \@glspresetentrycounter
2093     }%
2094     {%
2095       \let\theglspresetentrycounter\@glspresetthevalue
2096       \def\theglspresetentrycounter{\@glspresettheHvalue}%
2097     }%

Define sort key if necessary (in case of sort=use):
2098     \@glspresetsetsort{#2}%
2099     \@do@wrglossary{#2}%
2100   }%
2101 }%
2102 }

\glspreseteach  Performs \glspreset for each entry listed in the mandatory argument.
2103 \newrobustcmd*{\glspreseteach}[2][\%
2104   \@for\@glspresetthislabel:=#2\do{\glspreset[#1]{\@glspresetthislabel}}%
2105 }

```

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

```
2106 \newcommand*{\@glsxtr@field@linkdefs}{%
2107   \let\glsxtrifwasfirstuse\@secondoftwo
2108   \let\glsifplural\@secondoftwo
2109   \let\glscapscase\@firstofthree
2110   \let\glsinsert\@empty
2111 }
```

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

assignfieldfont

```
2112 \newcommand*{\glsxtrassignfieldfont}[1]{%
2113   \ifglstryexists{#1}%
2114   {%
2115     \ifglshasshort{#1}%
2116     {%
2117       \glssetabbrvfmt{\glscategory{#1}}%
2118       \glsifregular{#1}%
2119       {\let\@gls@field@font\glsxtrregularfont}%
2120       {\let\@gls@field@font\@firstofone}%
2121     }%
2122     {%
2123       \glsifnotregular{#1}%
2124       {\let\@gls@field@font\@firstofone}%
2125       {\let\@gls@field@font\glsxtrregularfont}%
2126     }%
2127   }%
2128   {%
2129     \let\@gls@field@font\@gobble
2130   }%
2131 }
```

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

```
2132 \def\@glstext@#1#2[#3]{%
2133   \glsxtrassignfieldfont{#2}%
2134   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2135 }
```

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

```
2136 \def\@GLStext@#1#2[#3]{%
2137   \glsxtrassignfieldfont{#2}%
2138   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2139   {\@gls@field@font{\@GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2140 }
```

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

```
2141 \def\@Glstext@#1#2[#3]{%
2142   \glsxtrassignfieldfont{#2}%
2143 }
```

```

2143 \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2144 {\@gls@field@font{\Glsaccessstext{#2}#3}}}%
2145 }

```

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`

```

2146 \newcommand*{\glstrchecknohyperfirst}[1]{%
2147 \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2148 }

```

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

```

2149 \def\@glsfirst@#1#2[#3]{%
2150 \glstrassignfieldfont{#2}%
    Ensure that \glsfirst honours the nohyperfirst attribute.
2151 \@gls@field@link
2152 [\let\glstrifwasfirstuse\@firstoftwo
2153 \glstrchecknohyperfirst{#2}%
2154 ]{#1}{#2}%
2155 {\@gls@field@font{\glsaccessfirst{#2}#3}}}%
2156 }

```

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

```

2157 \def\@Glsfirst@#1#2[#3]{%
2158 \glstrassignfieldfont{#2}%
    Ensure that \Glsfirst honours the nohyperfirst attribute.
2159 \@gls@field@link
2160 [\let\glstrifwasfirstuse\@firstoftwo
2161 \let\glscapscase\@secondofthree
2162 \glstrchecknohyperfirst{#2}%
2163 ]%
2164 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}}%
2165 }

```

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

```

2166 \def\@GLSfirst@#1#2[#3]{%
2167 \glstrassignfieldfont{#2}%
    Ensure that \GLSfirst honours the nohyperfirst attribute.
2168 \@gls@field@link
2169 [\let\glstrifwasfirstuse\@firstoftwo
2170 \let\glscapscase\@thirdofthree
2171 \glstrchecknohyperfirst{#2}%
2172 ]%
2173 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2174 }

```

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

```
2175 \def\@glsplural@#1#2[#3]{%
2176   \glsxtrassignfieldfont{#2}%
2177   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2178   {\@gls@field@font{\glsaccessplural{#2}#3}}%
2179 }
```

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

```
2180 \def\@Glsplural@#1#2[#3]{%
2181   \glsxtrassignfieldfont{#2}%
2182   \@gls@field@link
2183   [\let\glsifplural\@firstoftwo
2184    \let\glsupcase\@secondoftwo
2185   ]%
2186   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2187 }
```

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

```
2188 \def\@GLSplural@#1#2[#3]{%
2189   \glsxtrassignfieldfont{#2}%
2190   \@gls@field@link
2191   [\let\glsifplural\@firstoftwo
2192    \let\glsupcase\@thirdoftwo
2193   ]%
2194   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2195 }
```

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

```
2196 \def\glsfirstplural@#1#2[#3]{%
2197   \glsxtrassignfieldfont{#2}%
2198   \glsfirstplural@honours the nohyperfirst attribute.
2199   \@gls@field@link
2200   [\let\glsxtrifwasfirstuse\@firstoftwo
2201    \let\glsifplural\@firstoftwo
2202    \glsxtrchecknohyperfirst{#2}%
2203   ]%
2204   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
```

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

```
2205 \def\Glsfirstplural@#1#2[#3]{%
2206   \glsxtrassignfieldfont{#2}%
2207   \Glsfirstplural@honours the nohyperfirst attribute.
2208   \@gls@field@link
2209   [\let\glsxtrifwasfirstuse\@firstoftwo
2210    \let\glsifplural\@firstoftwo
2211    \let\glsupcase\@secondoftwo
```

```

2211 \glstrchecknohyperfirst{#2}%
2212 ]%
2213 {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2214 }

```

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

```

2215 \def\@GLSfirstplural@#1#2[#3]{%
2216 \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
2217 \@gls@field@link
2218 [\let\glstrifwasfirstuse\@firstoftwo
2219 \let\glsifplural\@firstoftwo
2220 \let\glscapscase\@thirdofthree
2221 \glstrchecknohyperfirst{#2}%
2222 ]%
2223 {#1}{#2}%
2224 {\@gls@field@font{\Glsaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2225 }

```

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

```

2226 \def\@glsname@#1#2[#3]{%
2227 \glstrassignfieldfont{#2}%
2228 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
2229 }

```

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

```

2230 \def\@Glsname@#1#2[#3]{%
2231 \glstrassignfieldfont{#2}%
2232 \@gls@field@link
2233 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2234 {\@gls@field@font{\Glsaccessname{#2}#3}}%
2235 }

```

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

```

2236 \def\@GLSname@#1#2[#3]{%
2237 \glstrassignfieldfont{#2}%
2238 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2239 {#1}{#2}%
2240 {\@gls@field@font{\Glsaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2241 }

```

\@glsdesc@

```

2242 \def\@glsdesc@#1#2[#3]{%
2243 \glstrassignfieldfont{#2}%
2244 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
2245 }

```

\@Glsdesc@ First letter uppercase version.

```
2246 \def\@Glsdesc@#1#2[#3]{%
2247   \glstrassignfieldfont{#2}%
2248   \@gls@field@link
2249   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2250   {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
2251 }
```

\@GLSdesc@ All uppercase version.

```
2252 \def\@GLSdesc@#1#2[#3]{%
2253   \glstrassignfieldfont{#2}%
2254   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2255   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2256 }
```

@glsdescplural@ No case-changing version.

```
2257 \def\@glsdescplural@#1#2[#3]{%
2258   \glstrassignfieldfont{#2}%
2259   \@gls@field@link
2260   [\let\glscapscase\@secondoftwo
2261    \let\glsifplural\@firstoftwo
2262   ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
2263 }
```

@Glsdescplural@ First letter uppercase version.

```
2264 \def\@Glsdescplural@#1#2[#3]{%
2265   \glstrassignfieldfont{#2}%
2266   \@gls@field@link
2267   [\let\glscapscase\@secondoftwo
2268    \let\glsifplural\@firstoftwo
2269   ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2270 }
```

@GLSdescplural@ All uppercase version.

```
2271 \def\@GLSdesc@#1#2[#3]{%
2272   \glstrassignfieldfont{#2}%
2273   \@gls@field@link
2274   [\let\glscapscase\@thirdoftwo
2275    \let\glsifplural\@firstoftwo
2276   ]%
2277   {#1}{#2}%
2278   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2279 }
```

\@glssymbol@

```
2280 \def\@glssymbol@#1#2[#3]{%
2281   \glstrassignfieldfont{#2}%
2282   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2283 }
```


\@Glssymbol@ First letter uppercase version.

```
2284 \def\@Glssymbol@#1#2[#3]{%
2285   \glstrassignfieldfont{#2}%
2286   \@gls@field@link
2287   [\let\glscapscase\@secondoftwo]%
2288   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2289 }
```

\@GLSsymbol@ All uppercase version.

```
2290 \def\@GLSsymbol@#1#2[#3]{%
2291   \glstrassignfieldfont{#2}%
2292   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2293   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2294 }
```

lssymbolplural@ No case-changing version.

```
2295 \def\@lssymbolplural@#1#2[#3]{%
2296   \glstrassignfieldfont{#2}%
2297   \@gls@field@link
2298   [\let\glscapscase\@secondoftwo
2299   \let\glsifplural\@firstoftwo
2300   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2301 }
```

lssymbolplural@ First letter uppercase version.

```
2302 \def\@lssymbolplural@#1#2[#3]{%
2303   \glstrassignfieldfont{#2}%
2304   \@gls@field@link
2305   [\let\glscapscase\@secondoftwo
2306   \let\glsifplural\@firstoftwo
2307   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2308 }
```

LSsymbolplural@ All uppercase version.

```
2309 \def\@LSsymbol@#1#2[#3]{%
2310   \glstrassignfieldfont{#2}%
2311   \@gls@field@link
2312   [\let\glscapscase\@thirdoftwo
2313   \let\glsifplural\@firstoftwo
2314   ]%
2315   {#1}{#2}%
2316   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2317 }
```

\@Glsuseri@ First letter uppercase version.

```
2318 \def\@Glsuseri@#1#2[#3]{%
2319   \glstrassignfieldfont{#2}%
2320   \@gls@field@link
```

```

2321 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2322 {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2323 }

```

\@GLSuseri@ All uppercase version.

```

2324 \def\@GLSuseri@#1#2[#3]{%
2325   \glstrassignfieldfont{#2}%
2326   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2327   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
2328 }

```

\@Glsuserii@ First letter uppercase version.

```

2329 \def\@Glsuserii@#1#2[#3]{%
2330   \glstrassignfieldfont{#2}%
2331   \@gls@field@link
2332   [\let\glscapscase\@secondoftwo]%
2333   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2334 }

```

\@GLSuserii@ All uppercase version.

```

2335 \def\@GLSuserii@#1#2[#3]{%
2336   \glstrassignfieldfont{#2}%
2337   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2338   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2339 }

```

\@Glsuseriii@ First letter uppercase version.

```

2340 \def\@Glsuseriii@#1#2[#3]{%
2341   \glstrassignfieldfont{#2}%
2342   \@gls@field@link
2343   [\let\glscapscase\@secondoftwo]%
2344   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
2345 }

```

\@GLSuseriii@ All uppercase version.

```

2346 \def\@GLSuseriii@#1#2[#3]{%
2347   \glstrassignfieldfont{#2}%
2348   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2349   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2350 }

```

\@Glsuseriv@ First letter uppercase version.

```

2351 \def\@Glsuseriv@#1#2[#3]{%
2352   \glstrassignfieldfont{#2}%
2353   \@gls@field@link
2354   [\let\glscapscase\@secondoftwo]%
2355   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
2356 }

```

\@GLSuseriv@ All uppercase version.

```
2357 \def\@GLSuseriv@#1#2[#3]{%
2358   \glstrassignfieldfont{#2}%
2359   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2360   {#1}{#2}%
2361   {\@gls@field@font{\mfirstucMakeUppercase{\glstentryuseriv{#2}#3}}}%
2362 }
```

\@Glsuserv@ First letter uppercase version.

```
2363 \def\@Glsuserv@#1#2[#3]{%
2364   \glstrassignfieldfont{#2}%
2365   \@gls@field@link
2366   [\let\glscapscase\@secondoftwo]%
2367   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}}%
2368 }
```

\@GLSuserv@ All uppercase version.

```
2369 \def\@GLSuserv@#1#2[#3]{%
2370   \glstrassignfieldfont{#2}%
2371   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2372   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glstentryuserv{#2}#3}}}%
2373 }
```

\@Glsuservi@ First letter uppercase version.

```
2374 \def\@Glsuservi@#1#2[#3]{%
2375   \glstrassignfieldfont{#2}%
2376   \@gls@field@link
2377   [\let\glscapscase\@secondoftwo]%
2378   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
2379 }
```

\@GLSuservi@ All uppercase version.

```
2380 \def\@GLSuservi@#1#2[#3]{%
2381   \glstrassignfieldfont{#2}%
2382   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2383   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glstentryuservi{#2}#3}}}%
2384 }
```

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

\@acrshort No case change.

```
2385 \def\@acrshort#1#2[#3]{%
2386   \glsdoifexists{#2}%
2387   {%
2388     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2389     \let\glstrifwasfirstuse\@secondoftwo
2390     \let\glsifplural\@secondoftwo
```

```

2391 \let\glscapscase\@firstofthree
2392 \let\glsinsert\@empty
2393 \def\glscustomtext{%
2394 \acronymfont{\glsaccessshort{#2}}#3%
2395 }%
2396 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2397 }%
2398 \glspostlinkhook
2399 }

```

\@Acrshort First letter uppercase.

```

2400 \def\@Acrshort#1#2[#3]{%
2401 \glsdoifexists{#2}%
2402 {%
2403 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2404 \let\glstrifwasfirstuse\@secondoftwo
2405 \let\glsifplural\@secondoftwo
2406 \let\glscapscase\@secondofthree
2407 \let\glsinsert\@empty
2408 \def\glscustomtext{%
2409 \acronymfont{\Glsaccessshort{#2}}#3%
2410 }%
2411 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2412 }%
2413 \glspostlinkhook
2414 }

```

\@ACRshort All uppercase.

```

2415 \def\@ACRshort#1#2[#3]{%
2416 \glsdoifexists{#2}%
2417 {%
2418 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2419 \let\glstrifwasfirstuse\@secondoftwo
2420 \let\glsifplural\@secondoftwo
2421 \let\glscapscase\@thirdofthree
2422 \let\glsinsert\@empty
2423 \def\glscustomtext{%
2424 \mfirstucMakeUppercase{\acronymfont{\glsaccessshort{#2}}#3}%
2425 }%
2426 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2427 }%
2428 \glspostlinkhook
2429 }

```

\@acrshortpl No case change.

```

2430 \def\@acrshortpl#1#2[#3]{%
2431 \glsdoifexists{#2}%
2432 {%
2433 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2434 \let\glxtrifwasfirstuse\@secondoftwo
2435 \let\glsifplural\@firstoftwo
2436 \let\glscapscase\@firstofthree
2437 \let\glsinsert\@empty
2438 \def\glscustomtext{%
2439     \acronymfont{\glsaccessshortpl{#2}}#3%
2440 }%
2441 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2442 }%
2443 \glspostlinkhook
2444 }

```

\@Acrshortpl First letter uppercase.

```

2445 \def\@Acrshortpl#1#2[#3]{%
2446     \glsdoifexists{#2}%
2447     {%
2448         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2449         \let\glxtrifwasfirstuse\@secondoftwo
2450         \let\glsifplural\@firstoftwo
2451         \let\glscapscase\@secondofthree
2452         \let\glsinsert\@empty
2453         \def\glscustomtext{%
2454             \acronymfont{\Glsaccessshortpl{#2}}#3%
2455         }%
2456         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2457     }%
2458     \glspostlinkhook
2459 }

```

\@ACRshortpl All uppercase.

```

2460 \def\@ACRshortpl#1#2[#3]{%
2461     \glsdoifexists{#2}%
2462     {%
2463         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2464         \let\glxtrifwasfirstuse\@secondoftwo
2465         \let\glsifplural\@firstoftwo
2466         \let\glscapscase\@thirdofthree
2467         \let\glsinsert\@empty
2468         \def\glscustomtext{%
2469             \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
2470         }%
2471         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2472     }%
2473     \glspostlinkhook
2474 }

```

\@acrlong No case change.

```

2475 \def\@acrlong#1#2[#3]{%
2476     \glsdoifexists{#2}%

```

```

2477 {%
2478   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2479   \let\glxtrifwasfirstuse\@secondoftwo
2480   \let\gl@ifplural\@secondoftwo
2481   \let\gl@scapscase\@firstofthree
2482   \let\gl@insert\@empty
2483   \def\glscustomtext{%
2484     \acronymfont{\gl@saccesslong{#2}}#3%
2485   }%
2486   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2487 }%
2488 \glspostlinkhook
2489 }

```

\@Acrlong First letter uppercase.

```

2490 \def\@Acrlong#1#2[#3]{%
2491   \gl@ifexists{#2}%
2492   {%
2493     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2494     \let\glxtrifwasfirstuse\@secondoftwo
2495     \let\gl@ifplural\@secondoftwo
2496     \let\gl@scapscase\@secondofthree
2497     \let\gl@insert\@empty
2498     \def\glscustomtext{%
2499       \acronymfont{\gl@saccesslong{#2}}#3%
2500     }%
2501     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2502   }%
2503   \glspostlinkhook
2504 }

```

\@ACRlong All uppercase.

```

2505 \def\@ACRlong#1#2[#3]{%
2506   \gl@ifexists{#2}%
2507   {%
2508     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2509     \let\glxtrifwasfirstuse\@secondoftwo
2510     \let\gl@ifplural\@secondoftwo
2511     \let\gl@scapscase\@thirdofthree
2512     \let\gl@insert\@empty
2513     \def\glscustomtext{%
2514       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslong{#2}}#3}%
2515     }%
2516     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2517   }%
2518   \glspostlinkhook
2519 }

```

\@acrlongpl No case change.

```

2520 \def\@acrlongpl#1#2[#3]{%
2521   \glsdoifexists{#2}%
2522   {%
2523     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2524     \let\glxtrifwasfirstuse\@secondoftwo
2525     \let\gl@sifplural\@firstoftwo
2526     \let\glscapscase\@firstofthree
2527     \let\gl$insert\@empty
2528     \def\glscustomtext{%
2529       \acronymfont{\gl@saccesslongpl{#2}}#3%
2530     }%
2531     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2532   }%
2533   \glspostlinkhook
2534 }

```

\@Acrlongpl First letter uppercase.

```

2535 \def\@Acrlongpl#1#2[#3]{%
2536   \glsdoifexists{#2}%
2537   {%
2538     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2539     \let\glxtrifwasfirstuse\@secondoftwo
2540     \let\gl@sifplural\@firstoftwo
2541     \let\glscapscase\@secondofthree
2542     \let\gl$insert\@empty
2543     \def\glscustomtext{%
2544       \acronymfont{\Glsaccesslongpl{#2}}#3%
2545     }%
2546     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2547   }%
2548   \glspostlinkhook
2549 }

```

\@ACRlongpl All uppercase.

```

2550 \def\@ACRlongpl#1#2[#3]{%
2551   \glsdoifexists{#2}%
2552   {%
2553     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2554     \let\glxtrifwasfirstuse\@secondoftwo
2555     \let\gl@sifplural\@firstoftwo
2556     \let\glscapscase\@thirdofthree
2557     \let\gl$insert\@empty
2558     \def\glscustomtext{%
2559       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslongpl{#2}}#3}%
2560     }%
2561     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2562   }%
2563   \glspostlinkhook
2564 }

```

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

\@glsaddkey

```

2565 \renewcommand*{\@glsaddkey}[7]{%
2566   \key@ifundefined{glossentry}{#1}%
2567   {%
2568     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2569     \appto\@gls@keymap{,{#1}{#1}}%
2570     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2571     \appto\@newglossaryentryposthook{%
2572       \letcs{\@glo@tmp}{@glo@#1}%
2573       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2574     }%
2575     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2576     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2577   \ifcsdef{@gls@user@#1@}%
2578   {%
2579     \PackageError{glossaries}%
2580     {Can't define '\string#5' as helper command
2581     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2582     exists}%
2583     {}%
2584   }%
2585   {%
2586     \expandafter\newcommand\expandafter*\expandafter
2587     {\csname @gls@user@#1\endcsname}[2][ ]{%
2588       \new@ifnextchar[%
2589         {\csuse{@gls@user@#1@}{##1}{##2}}%
2590         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2591     \csdef{@gls@user@#1@}##1##2[##3]{%
2592       \@gls@field@link{##1}{##2}{#3{##2}##3}%
2593     }%
2594     \newrobustcmd*{#5}{%
2595       \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2596   }%

```

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

```

2597   \ifcsdef{@Gls@user@#1@}%
2598   {%
2599     \PackageError{glossaries}%
2600     {Can't define '\string#6' as helper command
2601     '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2602     exists}%
2603     {}%
2604   }%
2605   {%
2606     \expandafter\newcommand\expandafter*\expandafter

```



```

2607      {\csname @Gls@user@#1\endcsname}[2] [] {%
2608        \new@ifnextchar[%
2609          {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2610          {\csuse{@Gls@user@#1@}{##1}{##2} [] }}}%
2611      \csdef{@Gls@user@#1@}##1##2[##3]{%
2612        \@gls@field@link[\let\glscapscase\secondofthree]%
2613        {##1}{##2}{#4{##2}##3}%
2614      }%
2615      \newrobustcmd*{#6}{%
2616        \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2617    }%

```

Finally the all caps version (modified):

```

2618      \ifcsdef{@GLS@user@#1@}%
2619      {%
2620        \PackageError{glossaries}%
2621        {Can't define '\string#7' as helper command
2622         '\expandafter\string\csname @GLS@user@#1\endcsname' already
2623         exists}%
2624      }%
2625    }%
2626    {%
2627      \expandafter\newcommand\expandafter*\expandafter
2628      {\csname @GLS@user@#1\endcsname}[2] [] {%
2629        \new@ifnextchar[%
2630          {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2631          {\csuse{@GLS@user@#1@}{##1}{##2} [] }}}%
2632      \csdef{@GLS@user@#1@}##1##2[##3]{%
2633        \@gls@field@link[\let\glscapscase\thirdofthree]%
2634        {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2635      }%
2636      \newrobustcmd*{#7}{%
2637        \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2638      }%
2639    }%
2640    {%
2641      \PackageError{glossaries-extra}{Key '#1' already exists}{}%
2642    }%
2643 }

```

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

```

2644 \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.

```

2645 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2646 \renewcommand*{\@gls@link@checkfirsthyper}{%

```

\ifglused 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.

```

2647 \ifglused{\glslabel}%
2648 {\let\glstrifwasfirstuse\@secondoftwo}
2649 {\let\glstrifwasfirstuse\@firstoftwo}%

Store the category label for convenience.

2650 \edef\glscategorylabel{\glscategory{\glslabel}}%
2651 \ifglused{\glslabel}%
2652 {%
2653   \glscategoryattribute{\glscategorylabel}{nohypernext}{true}%
2654   {\KV@glslink@hyperfalse}{}}%
2655 }%
2656 {%
2657   \glscategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
2658   {\KV@glslink@hyperfalse}{}}%
2659 }%
2660 \glslinkcheckfirsthyperhook
2661 }

```

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.

```

2662 \ifdefined\do@glsglisablehyperinlist
2663 {%
2664   \let\@glstr@do@glsglisablehyperinlist\do@glsglisablehyperinlist
2665   \renewcommand*{\do@glsglisablehyperinlist}{%
2666     \@glstr@do@glsglisablehyperinlist
2667     \glscategoryattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}}%
2668   }
2669 }
2670 {}

```

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

```

2671 \define@boolkey{glslink}{noindex}[true]{}
2672 \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

```

2673 \ifdefined\@gls@setdefault@glslink@opts
2674 {
2675   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2676     \KV@glslink@noindexfalse
2677     \@glstrsetaliasnoindex
2678   }
2679 }
2680 {

```

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

```
2681 \newcommand*{\@glsglslSetDefault@glslink@opts}{%
2682   \KV@glslink@noindexfalse
2683   \@glsxtrsetaliasnoindex
2684 }
2685 \preto\do@glsglslDisableHyperinlist{\@glsglslSetDefault@glslink@opts}
2686 }
```

`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.)

```
2687 \providecommand*{\glsxtrsetaliasnoindex}{%
2688   \KV@glslink@noindextrue
2689 }
```

`setaliasnoindex`

```
2690 \newcommand*{\@glsxtrsetaliasnoindex}{%
2691   \glsxtrifhasfield{alias}{\glslabel}%
2692   {%
2693     \let\glsxtrindexaliased\@glsxtrindexaliased
2694     \glsxtrsetaliasnoindex
2695     \let\glsxtrindexaliased\@no@glsxtrindexaliased
2696   }%
2697   {%
2698 }
```

`xtrindexaliased`

```
2699 \newcommand{\@glsxtrindexaliased}{%
2700   \ifKV@glslink@noindex
2701   \else
2702     \begingroup
2703     \let\@glslnumberformat\@glsxtr@defaultnumberformat
2704     \edef\@glsglsl@counter{\csname glo@glsglsl@detoklabel{\glslabel}@counter\endcsname}%
2705     \glsxtr@saveentrycounter
2706     \do@wrglossary{\glsxtralias{\glslabel}}%
2707   \endgroup
2708   \fi
2709 }
```

`xtrindexaliased`

```
2710 \newcommand{\@no@glsxtrindexaliased}{%
2711   \PackageError{glossaries-extra}{\string\glsxtrindexaliased\space
2712     not permitted outside definition of \string\glsxtrsetaliasnoindex}%
2713   {%
2714 }
```

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

```
2715 \let\glsxtrindexaliased\@no@glsxtrindexaliased
```

DefaultGlsOpts Set the default options for \glslink etc.

```
2716 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2717   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2718     \setkeys{glslink}{#1}%
2719     \@glsxtrsetaliasnoindex
2720   }%
2721 }
```

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

```
2722 \newcommand*{\glsxtrifindexing}[2]{%
2723   \ifKV@glslink@noindex #2\else #1\fi
2724 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
2725 \renewcommand*{\glswriteentry}[2]{%
2726   \glsxtrifindexing
2727   {%
2728     \ifglsindexonlyfirst
2729       \ifglsused{#1}
2730       {\glsxtrdoautoindexname{#1}{dualindex}}%
2731       {#2}%
2732     \else
2733       \glsifattribute{#1}{indexonlyfirst}{true}%
2734       {\ifglsused{#1}
2735        {\glsxtrdoautoindexname{#1}{dualindex}}%
2736        {#2}}%
2737       {#2}%
2738     \fi
2739   }%
2740   {%
2741 }
```

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

```
2742 \appto\@do@@wrglossary{\@glsxtr@do@@wrindex
2743   \glsxtrdowrglossaryhook{\@gls@label}%
2744 }
```

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

Similarly for the “noidx” version:

s@noidxglossary

```
2745 \appto\gls@noidxglossary{\@glsxtr@do@@wrindex
2746   \glsxtrdowrglossaryhook{\@gls@label}%
2747 }
```

xtr@do@@wrindex

```
2748 \newcommand*{\@glsxtr@do@@wrindex}{%
```

\csglossaryhook	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.)
-----------------	---

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

alt@glsl@hyp@opt User version

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

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

rSetAltModifier

org@dohyperlink

`\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.

```
2776 \ifdef\glsnavhyperlink
2777 {
2778   \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
2779     \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
2780   {%
2781     \let\glsdohyperlink\glsxtr@org@dohyperlink
2782     \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
2783   }%
2784 }%
2785 }
2786 {}
```

`\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 `\glsxtrshort` to use `\glsentryshort` and, similarly, the long form commands like `\acrlong` and `\glsxtrlong` to use `\glsentrylong`. Added attribute check.

```
2787 \renewcommand*{\glsdohyperlink}[2]{%
2788   \glsasattribute{\glslabel}{targeturl}%
2789   {%
2790     \glsasattribute{\glslabel}{targetname}%
2791     {%
2792       \glsasattribute{\glslabel}{targetcategory}%
2793       {%
2794         \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
2795           {\glsgetattribute{\glslabel}{targetcategory}}%
2796           {\glsgetattribute{\glslabel}{targetname}}%
2797           {\glsxtrprotectlinks#2}}%
2798       }%
2799     }%
2800     \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
2801       {%
2802         {\glsgetattribute{\glslabel}{targetname}}%
2803         {\glsxtrprotectlinks#2}}%
2804     }%
2805   }%
2806   {%
2807     \href{\glsgetattribute{\glslabel}{targeturl}}{%
2808       {\glsxtrprotectlinks#2}}%
2809   }%
2810 }%
2811 {}
```

Check for alias.

```

2812 \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2813 \ifdefvoid\gloaliaslabel
2814 {%
2815     \glsxtrhyperlink{#1}{\glsxtrprotectlinks#2}}%
2816 }%
2817 {%

```

Redirect link to the alias target.

```

2818     \glsxtrhyperlink
2819     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2820     {\glsxtrprotectlinks#2}}%
2821 }%
2822 }%
2823 }

```

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

```

2824 \ifdef\@glsshowtarget
2825 {
2826     \newcommand{\glsxtrhyperlink}[2]{%
2827         \@glsshowtarget{#1}%
2828         \hyperlink{#1}{#2}%
2829     }%
2830 }
2831 {
2832     \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
2833 }

```

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.

```

2834 \renewrobustcmd*{\glsdohyperlink}[2][\glssentrytext{\@glo@label}]{%
2835     \glsdoifexists{#2}%
2836     {%
2837         \def\@glo@label{#2}%
2838         {\edef\glslabel{#2}%
2839         \@glslink{\glolinkprefix\glslabel}{#1}}%
2840     }%
2841 }

```

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

```

2842 \renewcommand{\glsdisablehyper}{%
2843     \KV@glslink@hyperfalse
2844     \def\@glslink{\glsdohyperlink}%
2845     \let\@glstarget\@secondoftwo
2846 }

```

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

```
2847 \renewcommand{\glsenablehyper}{%
2848   \KV@glslink@hypertrue
2849   \def\@glslink{\glsdohyperlink}%
2850   \def\@glstarget{\glsdohypertarget}%
2851 }
```

`\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).

```
2852 \def\glsdonohyperlink#1#2{{\glstrprotectlinks #2}}
```

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

```
2853 \ifcsundef{hyperlink}%
2854 {%
2855   \def\@glslink{\glsdonohyperlink}
2856 }%
2857 {%
2858   \def\@glslink{\glsdohyperlink}
2859 }
```

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

```
2860 \newcommand*{\glstrprotectlinks}{%
2861   \KV@glslink@hyperfalse
2862   \KV@glslink@noindextrue
2863   \let\@gls\@glstr@p@text@
2864   \let\@Gls\@Glsxtr@p@text@
2865   \let\@GLS\@GLSxtr@p@text@
2866   \let\@glspl\@glstr@p@plural@
2867   \let\@Glspl\@Glsxtr@p@plural@
2868   \let\@GLSpl\@GLSxtr@p@plural@
2869   \let\@glstrshort\@glstr@p@short@
2870   \let\@Glsxtrshort\@Glsxtr@p@short@
2871   \let\@GLSxtrshort\@GLSxtr@p@short@
2872   \let\@glstrlong\@glstr@p@long@
2873   \let\@Glsxtrlong\@Glsxtr@p@long@
2874   \let\@GLSxtrlong\@GLSxtr@p@long@
2875   \let\@glstrshortpl\@glstr@p@shortpl@
2876   \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
2877   \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
2878   \let\@glstrlongpl\@glstr@p@longpl@
2879   \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
2880   \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
2881   \let\@acrshort\@glstr@p@acrshort@
2882   \let\@Acrshort\@Glsxtr@p@acrshort@
```



```

2883 \let\@ACRshort\@GLSxtr@p@acrshort@
2884 \let\@acrshortpl\@glxtr@p@acrshortpl@
2885 \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
2886 \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
2887 \let\@acrlong\@glxtr@p@acrlong@
2888 \let\@Acrlong\@Glsxtr@p@acrlong@
2889 \let\@ACRlong\@GLSxtr@p@acrlong@
2890 \let\@acrlongpl\@glxtr@p@acrlongpl@
2891 \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
2892 \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
2893 }

```

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

@glxtr@p@text@

```
2894 \def\@glxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtr@p@text@

```
2895 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtr@p@text@

```
2896 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
2897 \def\@lsxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lSxtr@p@plural@

```
2898 \def\@lSxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
2899 \def\@LSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtr@p@short@

```

2900 \def\@glxtr@p@short@#1#2[#3]{%
2901 {%
2902   \glsssetabbrvfmt{\glscategory{#2}}%
2903   \glssabbrvfont{\glssentryshort{#2}}#3%
2904 }%
2905 }

```

Glsxtr@p@short@

```

2906 \def\@Glsxtr@p@short@#1#2[#3]{%
2907 {%
2908   \glsssetabbrvfmt{\glscategory{#2}}%
2909   \glssabbrvfont{\Glsentryshort{#2}}#3%
2910 }%
2911 }

```

GLSxtr@p@short@

```
2912 \def\@GLSxtr@p@short@#1#2[#3]{%
2913   {%
2914     \glsetabbrvfmt{\glscategory{#2}}%
2915     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
2916   }%
2917 }
```

sxtr@p@shortpl@

```
2918 \def\@glsxtr@p@shortpl@#1#2[#3]{%
2919   {%
2920     \glsetabbrvfmt{\glscategory{#2}}%
2921     \glsabbrvfont{\glsentryshortpl{#2}}#3%
2922   }%
2923 }
```

Sxtr@p@shortpl@

```
2924 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2925   {%
2926     \glsetabbrvfmt{\glscategory{#2}}%
2927     \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2928   }%
2929 }
```

Sxtr@p@shortpl@

```
2930 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2931   {%
2932     \glsetabbrvfmt{\glscategory{#2}}%
2933     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
2934   }%
2935 }
```

@glsxtr@p@long@

```
2936 \def\@glsxtr@p@long@#1#2[#3]{\{\glsentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
2937 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
2938 \def\@GLSxtr@p@long@#1#2[#3]{%
2939   {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
2940 \def\@glsxtr@p@longpl@#1#2[#3]{\{\glsentrylongpl{#2}#3}}
```

lsxtr@p@longpl@

```
2941 \def\@Glsxtr@p@longpl@#1#2[#3]{\{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

```

LSxtr@p@longpl@
2942 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2943   {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
2944 \def\@glsxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
2945 \def\@GLSxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
2946 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2947   {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
2948 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
2949 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
2950 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2951   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
2952 \def\@glsxtr@p@acrlong@#1#2[#3]{\{\glsentrylong{#2}}#3}}

sxtr@p@acrlong@
2953 \def\@GLSxtr@p@acrlong@#1#2[#3]{\{\Glsentrylong{#2}}#3}}

Sxtr@p@acrlong@
2954 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2955   {\mfirstucMakeUppercase{\glsentrylong{#2}}#3}}}

tr@p@acrlongpl@
2956 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\{\glsentrylongpl{#2}}#3}}

tr@p@acrlongpl@
2957 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{\{\Glsentrylongpl{#2}}#3}}

tr@p@acrlongpl@
2958 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2959   {\mfirstucMakeUppercase{\glsentrylongpl{#2}}#3}}}

Commands to minimise conflict.

\@glsxtrp@opt
2960 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}

```

`\glsxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
2961 \newcommand*{\glsxtrsetpopts}[1]{%
2962   \renewcommand*{\@glsxtrp@opt}{#1}%
2963 }
```

`\glossxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
2964 \newcommand*{\glossxtrsetpopts}{%
2965   \glsxtrsetpopts{noindex}%
2966 }
```

`\@@glsxtrp`

```
2967 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
2968   {%
2969     \let\glspostlinkhook\relax
2970     \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
2971   }%
2972 }
```

`\@glsxtrp`

```
2973 \newrobustcmd*{\@glsxtrp}[2]{%
2974   \ifcsdef{gls#1}%
2975     {%
2976       \@glsxtrp{gls#1}{#2}%
2977     }%
2978     {%
2979       \ifcsdef{glsxtr#1}%
2980         {%
2981           \@glsxtrp{glsxtr#1}{#2}%
2982         }%
2983         {%
2984           \PackageError{glossaries-extra}{‘#1’ not recognised by
2985             \string\glsxtrp}{}%
2986         }%
2987       }%
2988 }
```

`\@Glsxtrp`

```
2989 \newrobustcmd*{\@Glsxtrp}[2]{%
2990   \ifcsdef{Gls#1}%
2991     {%
2992       \@glsxtrp{Gls#1}{#2}%
2993     }%
2994     {%
2995       \ifcsdef{Glsxtr#1}%
2996         {%
2997           \@glsxtrp{Glsxtr#1}{#2}%
2998         }%
2999     }
```

```

2999   {%
3000       \PackageError{glossaries-extra}{‘#1’ not recognised by
3001           \string\Glsxtrp}{}%
3002   }%
3003 }%
3004 }

```

\@GLSxtrp

```

3005 \newrobustcmd*{\@GLSxtrp}[2]{%
3006   \ifcsdef{GLS#1}%
3007   {%
3008       \@@glxtrp{GLS#1}{#2}%
3009   }%
3010   {%
3011       \ifcsdef{GLSxtr#1}%
3012       {%
3013           \@@glxtrp{GLSxtr#1}{#2}%
3014       }%
3015       {%
3016           \PackageError{glossaries-extra}{‘#1’ not recognised by
3017               \string\GLSxtrp}{}%
3018       }%
3019   }%
3020 }

```

\glxtr@entry@p

```

3021 \newrobustcmd*{\glxtr@headentry@p}[2]{%
3022   \gl@ifattribute{#1}{headuc}{true}%
3023   {%
3024       \mfirstucMakeUppercase{\@glx@entry@field{#1}{#2}}%
3025   }%
3026   {%
3027       \@glx@entry@field{#1}{#2}%
3028   }%
3029 }

```

\glxtrp Not robust as it needs to expand somewhat.

```

3030 \ifdef\texorpdfstring
3031 {
3032   \newcommand{\glxtrp}[2]{%
3033     \protect\NoCaseChange
3034     {%
3035         \protect\texorpdfstring
3036         {%
3037             \protect\glxtrifinmark
3038             {%
3039                 \ifcsdef{glxtrhead#1}%
3040                 {%
3041                     {\protect\csuse{glxtrhead#1}{#2}}%

```

```

3042         }%
3043         {%
3044             \glxstr@headentry@p{#2}{#1}%
3045         }%
3046     }%
3047     {%
3048         \@glxstrp{#1}{#2}%
3049     }%
3050 }%
3051 {%
3052     \protect\@gls@entry@field{#2}{#1}%
3053 }%
3054 }%
3055 }
3056 }
3057 {
3058     \newcommand{\glxstrp}[2]{%
3059         \protect\NoCaseChange
3060         {%
3061             \protect\glxstrifinmark
3062             {%
3063                 \ifcsdef{glxstrhead#1}%
3064                 {%
3065                     {\protect\csuse{glxstrhead#1}}%
3066                 }%
3067             }%
3068             \glxstr@headentry@p{#2}{#1}%
3069         }%
3070     }%
3071     {%
3072         \@glxstrp{#1}{#2}%
3073     }%
3074 }%
3075 }
3076 }

```

Provide short synonyms for the most common option.

`\glsps`

```
3077 \newcommand*{\glsps}{\glxstrp{short}}
```

`\glspt`

```
3078 \newcommand*{\glspt}{\glxstrp{text}}
```

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

```

3079 \ifdef\texorpdfstring
3080 {
3081     \newcommand{\Glsxtrp}[2]{%

```

```

3082 \protect\NoCaseChange
3083 {%
3084 \protect\texorpdfstring
3085 {%
3086 \protect\glxtrifinmark
3087 {%
3088 \ifcsdef{Glsxtrhead#1}%
3089 {%
3090 {\protect\csuse{Glsxtrhead#1}{#2}}%
3091 }%
3092 {%
3093 \protect\@Gls@entry@field{#2}{#1}%
3094 }%
3095 }%
3096 {%
3097 \@Glsxtrp{#1}{#2}%
3098 }%
3099 }%
3100 {%
3101 \protect\@gls@entry@field{#2}{#1}%
3102 }%
3103 }%
3104 }
3105 }
3106 {
3107 \newcommand{\Glsxtrp}[2]{%
3108 \protect\NoCaseChange
3109 {%
3110 \protect\glxtrifinmark
3111 {%
3112 \ifcsdef{Glsxtrhead#1}%
3113 {%
3114 {\protect\csuse{Glsxtrhead#1}}%
3115 }%
3116 {%
3117 \protect\@Gls@entry@field{#2}{#1}%
3118 }%
3119 }%
3120 {%
3121 \@Glsxtrp{#1}{#2}%
3122 }%
3123 }%
3124 }
3125 }

```

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

```

3126 \ifdef\texorpdfstring
3127 {
3128 \newcommand{\Glsxtrp}[2]{%

```

```

3129 \protect\NoCaseChange
3130 {%
3131 \protect\texorpdfstring
3132 {%
3133 \protect\glstrifinmark
3134 {%
3135 \ifcsdef{GLSxtr#1}%
3136 {%
3137 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3138 }%
3139 {%
3140 \protect\mfirstucMakeUppercase
3141 {%
3142 \protect\@gls@entry@field{#2}{#1}%
3143 }%
3144 }%
3145 }%
3146 {%
3147 \@GLSxtrp{#1}{#2}%
3148 }%
3149 }%
3150 {%
3151 \protect\@gls@entry@field{#2}{#1}%
3152 }%
3153 }%
3154 }
3155 }
3156 {
3157 \newcommand{\GLSxtrp}[2]{%
3158 \protect\NoCaseChange
3159 {%
3160 \protect\glstrifinmark
3161 {%
3162 \ifcsdef{GLSxtr#1}%
3163 {%
3164 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3165 }%
3166 {%
3167 \protect\mfirstucMakeUppercase
3168 {%
3169 \protect\@gls@entry@field{#2}{#1}%
3170 }%
3171 }%
3172 }%
3173 {%
3174 \@GLSxtrp{#1}{#2}%
3175 }%
3176 }%
3177 }

```


3178 }

1.3.5 Entry Counting

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

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

\@glsxtr@unset Global unset.

```
3179 \newcommand*{\@glsxtr@unset}[1]{%
3180   \@glsunset{#1}%
3181   \glsxtrpostunset{#1}%
3182 }%
```

\@glsunset Global unset.

```
3183 \let\@glsunset\@glsxtr@unset
```

glsxtrpostunset

```
3184 \newcommand*{\glsxtrpostunset}[1]{}
```

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

tUnsetBuffering

```
3185 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3186   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3187 }
```

tUnsetBuffering Unstarred version doesn't check for duplicates.

```
3188 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3189   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3190   \def\@glsxtr@unset@buffer{}%
3191   \let\@glsunset\@glsxtrbuffer@unset
3192 }
```

tUnsetBuffering Starred version checks for duplicates.

```
3193 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3194   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3195   \def\@glsxtr@unset@buffer{}%
```

```

3196 \let\@glsunset\@glsxtrbuffer@nodup@unset
3197 }

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

3198 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3199 \listxadd\@glsxtr@unset@buffer{#1}%
3200 }

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

3201 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3202 \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
3203 {\listxadd\@glsxtr@unset@buffer{#1}}%
3204 }

pUnsetBuffering

3205 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3206 \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3207 }

pUnsetBuffering Unstarred form (global unset).

3208 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3209 \let\@glsunset\@glsxtr@unset
3210 \forlistloop\@glsunset\@glsxtr@unset@buffer
3211 \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3212 }

pUnsetBuffering Starred form (local unset).

3213 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3214 \forlistloop\@glslocalunset\@glsxtr@unset@buffer
3215 \let\@glsunset\@glsxtr@unset
3216 }

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

3217 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3218 \forlistloop#1\@glsxtr@unset@buffer
3219 }

\@glslocalunset Local unset.

3220 \renewcommand*{\@glslocalunset}[1]{%
3221 \@glslocalunset{#1}%
3222 \glsxtrpostlocalunset{#1}%
3223 }%

rpostlocalunset

3224 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

```

\@glsreset   Global reset.
3225 \renewcommand*{\@glsreset}[1]{%
3226   \@glsreset{#1}%
3227   \glsxtrpostreset{#1}%
3228 }%

glsxtrpostreset
3229 \newcommand*{\glsxtrpostreset}[1]{%

\@glslocalreset   Local reset.
3230 \renewcommand*{\@glslocalreset}[1]{%
3231   \@glslocalreset{#1}%
3232   \glsxtrpostlocalreset{#1}%
3233 }%

rpostlocalreset
3234 \newcommand*{\glsxtrpostlocalreset}[1]{%

slocalreseteach   Locally reset a list of entries.
3235 \newcommand*{\glslocalreseteach}[1]{%
3236   \gls@ifnotmeasuring
3237   {%
3238     \@for\@gls@thislabel:=#1\do{%
3239       \glsdoifexists{\@gls@thislabel}%
3240       {%
3241         \@glslocalreset{\@gls@thislabel}%
3242       }%
3243     }%
3244   }%
3245 }

slocalunseteach   Locally unset a list of entries.
3246 \newcommand*{\glslocalunseteach}[1]{%
3247   \gls@ifnotmeasuring
3248   {%
3249     \@for\@gls@thislabel:=#1\do{%
3250       \glsdoifexists{\@gls@thislabel}%
3251       {%
3252         \@glslocalunset{\@gls@thislabel}%
3253       }%
3254     }%
3255   }%
3256 }

leEntryCounting   The first argument is the list of categories and the second argument is the value of the en-
trycount attribute.
3257 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```
3258 \glsenableentrycount
```

Redefine \gls etc:

```
3259 \renewcommand*{\gls}{\cglsl}%
3260 \renewcommand*{\Gls}{\cGls}%
3261 \renewcommand*{\glspl}{\cglspl}%
3262 \renewcommand*{\Glspl}{\cGlspl}%
3263 \renewcommand*{\GLS}{\cGLS}%
3264 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
3265 \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
3266 \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3267 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3268   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3269     can't be used with \string\GlsXtrEnableEntryCounting}%
3270   {Use one or other but not both commands}}%
3271 }
```

ycountunsetattr

```
3272 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
3273   \@for\@glsxtr@cat:=#1\do
3274   {%
3275     \ifdefempty{\@glsxtr@cat}{}%
3276     {%
3277       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3278     }%
3279   }%
3280 }
```

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

nableentrycount

```
3281 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
3282 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
3283 \renewcommand*{\gls@defdocnewglossaryentry}{%
3284   \renewcommand*{\newglossaryentry}[2]{%
3285     \PackageError{glossaries}{\string\newglossaryentry\space
3286       may only be used in the preamble when entry counting has
3287       been activated}{If you use \string\glsenableentrycount\space
3288       you must place all entry definitions in the preamble not in
3289       the document environment}%
3290   }%
3291 }
```

New commands to access new fields:

```

3292 \newcommand*{\glsentrycurrcount}[1]{%
3293   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3294   {0}{\@gls@entry@field{##1}{currcount}}%
3295   }%
3296 \newcommand*{\glsentryprevcount}[1]{%
3297   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3298   {0}{\@gls@entry@field{##1}{prevcount}}%
3299   }%

```

Adjust post unset and reset:

```

3300 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
3301 \renewcommand*{\glsxtrpostunset}[1]{%
3302   \@glsxtr@entrycount@org@unset{##1}%
3303   \@gls@increment@currcount{##1}%
3304   }%
3305 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
3306 \renewcommand*{\glsxtrpostlocalunset}[1]{%
3307   \@glsxtr@entrycount@org@localunset{##1}%
3308   \@gls@local@increment@currcount{##1}%
3309   }%
3310 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
3311 \renewcommand*{\glsxtrpostreset}[1]{%
3312   \@glsxtr@entrycount@org@reset{##1}%
3313   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3314   }%
3315 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
3316 \renewcommand*{\glsxtrpostlocalreset}[1]{%
3317   \@glsxtr@entrycount@org@localreset{##1}%
3318   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
3319   }%

```

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

```

3320 \let\@cgl@s\@cgl@s
3321 \let\@cgl spl\@cgl spl

3322 \let\@cGls\@cGls
3323 \let\@cGlspl\@cGlspl
3324 \let\@cGLS\@cGLS
3325 \let\@cGLSpl\@cGLSpl

```

The rest is as the original definition.

```

3326 \AtEndDocument{\@gls@write@entrycounts}%
3327 \renewcommand*{\@gls@entry@count}[2]{%
3328   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
3329   }%
3330 \let\glsenableentrycount\relax
3331 \renewcommand*{\glsenableentryunitcount}{%
3332   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space

```

```

3333     can't be used with \string\glsenableentrycount}%
3334     {Use one or other but not both commands}%
3335 }%
3336 }

```

`\writeentrycounts` 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.

```

3337 \renewcommand*{\@gls@write@entrycounts}{%
3338   \immediate\write\@auxout
3339   {\string\providecommand*\string\@gls@entry@count}[2]{}%
3340   \count@=0\relax
3341   \forallglsentries{\@glsentry}{%
3342     \glshasattribute{\@glsentry}{entrycount}%
3343     {%
3344       \ifglsused{\@glsentry}%
3345       {%
3346         \immediate\write\@auxout
3347         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3348       }%
3349     }%
3350     \advance\count@ by \@ne
3351   }%
3352 }%
3353 }%
3354 \ifnum\count@=0
3355   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3356   \MessageBreak with \string\glsenableentrycount\space but the
3357   \MessageBreak attribute 'entrycount' hasn't
3358   \MessageBreak been assigned to any of the defined
3359   \MessageBreak entries}%
3360 \fi
3361 }

```

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

```

3362 \newcommand*\glstrifcounttrigger[3]{%
3363   \glshasattribute{#1}{entrycount}%
3364   {%
3365     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
3366     #3%
3367   \else
3368     #2%
3369   \fi
3370 }%
3371 {#3}%
3372 }

```

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

\@@cgl's@

```

3373 \def\@@cgl's@#1#2[#3]{%
3374   \gl'sxtrifcounttrigger{#2}%
3375   {%
3376     \cgl'sformat{#2}{#3}%
3377     \gl'sunset{#2}%
3378   }%
3379   {%
3380     \@gl's@{#1}{#2}[#3]%
3381   }%
3382 }%
```

\@@cgl'spl@

```

3383 \def\@@cgl'spl@#1#2[#3]{%
3384   \gl'sxtrifcounttrigger{#2}%
3385   {%
3386     \cgl'splformat{#2}{#3}%
3387     \gl'sunset{#2}%
3388   }%
3389   {%
3390     \@gl'spl@{#1}{#2}[#3]%
3391   }%
3392 }%
```

\@@cGl's@

```

3393 \def\@@cGl's@#1#2[#3]{%
3394   \gl'sxtrifcounttrigger{#2}%
3395   {%
3396     \cGl'sformat{#2}{#3}%
3397     \gl'sunset{#2}%
3398   }%
3399   {%
3400     \@Gl's@{#1}{#2}[#3]%
3401   }%
3402 }%
```

\@@cGl'spl@

```

3403 \def\@@cGl'spl@#1#2[#3]{%
3404   \gl'sxtrifcounttrigger{#2}%
3405   {%
3406     \cGl'splformat{#2}{#3}%
3407     \gl'sunset{#2}%
3408   }%
3409   {%
3410     \@Gl'spl@{#1}{#2}[#3]%
3411   }%
3412 }%
```

\@@cGLS@

```
3413 \def\@@cGLS@#1#2[#3]{%
3414   \glxtrifcounttrigger{#2}%
3415   {%
3416     \cGLSformat{#2}{#3}%
3417     \glset{#2}%
3418   }%
3419   {%
3420     \@GLS@{#1}{#2}[#3]%
3421   }%
3422 }%
```

\@@cGLSpl@

```
3423 \def\@@cGLSpl@#1#2[#3]{%
3424   \glxtrifcounttrigger{#2}%
3425   {%
3426     \cGLSplformat{#2}{#3}%
3427     \glset{#2}%
3428   }%
3429   {%
3430     \@GLSpl@{#1}{#2}[#3]%
3431   }%
3432 }%
```

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

\@cgl's@

```
3433 \def\@cgl's@#1#2[#3]{\@gl's@{#1}{#2}[#3]}
```

\@cGL's@

```
3434 \def\@cGL's@#1#2[#3]{\@GL's@{#1}{#2}[#3]}
```

\@cgl'spl@

```
3435 \def\@cgl'spl@#1#2[#3]{\@gl'spl@{#1}{#2}[#3]}
```

\@cGL'spl@

```
3436 \def\@cGL'spl@#1#2[#3]{\@GL'spl@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

\cGLS

```
3437 \newrobustcmd*{\cGLS}{\@gl'shyp@opt\@cGLS}
```

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

```
3438 \newcommand*{\@cGLS}[2][ ]{%
3439   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[ ]}]%
3440 }
```



```

\@cGLS@
3441 \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.
3442 \newcommand*\cGLSformat}[2]{%
3443   \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
3444 }

\cGLSpl
3445 \newrobustcmd*\cGLSpl{\@gls@hyp@opt\@cGLSpl}

\@cGLSpl  Defined the un-starred form. Need to determine if there is a final optional argument
3446 \newcommand*\@cGLSpl}[2][{}]{%
3447   \new@ifnextchar[\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[{}]}%
3448 }

\@cGLSpl@
3449 \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.
3450 \newcommand*\cGLSplformat}[2]{%
3451   \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3452 }

      Modify the trigger formats to check for the regular attribute.

\cglformat
3453 \renewcommand*\cglformat}[2]{%
3454   \glsifregular{#1}
3455   {\glsentryfirst{#1}}%
3456   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3457 }

\cGlsformat
3458 \renewcommand*\cGlsformat}[2]{%
3459   \glsifregular{#1}
3460   {\Glsentryfirst{#1}}%
3461   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3462 }

\cglsplformat
3463 \renewcommand*\cglsplformat}[2]{%
3464   \glsifregular{#1}
3465   {\glsentryfirstplural{#1}}%
3466   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3467 }

```

\cGlsplformat

```
3468 \renewcommand*{\cGlsplformat}[2]{%
3469   \glsifregular{#1}
3470   {\Glsentryfirstplural{#1}}%
3471   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2}%
3472 }
```

New code similar to above for unit counting.

defunitcounters

```
3473 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3474   \edef\@glo@countunit{\csuse{\glstr@categoryattr\@glo@category @unitcount}}%
3475   \ifvoid\@glo@countunit
3476   {}%
3477   {%
3478     \@glstr@ifunitcounter{\@glo@countunit}%
3479     {}%
3480     {\expandafter\@glstr@addunitcounter\expandafter{\@glo@countunit}}%
3481   }%
3482 }
```

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

```
3483 \newcommand*{\@glstr@unitcountlist}{}%
```

@addunitcounter

```
3484 \newcommand*{\@glstr@addunitcounter}[1]{%
3485   \listadd{\@glstr@unitcountlist}{#1}%
3486   \ifcsundef{glstr@theunit@#1}
3487   {%
3488     \ifcsdef{theH#1}%
3489     {\csdef{glstr@theunit@#1}{\csuse{theH#1}}}%
3490     {\csdef{glstr@theunit@#1}{\csuse{the#1}}}%
3491   }%
3492   {}%
3493 }
```

r@ifunitcounter

```
3494 \newcommand*{\@glstr@ifunitcounter}[3]{%
3495   \xifinlist{#1}{\@glstr@unitcountlist}{#2}{#3}%
3496 }
```

urrentunitcount

```
3497 \newcommand*{\@glstr@currentunitcount}[1]{%
3498   glo@\glstetoklabel{#1}@currunit@\glstetattribute{#1}{unitcount}.%
3499   \csuse{glstr@theunit@\glstetattribute{#1}{unitcount}}%
3500 }
```

previousunitcount

```
3501 \newcommand*\@glxtr@previousunitcount[1]{%
3502   glo@\glstdetoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
3503   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
3504 }
```

t@currunitcount

```
3505 \newcommand*\@glxtr@increment@currunitcount[1]{%
3506   \glshasattribute{#1}{unitcount}%
3507   {%
3508     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3509     \ifcsundef{\@glxtr@csname}%
3510     {%
3511       \csgdef{\@glxtr@csname}{1}%
3512       \listcsxadd
3513         {glo@\glstdetoklabel{#1}@unitlist}%
3514         {\glsggetattribute{#1}{unitcount}.%
3515         \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
3516       }%
3517     }%
3518     {%
3519       \csxdef{\@glxtr@csname}%
3520       {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3521     }%
3522   }%
3523   {}%
3524 }
```

t@currunitcount

```
3525 \newcommand*\@glxtr@local@increment@currunitcount[1]{%
3526   \glshasattribute{#1}{unitcount}%
3527   {%
3528     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3529     \ifcsundef{\@glxtr@csname}%
3530     {%
3531       \csdef{\@glxtr@csname}{1}%
3532       \listcseadd
3533         {glo@\glstdetoklabel{#1}@unitlist}%
3534         {\glsggetattribute{#1}{unitcount}.%
3535         \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}}%
3536       }%
3537     }%
3538     {%
3539       \csedef{\@glxtr@csname}%
3540       {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3541     }%
3542   }%
3543   {}%
3544 }
```

r@currunitcount

```
3545 \newcommand*{\@glxstr@currunitcount}[2]{%
3546   \ifcsundef
3547     {glo@\glstetoklabel{#1}@currunit@#2}%
3548     {0}%
3549     {\csuse{glo@\glstetoklabel{#1}@currunit@#2}}%
3550 }%
```

r@prevunitcount

```
3551 \newcommand*{\@glxstr@prevunitcount}[2]{%
3552   \ifcsundef
3553     {glo@\glstetoklabel{#1}@prevunit@#2}%
3554     {0}%
3555     {\csuse{glo@\glstetoklabel{#1}@prevunit@#2}}%
3556 }%
```

entryunitcount

```
3557 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3558   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3559   \renewcommand*{\gls@defdocnewglossaryentry}{%
3560     \renewcommand*\newglossaryentry[2]{%
3561       \PackageError{glossaries}{\string\newglossaryentry\space
3562         may only be used in the preamble when entry counting has
3563         been activated}{If you use \string\glsenableentryunitcount\space
3564         you must place all entry definitions in the preamble not in
3565         the document environment}%
3566     }%
3567   }%
```

New commands to access new fields:

```
3568   \newcommand*{\glsentrycurrcount}[1]{%
3569     \@glxstr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3570     \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
3571   }%
3572   \newcommand*{\glsentryprevcount}[1]{%
3573     \@glxstr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
3574     \csuse{glxstr@theunit@\glsgetattribute{##1}{unitcount}}}%
3575   }%
```

Access total count:

```
3576   \newcommand*{\glsentryprevtotalcount}[1]{%
3577     \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
3578     {0}%
3579     {%
3580       \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
3581     }%
3582   }%
```

Access max value:

```

3583 \newcommand*{\glstentryprevmaxcount}[1]{%
3584 \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
3585 {0}%
3586 {%
3587 \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
3588 }%
3589 }%

```

Adjust post unset and reset:

```

3590 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
3591 \renewcommand*{\glstxtrpostunset}[1]{%
3592 \@glstxtr@entryunitcount@org@unset{##1}%
3593 \@glst@increment@currunitcount{##1}%
3594 }%
3595 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
3596 \renewcommand*{\glstxtrpostlocalunset}[1]{%
3597 \@glstxtr@entryunitcount@org@localunset{##1}%
3598 \@glst@local@increment@currunitcount{##1}%
3599 }%
3600 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
3601 \renewcommand*{\glstxtrpostreset}[1]{%
3602 \glshasattribute{##1}{unitcount}%
3603 {%
3604 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
3605 \ifcsundef{\@glstxtr@csname}%
3606 {}%
3607 {\csgdef{\@glstxtr@csname}{0}}%
3608 }%
3609 {}%
3610 }%
3611 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
3612 \renewcommand*{\glstxtrpostlocalreset}[1]{%
3613 \@glstxtr@entryunitcount@org@localreset{##1}%
3614 \glshasattribute{##1}{unitcount}%
3615 {%
3616 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
3617 \ifcsundef{\@glstxtr@csname}%
3618 {}%
3619 {\csdef{\@glstxtr@csname}{0}}%
3620 }%
3621 {}%
3622 }%

```

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

```

3623 \let\@cglst@\@cglst@
3624 \let\@cglstpl@\@cglstpl@
3625 \let\@cGlst@\@cGlst@

```

```

3626 \let\@cGlspl@\@cGlspl@
3627 \let\@cGLS@\@cGLS@
3628 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3629 \AtEndDocument{\@gls@write@entryunitcounts}%
3630 \renewcommand*{\@gls@entry@unitcount}[3]{%
3631   \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
3632   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3633   {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
3634   {%
3635     \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
3636       \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
3637     }%
3638     \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3639     {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
3640     {%
3641       \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3642       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3643     }%
3644   }%
3645 }%
3646 \let\glsenableentryunitcount\relax
3647 \renewcommand*{\glsenableentrycount}{%
3648   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3649     can't be used with \string\glsenableentryunitcount}%
3650   {Use one or other but not both commands}%
3651 }%
3652 }
3653 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3654 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3655 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3656   \immediate\write\@auxout
3657   {\string\@gls@entry@unitcount
3658     {\@glsentry}%
3659     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3660     }%
3661     {#1}}%
3662 }

```

entryunitcounts

```

3663 \newcommand*{\@gls@write@entryunitcounts}{%
3664   \immediate\write\@auxout
3665   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3666   \count@=0\relax

```

```

3667 \forallglentries{\@glentry}{%
3668   \glshasattribute{\@glentry}{unitcount}%
3669   {%
3670     \ifglused{\@glentry}%
3671     {%
3672       \forlistcsloop
3673         {\@glswrite@entryunitcounts@do}%
3674         {glo@\glsetoklabel{\@glentry}@unitlist}%
3675     }%
3676   }%
3677   \advance\count@ by \@ne
3678 }%
3679 {}%
3680 }%
3681 \ifnum\count@=0
3682   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3683     \MessageBreak with \string\glsenableentryunitcount\space but the
3684     \MessageBreak attribute ‘unitcount’ hasn’t
3685     \MessageBreak been assigned to any of the defined
3686     \MessageBreak entries}%
3687 \fi
3688 }

```

`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.

```

3689 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3690   \glsenableentryunitcount

```

Redefine `\gls` etc:

```

3691   \renewcommand*{\gls}{\cgl}%
3692   \renewcommand*{\Gls}{\cGls}%
3693   \renewcommand*{\glspl}{\cglspl}%
3694   \renewcommand*{\Glspl}{\cGlspl}%
3695   \renewcommand*{\GLS}{\cGLS}%
3696   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3697   \@glxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3698   \let\GlsXtrEnableEntryUnitCounting\@glxtr@setentryunitcountunsetattr
3699   \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3700     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3701       can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3702     {Use one or other but not both commands}}%
3703 }

```

`tcountunsetattr`

```

3704 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3705   \@for\@glsxtr@cat:=#1\do
3706   {%
3707     \ifdefempty{\@glsxtr@cat}{}%
3708     {%
3709       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3710       \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3711     }%
3712   }%
3713 }

```

1.3.6 Acronym Modifications

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

`\SetGenericNewAcronym`

```

3714 \renewcommand*{\SetGenericNewAcronym}{%
3715   \let\@Gls@entryname\@Gls@acrentryname
3716   \renewcommand{\newacronym}[4][{}]{%
3717     \ifdefempty{\@glsacronymlists}%
3718     {%
3719       \def\@glo@type{\acronymtype}%
3720       \setkeys{glossentry}{##1}%
3721       \DeclareAcronymList{\@glo@type}%
3722     }%
3723   }%
3724   \glskeylisttok{##1}%
3725   \glslabeltok{##2}%
3726   \glsshorttok{##3}%
3727   \glslongtok{##4}%
3728   \newacronymhook
3729   \protected@edef\@do@newglossaryentry{%
3730     \noexpand\newglossaryentry{\the\glslabeltok}%
3731     {%
3732       type=\acronymtype,%
3733       name={\expandonce{\acronymentry{##2}}},%
3734       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3735       text={\the\glsshorttok},%
3736       short={\the\glsshorttok},%
3737       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3738       long={\the\glslongtok},%
3739       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3740       category=acronym,

```



```

3741 \GenericAcronymFields,%
3742 \the\glskeylisttok
3743 }%
3744 }%
3745 \@do@newglossaryentry
3746 }%
3747 \renewcommand*{\acrfullfmt}[3]{%
3748 \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
3749 \renewcommand*{\Acrfullfmt}[3]{%
3750 \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
3751 \renewcommand*{\ACRfullfmt}[3]{%
3752 \glslink{##1}{##2}{%
3753 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
3754 \renewcommand*{\acrfullplfmt}[3]{%
3755 \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
3756 \renewcommand*{\Acrfullplfmt}[3]{%
3757 \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
3758 \renewcommand*{\ACRfullplfmt}[3]{%
3759 \glslink{##1}{##2}{%
3760 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
3761 \renewcommand*{\glstryfull}[1]{\genacrfullformat{##1}{}}}%
3762 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
3763 \renewcommand*{\glstryfullpl}[1]{\genplacrfullformat{##1}{}}}%
3764 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
3765 }

```

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:

```

3766 \let\@glstr@org@setacronymstyle\setacronymstyle
3767 \let\@glstr@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`.

```

3768 \newcommand*{\MakeAcronymsAbbreviations}{%
3769 \renewcommand*{\newacronym}[4][{}]{%
3770 \glstr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3771 }%
3772 \renewcommand*{\firstacronymfont}[1]{\glstr@firstabbrvfont{##1}}%
3773 \renewcommand*{\acronymfont}[1]{\glstr@abbrvfont{##1}}%
3774 \renewcommand*{\setacronymstyle}[1]{%
3775 \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
3776 unavailable.
3777 Use \string\setabbreviationstyle\space instead.
3778 The original acronym interface can be restored with
3779 \string\RestoreAcronyms}{}%
3780 }%
3781 \renewcommand*{\newacronymstyle}[1]{%

```

```

3782 \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3783 available unless you restore the original acronym interface with
3784 \string\RestoreAcronyms}%
3785 \@glxtr@org@newacronymstyle{##1}%
3786 }%
3787 }

```

Switch acronyms to abbreviations:

```
3788 \MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3789 \newcommand*{\RestoreAcronyms}{%
3790 \SetGenericNewAcronym
3791 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3792 \renewcommand{\acronymfont}[1]{##1}%
3793 \let\setacronymstyle\@glxtr@org@setacronymstyle
3794 \let\newacronymstyle\@glxtr@org@newacronymstyle

```

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

```

3795 \renewcommand*\@gls@link@checkfirsthyper{%
3796 \ifglused{\glslabel}%
3797 {\let\glxtrifwasfirstuse\@secondoftwo}
3798 {\let\glxtrifwasfirstuse\@firstoftwo}%
3799 \@glxtr@org@checkfirsthyper
3800 }
3801 \glssetcategoryattribute{acronym}{regular}{false}%
3802 \setacronymstyle{long-short}%
3803 }

```

\glsacspace Allow the user to customise the maximum value.

```

3804 \renewcommand*{\glsacspace}[1]{%
3805 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3806 \ifdim\dimen@<\glsacspacemax~\else\space\fi
3807 }

```

\glsacspacemax Value used in the above.

```
3808 \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

```
3809 \newcommand*{\@glxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
3810 \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
```

```
3811 \renewcommand*{\makeglossaries}[1] [] {%
3812   \ifx\@glxtr@record@setting\@glxtr@record@setting@only
3813     \PackageError{glossaries-extra}{\string\makeglossaries\space
3814       not permitted\MessageBreak with record=only package option}%
3815     {You may only use \string\makeglossaries\space with
3816       record=off or record=alsoindex options}%
3817   \else
3818     \ifblank{#1}%
3819     {\@glxtr@org@makeglossaries}%
3820     {%
3821       \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3822         \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3823           not permitted\MessageBreak with record=alsoindex package option}%
3824         {You may only use the hybrid \string\makeglossaries[...]\space with
3825           record=off option}%
3826       \else
3827         \edef\@glxtr@reg@glosslist{#1}%
3828         \ifundef{\glswrite}{\newwrite\glswrite}{}%
3829         \protected@write\@auxout{}{\string\providecommand
3830           \string\@glxtr@order[1]{}}
3831         \protected@write\@auxout{}{\string\providecommand
3832           \string\@istfilename[1]{}}
3833         \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3834         \protected@write\@auxout{}{\string\@glxtr@order{\glxtr@order}}
3835         \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}
3836         \write\@auxout{\string\providecommand\string\@glxtr@reference[3]{}}%

```

Iterate through each supplied glossary type and activate it.

```
3837   \@for\@glo@type:=#1\do{%
3838     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3839   }%
```

New glossaries must be created before `\makeglossaries`:

```
3840   \renewcommand*\newglossary[4] [] {%
3841     \PackageError{glossaries}{New glossaries
3842       must be created before \string\makeglossaries}{You need
3843       to move \string\makeglossaries\space after all your
3844       \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
3845   \let\@makeglossary\relax
3846   \let\makeglossary\relax
```

```

3847 \renewcommand\makeglossaries[1][]{%
  Disable all commands that have no effect after \makeglossaries
3848 \@disable@onlypremakeg
  Allow see key:
3849 \let\gls@checkseeallowed\relax
  Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment
  associated counter.
3850 \renewcommand*\@do@seeglossary}[2]{%
3851 \glsdoifexists{##1}%
3852 {%
3853 \edef\@gls@label{\glsdetoklabel{##1}}%
3854 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3855 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3856 {\@glsxtr@org@doseeglossary{##1}{##2}}%
3857 {%
3858 \@@glsxtrwrglossmark
3859 \protected@write\@auxout{}{%
3860 \string\@gls@reference
3861 {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
3862 }%
3863 }%
3864 }%
3865 }%
  Adjust \@do@@wrglossary
3866 \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
3867 \def\@do@@wrglossary{%
3868 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3869 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3870 {\@glsxtr@@do@@wrglossary}%
3871 {\gls@noidxglossary}%
3872 }%
  Suppress warning about no \makeglossaries
3873 \let\warn@nomakeglossaries\relax
3874 \def\warn@noprintglossary{%
3875 \GlossariesWarningNoLine{No \string\printglossary\space
3876 or \string\printglossaries\space
3877 found.^^J(Remove \string\makeglossaries\space if you don't want
3878 any glossaries.)^^JThis document will not have a glossary}%
3879 }%
  Only warn for glossaries not listed.
3880 \renewcommand{\@gls@noref@warn}[1]{%
3881 \edef\@gls@type{##1}%
3882 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3883 {%
3884 \GlossariesExtraWarning{Can't use
3885 \string\printnoidxglossary[type={\@gls@type}]

```

```

3886         when '@gls@type' is listed in the optional argument of
3887         \string\makeglossaries}%
3888     }%
3889     {%
3890         \GlossariesWarning{Empty glossary for
3891         \string\printnoidxglossary[type={##1}].
3892         Rerun may be required (or you may have forgotten to use
3893         commands like \string\gls)}%
3894     }%
3895 }%

```

Adjust display number list to check for type:

```

3896 \renewcommand*{\glsdisplaynumberlist}[1]{%
3897 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3898 {\@glsxtr@idx@displaynumberlist{##1}}%
3899 {\@glsxtr@noidx@displaynumberlist{##1}}%
3900 }%

```

Adjust entry list:

```

3901 \renewcommand*{\glsentrynumberlist}[1]{%
3902 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3903 {\@glsxtr@idx@entrynumberlist{##1}}%
3904 {\@glsxtr@noidx@entrynumberlist{##1}}%
3905 }%

```

Adjust number list loop

```

3906 \renewcommand*{\glsnumberlistloop}[2]{%
3907 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3908 {%
3909     \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3910     not available for glossary '@##1'}{%
3911     }%
3912     {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3913 }%

```

Only sanitize sort for normal indexing glossaries.

```

3914 \renewcommand*{\glsprestandardsort}[3]{%
3915 \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3916 {%
3917     \glsdosanitizesort
3918 }%
3919 {%
3920     \ifglssanitizesort
3921     \@gls@noidx@sanitizesort
3922     \else
3923     \@gls@noidx@nosanitizesort
3924     \fi
3925 }%
3926 }%

```

Unlike \makenoidxglossaries we can't automatically set sanitizesort=false. All entries must be defined in the preamble.

```

3927 \renewcommand*\new@glossaryentry[2]{%
3928 \PackageError{glossaries-extra}{Glossary entries must be defined
3929 in the preamble\MessageBreak when you use the optional argument
3930 of \string\makeglossaries}{Either move your definitions to the
3931 preamble or don't use the optional argument of
3932 \string\makeglossaries}%
3933 }%

```

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).

```

3934 \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
3935 \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3936 \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
3937 type=\glsdefaulttype,\@end@glxtr@gettype
3938 \def\@glo@sorttype{\@glo@default@sorttype}%
3939 }%

```

Check automake setting:

```

3940 \ifglautomake
3941 \renewcommand*\@gls@doautomake{%
3942 \@for\@gls@type:=\@glxtr@reg@glosslist\do{%
3943 \ifdefempty{\@gls@type}}{\@gls@automake{\@gls@type}}%
3944 }%
3945 }%
3946 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3947 \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3948 \fi
3949 }%
3950 \fi
3951 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\orgprintglossary` 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.)

```

3952 \newcommand{\@glxtr@orgprintglossary}[2]{%
3953 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3954 \def\glossarytitle{%
3955 \ifcsdef{\@glo@type@\@glo@type @title}%
3956 {\csuse{\@glo@type@\@glo@type @title}}%
3957 {\glossaryname}}%
3958 \def\glossarytoctitle{\glossarytitle}%

```

```

3959 \let\org@glossarytitle\glossarytitle
3960 \def\@glossarystyle{%
3961   \ifx\@glossary@default@style\relax
3962     \GlossariesWarning{No default glossary style provided \MessageBreak
3963       for the glossary '\@glo@type'. \MessageBreak
3964       Using deprecated fallback. \MessageBreak
3965       To fix this set the style with \MessageBreak
3966       \string\setglossarystyle\space or use the \MessageBreak
3967       style key=value option}%
3968   \fi
3969 }%
3970 \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
3971 \let\org@glossaryentrynumbers\glossaryentrynumbers
3972 \bgroup
3973   \@printgloss@setsort
3974   \setkeys{printgloss}{#1}%
3975   \ifx\glossarytitle\org@glossarytitle
3976   \else
3977     \cslet{\@glo@type@\@glo@type @title}{\glossarytitle}%
3978   \fi
3979   \let\currentglossary\@glo@type
3980   \let\org@glossaryentrynumbers\glossaryentrynumbers
3981   \let\glsnonextpages\@glsnonextpages
3982   \let\glsnextpages\@glsnextpages

3983   \glsxtractivatenopost
3984   \gls@dotoc@title
3985   \@glossarystyle
3986   \let\gls@org@glossaryentryfield\glossentry
3987   \let\gls@org@glossarysubentryfield\subglossentry
3988   \renewcommand{\glossentry}[1]{%
3989     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3990     \gls@org@glossaryentryfield{##1}%
3991   }%
3992   \renewcommand{\subglossentry}[2]{%
3993     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3994     \gls@org@glossarysubentryfield{##1}{##2}%
3995   }%
3996   \@gls@preglossaryhook
3997   #2%
3998 \egroup
3999 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
4000 \global\let\warn@noprntglossary\relax
4001 }

```

ractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

4002 \newcommand*{\glsxtractivatenopost}{%
4003   \let\nopostdesc\@nopostdesc
4004   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4005 }

```

lsxtrnopostpunc

```
4006 \newrobustcmd*{\glxtrnopostpunc}{}
```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```
4007 \newcommand{\@glxtr@nopostpunc}{%
4008 \let\@glxtr@org@postdescription\glspostdescription
4009 \ifglsnopostdot
4010 \renewcommand{\glspostdescription}{%
4011 \glsnopostdottrue
4012 \let\glspostdescription\@glxtr@org@postdescription
4013 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4014 \glxtrpostdescription
4015 \@glxtr@nopostpunc@postdesc}%
4016 \else
4017 \renewcommand{\glspostdescription}{%
4018 \let\glspostdescription\@glxtr@org@postdescription
4019 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4020 \glxtrpostdescription
4021 \@glxtr@nopostpunc@postdesc}%
4022 \fi
4023 \glsnopostdotfalse
4024 }
```

stpunc@postdesc

```
4025 \newcommand*{\@glxtr@nopostpunc@postdesc}{}
```

estore@postpunc

```
4026 \newcommand*{\@glxtr@restore@postpunc}{%
4027 \def\@glxtr@nopostpunc@postdesc{%
4028 \@glxtr@org@postdescription
4029 \let\@glxtr@nopostpunc@postdesc\@empty
4030 \let\glxtrrestorepostpunc\@empty
4031 }%
4032 }
```

restorepostpunc Does nothing outside of glossary.

```
4033 \newcommand*{\glxtrrestorepostpunc}{}
```

\@printglossary Redefine.

```
4034 \renewcommand{\@printglossary}[2]{%
4035 \def\@glxtr@printglossopts{#1}%
4036 \@glxtr@orgprintglossary{#1}{#2}%
4037 }
```

Add a key that switches off the entry targets:

```
4038 \define@choicekey{printgloss}{target}
4039 [\@glxtr@printglossval\@glxtr@printglossnr]%
```



```

4040 {true,false}[true]%
4041 {%
4042   \ifcase\@glxtr@printglossnr

4043   \def\@glstarget{\glsdohypertarget}%
4044   \else
4045   \let\@glstarget\@secondoftwo
4046   \fi
4047 }

```

hypernameprefix

```

4048 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:

4049 \define@key{printgloss}{targetnameprefix}{%
4050   \renewcommand{\@glxtrhypernameprefix}{#1}%
4051 }

4052 \define@key{printgloss}{prefix}{%
4053   \renewcommand{\glo@linkprefix}{#1}%
4054 }

```

glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```

4055 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4056 \renewcommand{\glsdohypertarget}[2]{%
4057   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4058 }

```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```

4059 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4060 \def\@glstarget{\glsdohypertarget}%
4061 \fi
4062 %\end{macro}

```

@makeglossaries For the benefit of makeglossaries

```

4063 \newcommand*{\@glxtr@makeglossaries}[1]{}

```

@glxtr@gettype Get just the type.

```

4064 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4065   \def\@glo@type{#2}%
4066 }

```

@assign@sortkey Assign the sort key.

```

4067 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4068   \edef\@glo@type{\@glo@type}%
4069   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4070   {%
4071     \@glo@no@assign@sortkey{#1}%

```

```

4072 }%
4073 {%
4074   \@glo@assign@sortkey{#1}%
4075 }%
4076 }%

```

Display number list for the regular version:

splaynumberlist

```

4077 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

4078 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4079   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4080   \ifdef\@gls@loclist
4081     {%
4082       \def\@gls@noidx@loclist@sep{%
4083         \def\@gls@noidx@loclist@sep{%
4084           \def\@gls@noidx@loclist@sep{%
4085             \glsnumlistsep
4086           }%
4087           \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4088         }%
4089       }%
4090       \def\@gls@noidx@loclist@finalsep{}%
4091       \def\@gls@noidx@loclist@prev{}%
4092       \forlistloop{\@gls@noidx@displayloclisthandler}{\@gls@loclist}%
4093       \@gls@noidx@loclist@finalsep
4094       \@gls@noidx@loclist@prev
4095     }%
4096     {%
4097       \glxtrundeftag
4098       \glsdoifexists{#1}%
4099       {%
4100         \GlossariesWarning{Missing location list for ‘#1’. Either
4101           a rerun is required or you haven’t referenced the entry.}%
4102       }%
4103     }%
4104 }%
4105

```

And for the number list loop:

@numberlistloop

```

4106 \newcommand*{\@glxtr@noidx@numberlistloop}[3]{%
4107   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4108   \let\@gls@org\@glsnoidxdisplayloc\@glsnoidxdisplayloc
4109   \let\@gls@org\@glsseeformat\@glsseeformat

```

```

4110 \let\glsnoidxdisplayloc#2\relax
4111 \let\glsseeformat#3\relax
4112 \ifdef\@gls@loclist
4113 {%
4114   \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4115 }%
4116 {%

4117   \glsxtrundeftag
4118   \glsdoifexists{#1}%
4119   {%
4120     \GlossariesWarning{Missing location list for ‘##1’. Either
4121       a rerun is required or you haven’t referenced the entry.}%
4122   }%
4123 }%
4124 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4125 \let\glsseeformat\@gls@org@glsseeformat
4126 }%

```

Same for entry number list.

entrynumberlist

```

4127 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4128   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4129   \ifdef\@gls@loclist
4130   {%
4131     \glsnoidxloclist{\@gls@loclist}%
4132   }%
4133   {%

4134     \glsxtrundeftag
4135     \glsdoifexists{#1}%
4136     {%
4137       \GlossariesWarning{Missing location list for ‘#1’. Either
4138         a rerun is required or you haven’t referenced the entry.}%
4139     }%
4140   }%
4141 }%

```

entrynumberlist

```

4142 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4143 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4144   \protected@edef\@glsxtr@titlelabel{#1}%
4145   \ifdefvoid\@glsxtr@titlelabel
4146   {}%
4147   {%
4148     \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4149   }%

```

```

4150 \ifdefvoid{\@glsxtr@titlelabel}%
4151 {%
4152   \DTLifint{#1}%
4153   {%
4154     \ifnum#1<256\relax
4155       \edef#2{\char#1\relax}%
4156     \else
4157       \edef#2{#1}%
4158     \fi
4159   }%
4160   {%
4161     \ifcsundef{#1groupname}%
4162     {\def#2{#1}}%
4163     {\letcs#2{#1groupname}}%
4164   }%
4165 }%
4166 {%
4167   \let#2\@glsxtr@titlelabel
4168 }%
4169 }

```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```

4170 \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.

```

4171 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4172   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4173   \@onelevel@sanitize\@glsxtr@titlelabel
4174   \ifcsdef{\@glsxtr@titlelabel}
4175   {\letcs{#2}{\@glsxtr@titlelabel}}%
4176   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4177 }
4178 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

4179 \newcommand{\glsxtrsetgrouptitle}[2]{%
4180   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4181   \@onelevel@sanitize\@glsxtr@titlelabel
4182   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4183 }

```

alsetgrouptitle As above put only locally defines the title.

```

4184 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4185   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4186   \@onelevel@sanitize\@glsxtr@titlelabel
4187   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4188 }

```

`\glsnavigation` Redefine to use new user-level command.

```

4189 \renewcommand*{\glsnavigation}{%
4190   \def\@gls@between{}%
4191   \ifcsundef\@gls@hypergroup\list\@glo@type}%
4192   {%
4193     \def\@gls@list{}%
4194   }%
4195   {%
4196     \expandafter\let\expandafter\@gls@list
4197       \csname @gls@hypergroup\list\@glo@type\endcsname
4198   }%
4199   \@for\@gls@tmp:=\@gls@list\do{%
4200     \@gls@between
4201     \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4202     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4203     \let\@gls@between\glshypernavsep
4204   }%
4205 }
```

`@noidx@glossary`

```

4206 \renewcommand*{\@print@noidx@glossary}{%
4207   \ifcsdef\@glsref\@glo@type}%
4208   {%
4209     \ifcsdef\@glo@sortmacro\@glo@sorttype}%
4210     {%
4211       \csuse\@glo@sortmacro\@glo@sorttype\@glo@type}%
4212     }%
4213     {%
4214       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4215     }%
4216     \glossarysection[\glossarytoctitle]{\glossarytitle}%
4217     \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4218   \def\@gls@currentlettergroup{}%
4219   \begin{theglossary}%
4220     \glossaryheader
4221     \glsresetentrylist
4222     \forlistcsloop{\@gls@noidx@do}{\@glsref\@glo@type}%
4223     \end{theglossary}%
4224     \glossarypostamble
4225   }%
4226   {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4227   \glsxtrifemptyglossary{\@glo@type}%
4228   {}%
4229   {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
```

```

4230    \@gls@noref@warn{\@glo@type}%
4231  }%
4232 }

```

noidxdisplayloc Patch to check for range formations.

```

4233 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4234   \setentrycounter[#1]{#2}%
4235   \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4236 }

```

xtr@display@loc Patch to check for range formations.

```

4237 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4238   \ifx#1\relax
4239     \glsxtrdisplaystartloc{#2}{#3}%
4240   \else
4241     \ifx#1\relax
4242       \glsxtrdisplayendloc{#2}{#3}%
4243     \else
4244       \glsxtrdisplaysingleloc{#1#2}{#3}%
4245     \fi
4246   \fi
4247 }

```

isplaysingleloc Single location.

```

4248 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4249   \csuse{#1}{#2}%
4250 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```

4251 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4252   \edef\glsxtrlocrangefmt{#1}%
4253   \ifx\glsxtrlocrangefmt\empty
4254     \def\glsxtrlocrangefmt{glsnumberformat}%
4255   \fi
4256   \expandafter\glsxtrdisplaysingleloc
4257   \expandafter{\glsxtrlocrangefmt}{#2}%
4258 }

```

trdisplayendloc End of a location range.

```

4259 \newcommand*{\glsxtrdisplayendloc}[2]{%
4260   \edef\@glsxtr@tmp{#1}%
4261   \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}}%
4262   \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4263   \else
4264     \GlossariesExtraWarning{Mismatched end location range
4265       (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%

```

```

4266 \fi
4267 \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
4268 \expandafter\glxtrdisplaysingleloc
4269 \expandafter{\glxtrlocrangefmt}{#2}%
4270 \def\glxtrlocrangefmt{%
4271 }

```

`\displayendlochook` Allow the user to hook into the end of range command.

```

4272 \newcommand*{\glxtrdisplayendlochook}[2]{%

```

`\glxtrlocrangefmt` Current range format. Empty if not in a range.

```

4273 \newcommand*{\glxtrlocrangefmt}{%

```

`\setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4274 \renewcommand*{\setentrycounter}[2][{}]{%
4275 \def\glxtrcounterprefix{#1}%
4276 \ifx\glxtrcounterprefix\@empty
4277 \def\@glo@counterprefix{.}%
4278 \else
4279 \def\@glo@counterprefix{.#1.}%
4280 \fi
4281 \def\glsetentrycounter{#2}%
4282 }

```

`\ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4283 \def\@gls@removespaces#1 #2\@nil{%
4284 \toks@=\expandafter{\the\toks@#1}%
4285 \ifx\@#2\%
4286 \edef\x{\the\toks@}%
4287 \ifx\x\empty
4288 \else
4289 \expandafter\glxtrlocationhyperlink\expandafter
4290 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4291 \fi
4292 \else
4293 \@gls@ReturnAfterFi{%
4294 \@gls@removespaces#2\@nil
4295 }%
4296 \fi
4297 }

```

`\locationhyperlink`

```

4298 \newcommand*{\glxtrlocationhyperlink}[3]{%
4299 \ifdefined\glxtrsupplocationurl
4300 {%
4301 \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4302 }%

```

```

4303  {%
4304    \hyperref{\glxtrsupplocationurl}{\#1\#2\#3}{\#3}%
4305  }%
4306 }

```

supphypernumber

```

4307 \newcommand*{\glxtrsupphypernumber}[1]{%
4308  {%
4309    \glshasattribute{\glscurrententrylabel}{externallocation}%
4310    {%
4311      \def\glxtrsupplocationurl{%
4312        \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4313    }%
4314    {%
4315      \def\glxtrsupplocationurl{}%
4316    }%
4317    \glshypernumber{\#1}%
4318  }%
4319 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4320 \renewcommand{\@print@glossary}{%
4321  \makeatletter
4322  \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4323  \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4324  {}%
4325  {\glxtrNoGlossaryWarning{\@glo@type}}%
4326  \ifglxindy
4327    \ifcsundef{\xdy@\@glo@type @language}%
4328    {%
4329      \edef\@do@auxoutstuff{%
4330        \noexpand\AtEndDocument{%
4331          \noexpand\immediate\noexpand\write\@auxout{%
4332            \string\providecommand\string\@xdylanguage[2]{}}%
4333          \noexpand\immediate\noexpand\write\@auxout{%
4334            \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4335        }%
4336      }%
4337    }%
4338    {%
4339      \edef\@do@auxoutstuff{%
4340        \noexpand\AtEndDocument{%
4341          \noexpand\immediate\noexpand\write\@auxout{%
4342            \string\providecommand\string\@xdylanguage[2]{}}%
4343          \noexpand\immediate\noexpand\write\@auxout{%
4344            \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type

```



```

4345         @language\endcsname}}}%
4346     }%
4347 }%
4348 }%
4349 \do@auxoutstuff
4350 \edef\do@auxoutstuff{%
4351     \noexpand\AtEndDocument{%
4352         \noexpand\immediate\noexpand\write\@auxout{%
4353             \string\providecommand\string\@gls@codepage[2]{}}}%
4354         \noexpand\immediate\noexpand\write\@auxout{%
4355             \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
4356     }%
4357 }%
4358 \do@auxoutstuff
4359 \fi
4360 \renewcommand*{\@warn@nomakeglossaries}{%
4361     \GlossariesWarningNoLine{\string\makeglossaries\space
4362     hasn't been used,^^Jthe glossaries will not be updated}%
4363 }%
4364 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4365 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4366     This document is incomplete. The external file associated with
4367     the glossary '#1' (which should be called \texttt{#2})
4368     hasn't been created.%
4369 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4370 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4371     This has probably happened because there are no entries defined
4372     in this glossary.%
4373 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4374 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4375     If you don't want this glossary,
4376     add \texttt{nomain} to your package option list when you load
4377     \texttt{glossaries-extra.sty}. For example:%
4378 }

```

`\GlsWarningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4379 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4380     Did you forget to use \texttt{type=#1} when you defined your
4381     entries? If you tried to load entries into this glossary with
4382     \texttt{\string\loadglsentries} did you remember to use
4383     \texttt{[#1]} as the optional argument? If you did, check that

```

```

4384 the definitions in the file you loaded all had the type set
4385 to \texttt{\string\glsdefaulttype}%.
4386 }

```

WarningCheckFile Advisory message to check the file contents.

```

4387 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4388   Check the contents of the file \texttt{#1}. If
4389   it's empty, that means you haven't indexed any of your entries in this
4390   glossary (using commands like \texttt{\string\gls} or
4391   \texttt{\string\glsadd}) so this list can't be generated.
4392   If the file isn't empty, the document build process hasn't been
4393   completed.
4394 }

```

WarningAutoMake Message when automake option has been used.

```

4395 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4396   You may need to rerun \LaTeX. If you already have, it may be that
4397   \TeX's shell escape doesn't allow you to run
4398   \ifglxindy xindy\else makeindex\fi. Check the
4399   transcript file \texttt{\jobname.log}. If the shell escape is
4400   disabled, try one of the following:
4401
4402   \begin{itemize}
4403     \item Run the external (Lua) application:
4404
4405       \texttt{makeglossaries-lite.lua \string\jobname\string}
4406
4407     \item Run the external (Perl) application:
4408
4409       \texttt{makeglossaries \string\jobname\string}
4410   \end{itemize}
4411
4412   Then rerun \LaTeX\ on this document.
4413   \GlossariesExtraWarning{Rerun required to build the
4414   glossary '#1' or check \TeX's shell escape allows
4415   you to run \ifglxindy xindy\else makeindex\fi}%
4416 }

```

WarningMisMatch Mismatching \makenoidxglossaries.

```

4417 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4418   You need to either replace \texttt{\string\makenoidxglossaries}
4419   with \texttt{\string\makeglossaries} or replace
4420   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4421   \texttt{\string\printnoidxglossary}
4422   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4423   this document.
4424 }

```

arningBuildInfo Build advice.

```
4425 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4426   Try one of the following:
4427   \begin{itemize}
4428     \item Add \texttt{automake} to your package option list when you load
4429       \texttt{glossaries-extra.sty}. For example:
4430
4431       \texttt{\string\usepackage[automake]%
4432         \glsopenbrace glossaries-extra\glsclosebrace}
4433
4434     \item Run the external (Lua) application:
4435
4436     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4437
4438     \item Run the external (Perl) application:
4439
4440     \texttt{makeglossaries \string"\jobname\string"}
4441   \end{itemize}
4442
4443   Then rerun \LaTeX\ on this document.%
4444 }
```

trRecordWarning Paragraph for record=only.

```
4445 \newcommand{\GlsXtrRecordWarning}[1]{%
4446   \texttt{\string\printglossary} doesn't work
4447   with the \texttt{record=only} package option
4448   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4449   instead (or change the package option).%
4450 }
```

oGlsWarningTail Final paragraph.

```
4451 \newcommand{\GlsXtrNoGlsWarningTail}{%
4452   This message will be removed once the problem has been fixed.%
4453 }
```

GlsWarningNoOut No out file created. Build advice.

```
4454 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4455   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4456   \texttt{\string\makeglossaries} or you have used
4457   \texttt{\string\nofiles}. If this is just a draft version of the
4458   document, you can suppress this message using the
4459   \texttt{nomissingglstext} package option.%
4460 }
```

glossarywarning

```
4461 \newcommand*{@@glsxtr@defaultnoglossarywarning}[1]{%
4462   \glossarysection[\glossarytoctitle]{\glossarytitle}
4463   \GlsXtrNoGlsWarningHead{#1}{\jobname.\cstype @glo@type @in\endcsname}
4464   \par
```

```

4465 \glstrifemptyglossary{#1}%
4466 {%
4467     \GlsXtrNoGlsWarningEmptyStart\space
4468     \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4469     \medskip
4470     \noindent\texttt{\string\usepackage[nomain\ifglssacronym ,acronym\fi]%
4471         \glsoopenbrace glossaries-extra\glsclosebrace}
4472     \medskip
4473     }%
4474     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4475 }%
4476 {%
4477     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4478     {%
4479         \GlsXtrNoGlsWarningCheckFile
4480         {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4481
4482         \ifglssautomake
4483
4484         \GlsXtrNoGlsWarningAutoMake{#1}
4485
4486     \else
4487
4488         \ifthenelse{\equal{#1}{main}}{%
4489             {%
4490                 \GlsXtrNoGlsWarningEmptyMain\par
4491                 \medskip
4492                 \noindent\texttt{\string\usepackage[nomain]%
4493                     \glsoopenbrace glossaries-extra\glsclosebrace}
4494                 \medskip
4495             }%
4496             {}%
4497
4498             \ifdefequal\makeglossaries\@no@makeglossaries
4499             {%
4500                 \GlsXtrNoGlsWarningMisMatch
4501             }%
4502             {%
4503                 \GlsXtrNoGlsWarningBuildInfo
4504             }%
4505             \fi
4506         }%
4507     }%
4508     \GlsXtrNoGlsWarningNoOut
4509     {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4510     }%
4511 }%
4512 \par
4513 \GlsXtrNoGlsWarningTail

```

4514 }

glossarywarning Warn about using `\printglossary` with `record`

```
4515 \newcommand*{\@glstr@record@noglossarywarning}[1]{%
4516   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4517   with record=only package option\MessageBreak(use
4518   \string\printunsrtglossary[type=#1])\MessageBreak
4519   instead (or change the package option)}%
4520   \glossarysection[\glossarytoctitle]{\glossarytitle}
4521   \GlsXtrRecordWarning{#1}
4522   \GlsXtrNoGlsWarningTail
4523 }
```

Provide some commands to accompany the `record` option for use with **bib2gls**.

glstrresourcefile 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.

```
4524 \newcommand*{\glstrresourcefile}[2] [] {%
```

The `record` option can't be set after this command.

```
4525   \disable@keys{glossaries-extra.sty}{record}%
4526   \glstr@writefields
4527   \protected@write\@auxout{\glstrresourceinit}{\string\glstr@resource{#1}{#2}}%
4528   \let\@glstr@org@see@noindex\@glstr@see@noindex
4529   \let\@glstr@see@noindex\relax
4530   \IfFileExists{#2.glstex}%
4531   {%
```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```
4532   \edef\@bibglstr@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4533   \makeatletter
4534   \@input{#2.glstex}%
4535   \@bibglstr@restreat
4536   }%
4537   {%
4538   \GlossariesExtraWarning{No file '#2.glstex'}%
4539   }%
4540   \let\@glstr@see@noindex\@glstr@org@see@noindex
4541 }
4542 \@onlypreamble\glstrresourcefile
```

glstrresourceinit Code used during the protected write operation.

```
4543 \newcommand*{\glstrresourceinit}{}
```

glstrresourcecount

```
4544 \newcount\glstrresourcecount
```

glstrLoadResources Short cut that uses `\glstrresourcefile` with `\jobname` as the mandatory argument.

```

4545 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
4546   \ifnum\glstrresourcecount=0\relax
4547     \glstrresourcefile[#1]{\jobname}%
4548   \else
4549     \glstrresourcefile[#1]{\jobname-\the\glstrresourcecount}%
4550   \fi
4551   \advance\glstrresourcecount by 1\relax
4552 }

glstr@resource
4553 \newcommand*{\glstr@resource}[2]{%

\glstr@fields
4554 \newcommand*{\glstr@fields}[1]{%

xtr@texencoding
4555 \newcommand*{\glstr@texencoding}[1]{%

\glstr@langtag
4556 \newcommand*{\glstr@langtag}[1]{%

@pluralsuffixes
4557 \newcommand*{\glstr@pluralsuffixes}[4]{%

tr@shortcutsval
4558 \newcommand*{\glstr@shortcutsval}[1]{%

sctr@linkprefix
4559 \newcommand*{\glstr@linkprefix}[1]{%

xtr@writefields This information only needs to be written once, so disable it after it's been used.
4560 \newcommand*{\glstr@writefields}{%

4561   \protected@write\@auxout{}%
4562     {\string\providecommand*{\string\glstr@fields}[1]{}}%
4563   \protected@write\@auxout{}%
4564     {\string\providecommand*{\string\glstr@resource}[2]{}}%
4565   \protected@write\@auxout{}%
4566     {\string\providecommand*{\string\glstr@pluralsuffixes}[4]{}}%
4567   \protected@write\@auxout{}%
4568     {\string\providecommand*{\string\glstr@shortcutsval}[1]{}}%
4569   \protected@write\@auxout{}%
4570     {\string\providecommand*{\string\glstr@linkprefix}[1]{}}%
4571   \protected@write\@auxout{}{\string\glstr@fields{\@gls@keymap}}%

4572   \protected@write\@auxout{}%
4573     {\string\providecommand*{\string\glstr@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`.

```

4574 \ifdef\CurrentTrackedLanguageTag
4575 {%
4576   \protected@write\@auxout{}\string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
4577 }%
4578 {}%
4579 \protected@write\@auxout{}\string\glxtr@pluralsuffixes
4580   {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4581   {\glxtrabbrvpluralsuffix}}%
4582 \ifdef\inputencodingname
4583 {%
4584   \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
4585 }%
4586 {}%
4587 {}%
```

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.)

```

4588 \ifpackageloaded{fontspec}%
4589   {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}%
4590   {}%
4591 }%
4592 \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.

```

4593 \AtBeginDocument
4594   {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}%
4595   \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.

```

4596 \ifglautomake
4597   \IfFileExists{\jobname.aux}%
4598   {\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`.

```

4599 \ifx\gls@doautomake\gls@doautomake@err
4600   \let\gls@doautomake\relax
4601 \fi
4602 \fi
4603 }
```

`do@automake@err`

```

4604 \newcommand*{\@gls@doautomake@err}{%
4605   \PackageError{glossaries}{You must use
```

```

4606 \string\makeglossaries\space with automake=true}
4607 {%
4608     Either remove the automake=true setting or
4609     add \string\makeglossaries\space to your document preamble.%
4610 }%
4611 }

```

Allow locations specific to a particular counter to be recorded.

\glxtr@record

```

4612 \newcommand*{\glxtr@record}[5]{}

```

r@counterrecord Aux file command.

```

4613 \newcommand*{\glxtr@counterrecord}[3]{%
4614     \glxtrfieldlistgadd{#1}{record.#2}{#3}%
4615 }

```

unterrecordhook Hook used by \@glxtr@dorecord.

```

4616 \newcommand*{\@glxtr@counterrecordhook}{}

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

4617 \newcommand*{\GlsXtrRecordCounter}[1]{%
4618     \@glxtr@recordcounter{#1}%
4619 }
4620 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

4621 \newcommand*{\@glxtr@docounterrecord}[1]{%
4622     \protected@write\@auxout{}\string\glxtr@counterrecord
4623         {\@glxtr@label}{#1}{\csuse{the#1}}}%
4624 }

```

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.

```

4625 \newcommand*{\glxtrglossentry}[1]{%
4626     \glxtrtitleorpdforheading
4627     {\@glxtrglossentry{#1}}%
4628     {\glsentryname{#1}}%
4629     {\glxtrheadname{#1}}%
4630 }

```

lsxtrglossentry Another test is needed in case \@glxtrglossentry has been written to the table of contents.

```

4631 \newrobustcmd*{\@glxtrglossentry}[1]{%

```



```

4632 \glxtrtitleorpdforheading
4633 {%
4634   \glsdoifexists{#1}%
4635   {%
4636     \begingroup
4637     \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4638     \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4639     \ifglshasparent{#1}%
4640     {\GlsXtrStandaloneSubEntryItem{#1}}%
4641     {\glsenentryitem{#1}}%
4642     \glstarget{#1}{\glossentryname{#1}}%
4643   \endgroup
4644   }%
4645 }%
4646 {\glsenentryname{#1}}%
4647 {\glxtrheadname{#1}}%
4648 }

```

oneGlossaryType To make it easier to adjust the definition of `\currentglossary` within `\glxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4649 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsenentrytype{\glscurrententrylabel}}

```

oneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```

4650 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4651   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}}%
4652 }

```

glossentryother As `\glxtrglossentry` but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```

4653 \newcommand*{\glxtrglossentryother}[3]{%
4654   \ifstrempy{#1}%
4655   {%
4656     \ifcsdef{glxtrhead#3}%
4657     {%
4658       \glxtrtitleorpdforheading
4659       {\@glxtrglossentryother{#2}{#3}{#1}}%
4660       {\@gls@entry@field{#2}{#3}}%
4661       {\csuse{glxtrhead#3}{#2}}%
4662     }%
4663     {%
4664       \glxtrtitleorpdforheading
4665       {\@glxtrglossentryother{#2}{#3}{#1}}%
4666       {\@gls@entry@field{#2}{#3}}%
4667       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4668     }%

```

```

4669 }%
4670 {%
4671   \glstrtitleorpdforheading
4672   {\@glstrglossentryother{#2}{#3}{#1}}%
4673   {\@gls@entry@field{#2}{#3}}%
4674   {#1}%
4675 }%
4676 }

```

`glossentryother` As `\@glstrglossentry` but uses a different field.

```

4677 \newrobustcmd*{\@glstrglossentryother}[3]{%
4678   \glstrtitleorpdforheading
4679   {%
4680     \glsoifexists{#1}%
4681     {%
4682       \begingroup
4683       \edef\glscurrententrylabel{\glsetoklabel{#1}}%
4684       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4685       \ifglshasparent{#1}%
4686       {\GlsXtrStandaloneSubEntryItem{#1}}%
4687       {\glentryitem{#1}}%
4688       \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4689       \endgroup
4690     }%
4691   }%
4692   {\@gls@entry@field{#1}{#2}}%
4693   {#3}%
4694 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4695 \newcommand*{\printunsrtglossary}{%
4696   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4697 }

```

`ntunsrtglossary` Unstarred version.

```

4698 \newcommand*{\@printunsrtglossary}[1][ ]{%
4699   \@printglossary{type=\glstdefaulttype,#1}{\@print@unsrt@glossary}%
4700 }

```

`ntunsrtglossary` Starred version.

```

4701 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4702   \begingroup
4703     #2%
4704     \@printglossary{type=\glstdefaulttype,#1}{\@print@unsrt@glossary}%
4705   \endgroup
4706 }

```

`\unsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```
4707 \newcommand*{\printunsrtglossaries}{%
4708   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4709 }
```

`@unsrt@glossary`

```
4710 \newcommand*{\@print@unsrt@glossary}{%
4711   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4712   \glossarypreamble
4713   check for empty list
4714   \glstrifemptyglossary{\@glo@type}%
4715   {%
4716     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4717   }%
4718   \key@ifundefined{glossentry}{group}%
4719   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4720   {\let\@gls@getgrouptitle\@glstr@unsrt@getgrouptitle}%
4721   \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
4722 \def\@glstr@doglossary{%
4723   \begin{theglossary}%
4724   \glossaryheader
4725   \glsresetentrylist
4726   }%
4727   \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4728   : \expandafter=\csname glo@list@\@glo@type\endcsname\do{%
4729     \ifdefempty{\glscurrententrylabel}
4730     {}%
4731     {%
```

Provide a hook (for example to measure width).

```
4732     \let\glstr@process\@firstofone
4733     \let\printunsrtglossaryskipentry
4734         \glstr@printunsrtglossaryskipentry
4735     \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```
4736     \glstr@process
4737     {%
4738     \ifglshasparent{\glscurrententrylabel}{}%
4739     {%
4740     \glstr@checkgroup\glscurrententrylabel
4741     \expandafter\appto\expandafter\@glstr@doglossary\expandafter
4742     {\@glstr@groupheading}%
4743     }%
```

```

4744         \eappto\@glxtr@doglossary{%
4745         \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4746     }%
4747 }%
4748 }%
4749 \appto\@glxtr@doglossary{\end{theglossary}}%
4750 \printunsrtglossarypredoglossary
4751 \@glxtr@doglossary
4752 }%
4753 \glossarypostamble
4754 }

```

entryprocesshook

```

4755 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

glossaryskipentry

```

4756 \newcommand*{\printunsrtglossaryskipentry}{%
4757   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4758   can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4759 }

```

entryprocesshook

```

4760 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
4761   \let\glxtr@process\@gobble
4762 }

```

glossarypredoglossary

```

4763 \newcommand*{\printunsrtglossarypredoglossary}{}

```

glossary@handler

```

4764 \newcommand{\@printunsrt@glossary@handler}[1]{%
4765   \xdef\glscurrententrylabel{#1}%
4766   \printunsrtglossaryhandler\glscurrententrylabel
4767 }

```

glossaryhandler

```

4768 \newcommand{\printunsrtglossaryhandler}[1]{%
4769   \glxtrunsrtdo{#1}%
4770 }

```

glxtriflabelinlist

```

\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 `\@glx@ifinlist` which ensures the label and list are fully expanded.

```

4771 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4772   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@glx@ifinlist{#1}{#2}}%
4773   \@glxtr@doiflabelinlist{#3}{#4}%
4774 }

```

srtglossaryunit

```

4775 \newcommand{\print@op@unsrtglossaryunit}[2][{}]{%
4776   \s@printunsrtglossary[type=\glstypedefaulttype,#1]{%
4777     \printunsrtglossaryunitsetup{#2}%
4778   }%
4779 }

```

ossaryunitsetup

```

4780 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4781   \renewcommand{\printunsrtglossaryhandler}[1]{%
4782     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4783     {\glxtrunsrtdo{##1}}%
4784   }%
4785 }%

```

Only the target names should have the prefixes adjusted as \glx etc need the original \glolinkprefix. The \@gobble part discards \glolinkprefix.

```

4786 \ifcsundef{theH#1}%
4787 {%
4788   \renewcommand*{\@glxtr@hypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4789 }%
4790 {%
4791   \renewcommand*{\@glxtr@hypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4792 }%
4793 \renewcommand*{\glossarysection}[2][{}]{%
4794   \appto\glossarypostamble{\glspare\medskip\glspare}%
4795 }

```

srtglossaryunit

```

4796 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
4797   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4798     requires the record=only or record=alsoindex package option}{}%
4799 }

```

t@getgrouptitle

```

4800 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4801   \protected@edef\@glxtr@titlelabel{\glxtr@grouptitle@#1}%
4802   \@onelevel@sanitize\@glxtr@titlelabel
4803   \ifcsdef{\@glxtr@titlelabel}
4804   {\letcs{#2}{\@glxtr@titlelabel}}%
4805   {\def#2{#1}}%
4806 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

4807 \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).

```
4808 \newcommand*{\glsxtrgroupfield}{group}
```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while `\@glsxtr@doglossary` is being constructed rather than in the handler.

`lsxtr@checkgroup` The argument is the entry's label. (This block of code was formerly in `\@glsxtr@noidx@do`.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in `\@glsxtr@groupheading`, which will be empty if no heading is required.

```
4809 \newcommand*{\@glsxtr@checkgroup}[1]{%
4810   \def\@glsxtr@groupheading{}%
4811   \key@ifundefined{glossentry}{group}%
4812   {%
4813     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
4814     \expandafter\glo@grabfirst\@gls@sort{}\}\@nil
4815   }%
4816   {%
4817     \protected@edef\@glo@thislettergrp{%
4818       \csuse{glo@\glsdetoklabel{#1}@\glsxtrgroupfield}}%
4819     }%
4820     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4821     {}%
4822     {%
4823       \ifdefempty{\@gls@currentlettergroup}{}%
4824       {\def\@glsxtr@groupheading{\glsgroupskip}}%
4825       \eappto\@glsxtr@groupheading{%
4826         \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
4827       }%
4828     }%
4829     \let\@gls@currentlettergroup\@glo@thislettergrp
4830 }
```

`glsxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```
4831 \newcommand{\@glsxtr@noidx@do}[1]{%
4832   \ifglsentryexists{#1}%
4833   {%
4834     \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4835     \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
4836     \ifglshasparent{#1}%
4837     {%
4838       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
4839       \ifdefvoid{\@gls@location}%

```

```

4840     {%
4841         \ifdefvoid{\@gls@loclist}%
4842         {%
4843             \subglossentry{\gls@level}{#1}{}%
4844         }%
4845         {%
4846             \subglossentry{\gls@level}{#1}%
4847             {%
4848                 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4849             }%
4850         }%
4851     }%
4852     {%
4853         \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4854     }%
4855 }%
4856 {%

4857     \ifdefvoid{\@gls@location}%
4858     {%
4859         \ifdefvoid{\@gls@loclist}
4860         {%
4861             \glossentry{#1}{}%
4862         }%
4863         {%
4864             \glossentry{#1}%
4865             {%
4866                 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4867             }%
4868         }%
4869     }%
4870     {%
4871         \glossentry{#1}%
4872         {%
4873             \glossaryentrynumbers{\@gls@location}%
4874         }%
4875     }%
4876 }%
4877 }%
4878 {}%
4879 }

```

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`

```
4880 \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>}`

```

4881 \newcommand*{\@glsxtrnewgls}[4]{%
4882   \ifdef{#3}%
4883   {%
4884     \PackageError{glossaries-extra}{Command \string#3\space already
4885 defined}{}%
4886   }%
4887   {%
4888     \ifcsdef{@#4like@#2}%
4889     {%
4890       \advance\@glsxtrnewgls@inner by \@ne
4891       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4892     }%
4893     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4894     \expandafter\newrobustcmd\expandafter*\expandafter
4895       #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4896     \ifstrempy{#1}%
4897     {%
4898       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
4899         \new@ifnextchar[%
4900           {\csname @#4@\endcsname{##1}{#2##2}}%
4901           {\csname @#4@\endcsname{##1}{#2##2} [ ]}%
4902         }%
4903       }%
4904     }%
4905     \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
4906       \new@ifnextchar[%
4907         {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4908         {\csname @#4@\endcsname{#1,##1}{#2##2} [ ]}%
4909       }%
4910     }%
4911   }%
4912 }

```

`\glsxtrnewgls` `\glsxtrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4913 \newrobustcmd*{\glsxtrnewgls}[3][ ]{%
4914   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4915 }

```


`\lsxtrnewglslike` 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.

```
4916 \newrobustcmd*{\glsxtrnewglslike}[6] [] {%
4917   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4918   \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
4919   \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
4920   \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
4921 }
```

`\lsxtrnewGLSlike` 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.

```
4922 \newrobustcmd*{\glsxtrnewGLSlike}[4] [] {%
4923   \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
4924   \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
4925 }
```

`\glsxtrnewrgls` As `\glsxtrnewgls` but for `\rgls`.

```
4926 \newrobustcmd*{\glsxtrnewrgls}[3] [] {%
4927   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4928 }
```

`\sxtrnewrglslike` As `\glsxtrnewglslike` but for `\rgls` etc.

```
4929 \newrobustcmd*{\glsxtrnewrglslike}[6] [] {%
4930   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4931   \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
4932   \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
4933   \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
4934 }
```

`\sxtrnewrGLSlike` As `\glsxtrnewGLSlike` but for `\rGLS` etc.

```
4935 \newrobustcmd*{\glsxtrnewrGLSlike}[4] [] {%
4936   \@glsxtrnewgls{#1}{#2}{#3}{rGLS}%
4937   \@glsxtrnewgls{#1}{#2}{#4}{rGLSpl}%
4938 }
```

Provide easy access to record count fields.

`\totalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```
4939 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
4940   \ifcsdef{glo@#1\glsdetoklabel{#1}@recordcount}%
4941   {\csname glo@#1\glsdetoklabel{#1}@recordcount\endcsname}%
4942   {0}%
4943 }
```

`\sXtrRecordCount` Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4944 \newcommand*{\GlsXtrRecordCount}[2] {%
```

```

4945 \ifcsdef{glo@glxstrdetoklabel{#1}@recordcount.#2}%
4946 {\csname glo@glxstrdetoklabel{#1}@recordcount.#2\endcsname}%
4947 {0}%
4948 }

```

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 `\glxstrdetoklocation` can be set to something sensible.

```

4949 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4950 \ifcsdef{glo@glxstrdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}}%
4951 {\csname glo@glxstrdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}\endcsname}%
4952 {0}%
4953 }

```

trdetoklocation

```

4954 \newcommand*{\glxstrdetoklocation}[1]{#1}

```

ablerecordcount

```

4955 \newcommand*{\glxstreablerecordcount}{%
4956 \renewcommand*{\gls}{\rgls}%
4957 \renewcommand*{\Gls}{\rGls}%
4958 \renewcommand*{\glspl}{\rglspl}%
4959 \renewcommand*{\Glspl}{\rGlspl}%
4960 \renewcommand*{\GLS}{\rGLS}%
4961 \renewcommand*{\GLSpl}{\rGLSpl}%
4962 }

```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```

4963 \newcommand*{\glxstrrecordtriggervalue}[1]{%
4964 \GlsXtrTotalRecordCount{#1}%
4965 }

```

dCountAttribute

```

4966 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4967 \@for\@glxstr@cat:=#1\do
4968 {%
4969 \ifdefempty{\@glxstr@cat}{}%
4970 {%
4971 \glsssetcategoryattribute{\@glxstr@cat}{recordcount}{#2}%
4972 }%
4973 }%
4974 }

```

rifrecordtrigger

```
\glxstrifrecordtrigger{<label>}{<trigger format>}{<normal>}
```

```

4975 \newcommand*{\glxtrifrecordtrigger}[3]{%
4976   \glshasattribute{#1}{recordcount}%
4977   {%
4978     \ifnum\glxtrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
4979     #3%
4980     \else
4981     #2%
4982     \fi
4983   }%
4984   {#3}%
4985 }

```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```

4986 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
4987   \edef\glslabel{\glsdetoklabel{#2}}%
4988   \let\@gls@link@label\glslabel
4989   \def\@glxtr@thevalue{%
4990     \def\@glxtr@theHvalue{\@glxtr@thevalue}%
4991     \def\@glsnumberformat{glstriggerrecordformat}%
4992     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
4993     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
4994     \def\@glxtr@thevalue{%
4995       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
4996       \glxtrinitwrgloss
4997       \glslinkpresetkeys
4998       \setkeys{glslink}{#1}%
4999       \glslinkpostsetkeys
5000       \ifdefempty{\@glxtr@thevalue}%
5001       {%
5002         \@gls@saveentrycounter
5003       }%
5004       {%
5005         \let\theglentrycounter\@glxtr@thevalue
5006         \def\theHglentrycounter{\@glxtr@theHvalue}%
5007       }%
5008       \ifglxtrinitwrglossbefore
5009         \@do@wrglossary{#2}%
5010       \fi
5011       #3%
5012       \ifglxtrinitwrglossbefore
5013       \else
5014         \@do@wrglossary{#2}%
5015       \fi
5016       \ifKV@glslink@local
5017         \glslocalunset{#2}%
5018       \else
5019         \glsunset{#2}%
5020       \fi

```

```

5021 }

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by
bib2gls.
5022 \newcommand*{\glstriggerrecordformat}[1]{

\rgls
5023 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

\@rgls
5024 \newcommand*{\@rgls}[2][]{%
5025   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5026 }

\@rgls@
5027 \def\@rgls@#1#2[#3]{%
5028   \glstrifrecordtrigger{#2}%
5029   {%
5030     \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5031   }%
5032   {%
5033     \@gls@{#1}{#2}[#3]%
5034   }%
5035 }%

\rglspl
5036 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

\@rglspl
5037 \newcommand*{\@rglspl}[2][]{%
5038   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5039 }

\@rglspl@
5040 \def\@rglspl@#1#2[#3]{%
5041   \glstrifrecordtrigger{#2}%
5042   {%
5043     \@glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5044   }%
5045   {%
5046     \@glspl@{#1}{#2}[#3]%
5047   }%
5048 }%

\rGls
5049 \newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}

```

```

\@rGls
5050 \newcommand*{\@rGls}[2][{}]{%
5051   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5052 }

\@rGls@
5053 \def\@rGls@#1#2[#3]{%
5054   \glstrifrecordtrigger{#2}%
5055   {%
5056     \@glstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5057   }%
5058   {%
5059     \@Gls@{#1}{#2}[#3]%
5060   }%
5061 }%

\rGlspl
5062 \newrobustcmd*{\rGlspl}{\@glshyp@opt\@rGlspl}

\@rGlspl
5063 \newcommand*{\@rGlspl}[2][{}]{%
5064   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5065 }

\@rGlspl@
5066 \def\@rGlspl@#1#2[#3]{%
5067   \glstrifrecordtrigger{#2}%
5068   {%
5069     \@glstr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5070   }%
5071   {%
5072     \@Glspl@{#1}{#2}[#3]%
5073   }%
5074 }%

\rGLS
5075 \newrobustcmd*{\rGLS}{\@glshyp@opt\@rGLS}

\@rGLS
5076 \newcommand*{\@rGLS}[2][{}]{%
5077   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5078 }

\@rGLS@
5079 \def\@rGLS@#1#2[#3]{%
5080   \glstrifrecordtrigger{#2}%
5081   {%
5082     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%

```

```

5083 }%
5084 {%
5085 \@GLS@{#1}{#2}[#3]%
5086 }%
5087 }%

```

\rGLSp1

```
5088 \newrobustcmd*{\rGLSp1}{\@gls@hyp@opt\rGLSp1}
```

\@rGLSp1

```

5089 \newcommand*{\@rGLSp1}[2][{}]{%
5090 \new@ifnextchar[{\@rGLSp1@{#1}{#2}}{\@rGLSp1@{#1}{#2}[{}]}%
5091 }

```

\@rGLSp1@

```

5092 \def\rGLSp1@#1#2[#3]{%
5093 \glsxtrifrecordtrigger{#2}%
5094 {%
5095 \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSp1format{#2}{#3}}%
5096 }%
5097 {%
5098 \@GLSp1@{#1}{#2}[#3]%
5099 }%
5100 }%

```

\rglsformat

```

5101 \newcommand*{\rglsformat}[2]{%
5102 \glsifregular{#1}
5103 {\glsentryfirst{#1}}%
5104 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5105 }

```

\rglsp1format

```

5106 \newcommand*{\rglsp1format}[2]{%
5107 \glsifregular{#1}
5108 {\glsentryfirstplural{#1}}%
5109 {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5110 }

```

\rGlsformat

```

5111 \newcommand*{\rGlsformat}[2]{%
5112 \glsifregular{#1}
5113 {\Glsentryfirst{#1}}%
5114 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5115 }

```

\rGlsplformat

```
5116 \newcommand*{\rGlsplformat}[2]{%
```

```

5117 \glsifregular{#1}
5118 {\Glsentryfirstplural{#1}}%
5119 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5120 }

```

\rGLSformat

```

5121 \newcommand*{\rGLSformat}[2]{%
5122 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5123 }

```

\rGLSplformat

```

5124 \newcommand*{\rGLSplformat}[2]{%
5125 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5126 }

```

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.

```

5127 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
5128 \glsifattribute{\glslabel}{linkcount}{true}%
5129 {%
  Does the counter exist?
5130 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5131 {%
  Counter doesn’t exist, so define it.
5132 \newcounter{glsxtr@linkcount@\glslabel}%
  If linkcountmaster is set, add to counter reset.
5133 \glshasattribute{\glslabel}{linkcountmaster}%
5134 {%
  Need to ensure values are fully expanded.
5135 \begingroup
5136 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
5137 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
5138 \x
5139 }%

```

```

5140     {}%
5141   }%
    Increment counter:
5142   \glxstrinclinkcounter{glxstr@linkcount@glslabel}%
5143 }%
5144 {}%
5145 }

rlinkcounter   May be redefined to use \refstepcounter if required.
5146 \newcommand*{\glxstrinclinkcounter}[1]{\stepcounter{#1}}

linkCounterValue   Expands to the associated link counter register or 0 if not defined.
5147 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5148   \ifcsundef{c@glxstr@linkcount@#1}{0}{\csname c@glxstr@linkcount@#1\endcsname}%
5149 }

rTheLinkCounter   Expands to the display value of the associated link counter or 0 if not defined.
5150 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5151   \ifcsundef{theglsxtr@linkcount@#1}{0}%
5152   {\csname theglxstr@linkcount@#1\endcsname}%
5153 }

fLinkCounterDef   Tests if the counter has been defined
5154 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5155   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5156 }

LinkCounterName   Expands to the associated link counter name. (No check for existence.)
5157 \newcommand*{\GlsXtrLinkCounterName}[1]{glxstr@linkcount@#1}

ableLinkCounting   \GlsXtrEnableLinkCounting[master counter]{categories}

    Enable link counting for the given categories.
5158 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5159   \let\glxstr@inc@linkcount\@glxstr@do@inc@linkcount
5160   \@for\@glxstr@label:=#2\do
5161   {%
5162     \glsssetcategoryattribute{\@glxstr@label}{linkcount}{true}%
5163     \ifstrepty{#1}{}%
5164     {%
5165       \ifcsundef{c@#1}%
5166       {\@nocounterr{#1}}%
5167       {\glsssetcategoryattribute{\@glxstr@label}{linkcountmaster}{#1}}%
5168     }%
5169   }%
5170 }
5171 \@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.

```
5172 \@ifpackageloaded{glossaries-accsupp}
5173 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5174 \newcommand*{\glsaccessname}[1]{%
5175 \glsnameaccessdisplay
5176 {%
5177 \glstryname{#1}%
5178 }%
5179 {#1}%
5180 }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
5181 \newcommand*{\Glsaccessname}[1]{%
5182 \glsnameaccessdisplay
5183 {%
5184 \Glsstryname{#1}%
5185 }%
5186 {#1}%
5187 }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
5188 \newcommand*{\GLSaccessname}[1]{%
5189 \glsnameaccessdisplay
5190 {%
5191 \mfirstucMakeUppercase{\glstryname{#1}}%
5192 }%
5193 {#1}%
5194 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
5195 \newcommand*{\glsaccesstext}[1]{%
5196 \glstextaccessdisplay
5197 {%
5198 \glstrytext{#1}%
5199 }%
5200 {#1}%
5201 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
5202 \newcommand*{\Glsaccesstext}[1]{%
5203   \glstextaccessdisplay
5204   {%
5205     \Glsentrytext{#1}%
5206   }%
5207   {#1}%
5208 }
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```
5209 \newcommand*{\GLSaccesstext}[1]{%
5210   \glstextaccessdisplay
5211   {%
5212     \mfirstucMakeUppercase{\Glsentrytext{#1}}%
5213   }%
5214   {#1}%
5215 }
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```
5216 \newcommand*{\glsaccessplural}[1]{%
5217   \glspluralaccessdisplay
5218   {%
5219     \Glsentryplural{#1}%
5220   }%
5221   {#1}%
5222 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
5223 \newcommand*{\GLSaccessplural}[1]{%
5224   \glspluralaccessdisplay
5225   {%
5226     \Glsentryplural{#1}%
5227   }%
5228   {#1}%
5229 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) converted to upper case.

```
5230 \newcommand*{\GLSaccessplural}[1]{%
5231   \glspluralaccessdisplay
5232   {%
5233     \mfirstucMakeUppercase{\Glsentryplural{#1}}%
5234   }%
5235   {#1}%
5236 }
```

`\glsaccessfirst` Display the first value (no link and no check for existence).

```

5237 \newcommand*{\glsaccessfirst}[1]{%
5238   \glsfirstaccessdisplay
5239   {%
5240     \glstryfirst{#1}%
5241   }%
5242   {#1}%
5243 }

```

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5244 \newcommand*{\Glsaccessfirst}[1]{%
5245   \glsfirstaccessdisplay
5246   {%
5247     \Glsstryfirst{#1}%
5248   }%
5249   {#1}%
5250 }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5251 \newcommand*{\GLSaccessfirst}[1]{%
5252   \glsfirstaccessdisplay
5253   {%
5254     \mfirstucMakeUppercase{\glstryfirst{#1}}%
5255   }%
5256   {#1}%
5257 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5258 \newcommand*{\glsaccessfirstplural}[1]{%
5259   \glsfirstpluralaccessdisplay
5260   {%
5261     \glstryfirstplural{#1}%
5262   }%
5263   {#1}%
5264 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5265 \newcommand*{\Glsaccessfirstplural}[1]{%
5266   \glsfirstpluralaccessdisplay
5267   {%
5268     \Glsstryfirstplural{#1}%
5269   }%
5270   {#1}%
5271 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5272 \newcommand*{\GLSaccessfirstplural}[1]{%

```

```

5273 \glsfirstpluralaccessdisplay
5274 {%
5275     \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5276 }%
5277 {#1}%
5278 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

5279 \newcommand*{\glsaccesssymbol}[1]{%
5280     \glssymbolaccessdisplay
5281     {%
5282         \glsentrysymbol{#1}%
5283     }%
5284     {#1}%
5285 }

```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5286 \newcommand*{\Glsaccesssymbol}[1]{%
5287     \glssymbolaccessdisplay
5288     {%
5289         \glsentrysymbol{#1}%
5290     }%
5291     {#1}%
5292 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5293 \newcommand*{\GLSaccesssymbol}[1]{%
5294     \glssymbolaccessdisplay
5295     {%
5296         \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5297     }%
5298     {#1}%
5299 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5300 \newcommand*{\glsaccesssymbolplural}[1]{%
5301     \glssymbolpluralaccessdisplay
5302     {%
5303         \glsentrysymbolplural{#1}%
5304     }%
5305     {#1}%
5306 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5307 \newcommand*{\Glsaccesssymbolplural}[1]{%
5308     \glssymbolpluralaccessdisplay

```

```

5309   {%
5310       \Glsentrysymbolplural{#1}%
5311   }%
5312   {#1}%
5313 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5314 \newcommand*{\GLSaccesssymbolplural}[1]{%
5315     \glssymbolpluralaccessdisplay
5316     {%
5317         \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5318     }%
5319     {#1}%
5320 }

```

\glsaccessdesc Display the desc value (no link and no check for existence).

```

5321 \newcommand*{\glsaccessdesc}[1]{%
5322     \glsdescriptionaccessdisplay
5323     {%
5324         \glentrydesc{#1}%
5325     }%
5326     {#1}%
5327 }

```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5328 \newcommand*{\Glsaccessdesc}[1]{%
5329     \glsdescriptionaccessdisplay
5330     {%
5331         \Glsentrydesc{#1}%
5332     }%
5333     {#1}%
5334 }

```

\GLSaccessdesc Display the desc value (no link and no check for existence) converted to upper case.

```

5335 \newcommand*{\GLSaccessdesc}[1]{%
5336     \glsdescriptionaccessdisplay
5337     {%
5338         \mfirstucMakeUppercase{\glentrydesc{#1}}%
5339     }%
5340     {#1}%
5341 }

```

ccessdescplural Display the descplural value (no link and no check for existence).

```

5342 \newcommand*{\glsaccessdescplural}[1]{%
5343     \glsdescriptionpluralaccessdisplay
5344     {%
5345         \glentrydescplural{#1}%

```

```

5346 }%
5347 {#1}%
5348 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5349 \newcommand*{\Glsaccessdescplural}[1]{%
5350   \glsdescriptionpluralaccessdisplay
5351   {%
5352     \Glsentrydescplural{#1}%
5353   }%
5354   {#1}%
5355 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5356 \newcommand*{\GLSaccessdescplural}[1]{%
5357   \glsdescriptionpluralaccessdisplay
5358   {%
5359     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5360   }%
5361   {#1}%
5362 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5363 \newcommand*{\glsaccessshort}[1]{%
5364   \glsshortaccessdisplay
5365   {%
5366     \glsentryshort{#1}%
5367   }%
5368   {#1}%
5369 }

```

`\GLSaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5370 \newcommand*{\GLSaccessshort}[1]{%
5371   \glsshortaccessdisplay
5372   {%
5373     \Glsentryshort{#1}%
5374   }%
5375   {#1}%
5376 }

```

`\GLSAccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5377 \newcommand*{\GLSAccessshort}[1]{%
5378   \glsshortaccessdisplay
5379   {%
5380     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5381   }%

```

```

5382     {#1}%
5383 }

```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```

5384 \newcommand*{\lsaccessshortpl}[1]{%
5385     \glshortpluralaccessdisplay
5386     {%
5387         \glentryshortpl{#1}%
5388     }%
5389     {#1}%
5390 }

```

LSaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5391 \newcommand*{\LSaccessshortpl}[1]{%
5392     \glshortpluralaccessdisplay
5393     {%
5394         \GLentryshortpl{#1}%
5395     }%
5396     {#1}%
5397 }

```

LSaccessshortpl Display the shortplural value (no link and no check for existence) converted to upper case.

```

5398 \newcommand*{\LSaccessshortpl}[1]{%
5399     \glshortpluralaccessdisplay
5400     {%
5401         \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5402     }%
5403     {#1}%
5404 }

```

\glsaccesslong Display the long form (no link and no check for existence).

```

5405 \newcommand*{\glsaccesslong}[1]{%
5406     \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5407 }

```

\Glsaccesslong Display the long form (no link and no check for existence).

```

5408
5409 \newcommand*{\Glsaccesslong}[1]{%
5410     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5411 }

```

\GLSaccesslong Display the long value (no link and no check for existence) converted to upper case.

```

5412 \newcommand*{\GLSaccesslong}[1]{%
5413     \glslongaccessdisplay
5414     {%
5415         \mfirstucMakeUppercase{\glentrylong{#1}}%
5416     }%

```

```

5417     {#1}%
5418 }

glsaccesslongpl Display the long plural form (no link and no check for existence).
5419 \newcommand*{\glsaccesslongpl}[1]{%
5420   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5421 }

Glsaccesslongpl Display the long plural form (no link and no check for existence).
5422
5423 \newcommand*{\Glsaccesslongpl}[1]{%
5424   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5425 }

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.
5426 \newcommand*{\GLSaccesslongpl}[1]{%
5427   \glslongpluralaccessdisplay
5428   {%
5429     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5430   }%
5431   {#1}%
5432 }

Keys for accessibility support.
5433 \define@key{glsxtrabbrv}{access}{%
5434   \def\@gls@nameaccess{#1}%
5435 }
5436 \define@key{glsxtrabbrv}{textaccess}{%
5437   \def\@gls@textaccess{#1}%
5438 }
5439 \define@key{glsxtrabbrv}{firstaccess}{%
5440   \def\@gls@firstaccess{#1}%
5441 }
5442 \define@key{glsxtrabbrv}{shortaccess}{%
5443   \def\@gls@shortaccess{#1}%
5444 }
5445 \define@key{glsxtrabbrv}{shortpluralaccess}{%
5446   \def\@gls@shortaccesspl{#1}%
5447 }

@initaccesskeys
5448 \newcommand*{\@gls@initaccesskeys}{%
5449   \def\@gls@nameaccess{}%
5450   \def\@gls@textaccess{}%
5451   \def\@gls@firstaccess{}%
5452   \def\@gls@shortaccess{}%
5453   \def\@gls@shortaccesspl{}%
5454 }

```


essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```
5455 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5456   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5457   {#2}%
5458   {%
5459     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5460     {#3}%
5461     {%
5462       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5463       {#2}%
5464       {#3}%
5465     }%
5466   }%
5467 }
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5468 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
5469 \ifdefempty\@gls@shortaccess
5470 {%
5471   \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5472   {%
5473     \@glsxtr@insertdots\@gls@shortaccess{#1}%
5474     \eappto\ExtraCustomAbbreviationFields{%
5475       shortaccess={\expandonce\@gls@shortaccess},}%
5476   }%
5477   {}%
5478 }%
5479 {}%
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5480 \ifdefempty\@gls@shortaccess
5481 {}%
5482 {%
5483   \ifdefempty\@gls@shortaccesspl
5484   {%
5485     \@gls@ifaccessattribute@set{aposplural}%
5486     {%
5487       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5488         \@gls@shortaccess'\abrvpluralsuffix}%
5489     }%
5490   }%
5491   \@gls@ifaccessattribute@set{noshortplural}%
5492   {}%
```

```

5493         \let\@gls@shortaccesspl\@gls@shortaccess
5494     }%
5495     {%
5496         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5497             \@gls@shortaccess\abbrvpluralsuffix}%
5498     }%
5499 }%
5500 \eappto\ExtraCustomAbbreviationFields{%
5501     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5502 }%
5503 {}%
5504 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5505 \ifdefempty\@gls@nameaccess
5506 {%
5507     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5508     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5509         \ifdefempty\@gls@shortaccess
5510         {}%
5511         {%
5512             \eappto\ExtraCustomAbbreviationFields{%
5513                 access={\expandonce\@gls@shortaccess},%
5514             }%
5515         }%
5516     }%
5517     {}%
5518 }%
5519 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5520 \ifdefempty\@gls@textaccess
5521 {%
5522     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5523     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5524         \ifdefempty\@gls@shortaccess
5525         {}%
5526         {%
5527             \eappto\ExtraCustomAbbreviationFields{%
5528                 textaccess={\expandonce\@gls@shortaccess},%
5529             }%
5530         }%
5531     }%
5532     {}%
5533 }%
5534 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5535 \ifdefempty\@gls@firstaccess
5536 {%
5537 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5538 {%
```

Do nothing if the shortaccess key hasn't been set.

```
5539 \ifdefempty\@gls@shortaccess
5540 {}%
5541 {%
5542 \eappto\ExtraCustomAbbreviationFields{%
5543 firstaccess={\expandonce\@gls@shortaccess},%
5544 }%
5545 }%
5546 }%
5547 {}%
5548 }%
5549 {}%
5550 }
```

End of if accsupp part

```
5551 }
5552 {
```

No accessibility support. Just define these commands to do \glsentry<xxx>

```
\glsaccessname Display the name value (no link and no check for existence).
5553 \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.
5554 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
5555 \newcommand*{\GLSaccessname}[1]{%
5556 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}}

\glsaccesstext Display the text value (no link and no check for existence).
5557 \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.
5558 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
5559 \newcommand*{\GLSaccesstext}[1]{%
5560 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}}

glsaccessplural Display the plural value (no link and no check for existence).
5561 \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.

5562 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.

5563 `\newcommand*{\GLSaccessplural}[1]{%`
5564 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).

5565 `\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.

5566 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

5567 `\newcommand*{\GLSaccessfirst}[1]{%`
5568 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).

5569 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

5570 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

5571 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5572 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

5573 `\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.

5574 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

5575 `\newcommand*{\GLSaccesssymbol}[1]{%`
5576 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

5577 `\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

5578 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
5579 \newcommand*{\GLSaccesssymbolplural}[1]{%  
5580 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
5581 \newcommand*{\glsaccessdesc}[1]{\glentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
5582 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
5583 \newcommand*{\GLSaccessdesc}[1]{%  
5584 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
5585 \newcommand*{\glsaccessdescplural}[1]{\glentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
5586 \newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
5587 \newcommand*{\GLSaccessdescplural}[1]{%  
5588 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
5589 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
5590 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
5591 \newcommand*{\GLSaccessshort}[1]{%  
5592 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
5593 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
5594 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.

```
5595 \newcommand*{\GLSaccessshortpl}[1]{%
5596 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

\glsaccesslong Display the long form (no link and no check for existence).

```
5597 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```
5598 \newcommand*{\Glsaccesslong}[1]{\Glentrylong{#1}}
```

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.

```
5599 \newcommand*{\GLSaccesslong}[1]{%
5600 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5601 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```
5602 \newcommand*{\Glsaccesslongpl}[1]{\Glentrylongpl{#1}}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```
5603 \newcommand*{\GLSaccesslongpl}[1]{%
5604 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
5605 \newcommand*{@gls@initaccesskeys}{}%
```

lt@short@access This does nothing if there's no accessibility support.

```
5606 \newcommand{\@gls@setup@default@short@access}[1]{%
End of else part
5607 }
```

1.6 Categories

\glscategory Add a new storage key that can be used to indicate a category. The default category is general.

```
5608 \glsaddstoragekey{category}{general}{\glscategory}
```

\glsifcategory Convenient shortcut to determine if an entry has the given category.

```
5609 \newcommand{\glsifcategory}[4]{%
5610 \ifglsfield{#1}{category}{#2}{#3}{#4}%
5611 }
```

Categories can have attributes.

categoryattribute

```
\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
5612 \newcommand*{\glissetcategoryattribute}[3]{%  
5613   \csdef{@glsextr@categoryattr@@#1@#2}{#3}%  
5614 }
```

categoryattribute

```
\glsggetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5615 \newcommand*{\glsggetcategoryattribute}[2]{%  
5616   \csuse{@glsextr@categoryattr@@#1@#2}%  
5617 }
```

categoryattribute

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5618 \newcommand*{\glshascategoryattribute}[4]{%  
5619   \ifcsvoid{@glsextr@categoryattr@@#1@#2}{#4}{#3}%  
5620 }
```

\glissetattribute

```
\glissetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5621 \newcommand*{\glissetattribute}[3]{%  
5622   \glissetcategoryattribute{\glscategory{#1}}{#2}{#3}%  
5623 }
```

\glsggetattribute

```
\glsggetattribute{<entry label>}{<attribute-label>}
```

Short cut where the category label is obtained from the entry information.

```
5624 \newcommand*{\glsggetattribute}[2]{%  
5625   \glsggetcategoryattribute{\glscategory{#1}}{#2}%  
5626 }
```

`\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.

```
5627 \newcommand*{\glshasattribute}[4]{%
5628   \ifglentryexists{#1}%
5629   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5630   {#4}}%
5631 }
```

`categoryattribute` `\glcifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```
5632 \newcommand{\glcifcategoryattribute}[5]{%
5633   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5634   {#5}%
5635   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5636 }
```

`\glcifattribute` `\glcifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```
5637 \newcommand{\glcifattribute}[5]{%
5638   \ifglentryexists{#1}%
5639   {\glcifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5640   {#5}%
5641 }
```

Set attributes for the default general category:

```
5642 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5643 \glsetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
5644 \newcommand*{\glsetregularcategory}[1]{%
5645   \glsetcategoryattribute{#1}{regular}{true}%
5646 }
```


`\regularcategory` `\glsifregularcategory{<category>}{<true part>}{<>false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5647 \newcommand{\glsifregularcategory}[3]{%
5648   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5649 }
```

`\notregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<>false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5650 \newcommand{\glsifnotregularcategory}[3]{%
5651   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5652 }
```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<>false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5653 \newcommand{\glsifregular}[3]{%
5654   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
5655 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<>false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5656 \newcommand{\glsifnotregular}[3]{%
5657   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5658 }
```

`\foreachincategory` `\glsforeachincategory[<glossary labels>]{<category-label>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
5659 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```

5660 \forall glossaries[#1]{#3}%
5661 {%
5662     \forall sentries[#3]{#4}%
5663     {%
5664         \glsifcategory{#4}{#2}{#5}{}%
5665     }%
5666 }%
5667 }

```

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.

```

5668 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5669     \forall glossaries[#1]{#4}%
5670     {%
5671         \forall sentries[#4]{#5}%
5672         {%
5673             \glsifattribute{#5}{#2}{#3}{#6}{}%
5674         }%
5675     }%
5676 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glstrpostdescription`.

```

5677 \ifdef\newterm
5678 {%

```

`\newterm`

```

5679     \renewcommand*{\newterm}[2][ ]{%
5680         \newglossaryentry{#2}%
5681         {type=index,category=index,name={#2},%
5682          description={\glstrpostdescription\nopostdesc},#1}%
5683     }

```

Indexed terms are regular by default.

```

5684     \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5685     \newcommand*{\glstrpostdescindex}{}

5686 }
5687 {}

```

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.

```
5688 \ifdef\printsymbols
5689 {%
```

`glsxtrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```
5690 \newcommand*{\glsxtrnewsymbol}[3] [] {%
5691   \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5692 }
```

Symbols are regular by default.

```
5693 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
5694 \newcommand*{\glsxtrpostdescsymbol}{}

5695 }
5696 {%
```

Similar for the numbers option.

```
5697 \ifdef\printnumbers
5698 {%
```

`glsxtrnewnumber`

```
5699 \ifdef\printnumbers
5700 \newcommand*{\glsxtrnewnumber}[3] [] {%
5701   \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5702 }
```

Numbers are regular by default.

```
5703 \glsssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
5704 \newcommand*{\glsxtrpostdescnumber}{}

5705 }
5706 {%
```

`sxtrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5707 \newcommand*{\glsxtrsetcategory}[2] {%
5708   \@for\@glsxtr@label:=#1\do
5709   {%
5710     \glsfieldxdef{\@glsxtr@label}{category}{#2}%
5711   }%
5712 }
```

`\categoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```

5713 \newcommand*{\glxtrsetcategoryforall}[2]{%
5714   \forallglossaries[#1]{\@glxtr@type}{%
5715     \forallglsentries[\@glxtr@type]{\@glxtr@label}%
5716     {%
5717       \glsfieldxdef{\@glxtr@label}{category}{#2}%
5718     }%
5719   }%
5720 }

```

`\trfieldtitlecase` `\glxtrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```

5721 \newcommand*{\glxtrfieldtitlecase}[2]{%
5722   \expandafter\glxtrfieldtitlecasesecs\expandafter
5723   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
5724 }

```

`\fieldtitlecasesecs` The command used by `\glxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```

5725 \newcommand*{\glxtrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}

```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5726 \@ifpackageloaded{glossaries-accsupp}
5727 {
5728   \renewcommand*{\glossentrydesc}[1]{%
5729     \glsdoifexistsorwarn{#1}%
5730     {%
5731       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossdescfont` attribute to determine the font applied.

```

5732   \glshasattribute{#1}{glossdescfont}%
5733   {%
5734     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5735     \ifcsdef{\@glxtr@attrval}%
5736     {%
5737       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5738     }%
5739     {%
5740       \GlossariesExtraWarning{Unknown control sequence name
5741         '\@glxtr@attrval' supplied in glossdescfont attribute

```

```

5742         for entry '#1'. Ignoring}%
5743         \let\@glxtr@glossdescfont\@firstofone
5744     }%
5745 }%
5746 {\let\@glxtr@glossdescfont\@firstofone}%
5747 \glsifattribute{#1}{glossdesc}{firstuc}%
5748 {%
5749     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5750 }%
5751 {%
5752     \glsifattribute{#1}{glossdesc}{title}%
5753     {%
5754         \@glxtr@do@titlecaps@warn
5755         \glsdescriptionaccessdisplay
5756         {%
5757             \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5758         }%
5759         {#1}%
5760     }%
5761     {%
5762         \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5763     }%
5764 }%
5765 }%
5766 }
5767 }
5768 {
5769 \renewcommand*{\glossentrydesc}[1]{%
5770     \glsdoifexistsorwarn{#1}%
5771     {%
5772         \glssetabbrvfmt{\glscategory{#1}}%
5773         \glsattribute{#1}{glossdescfont}%
5774         {%
5775             \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5776             \ifcsdef{\@glxtr@attrval}%
5777             {%
5778                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5779             }%
5780             {%
5781                 \GlossariesExtraWarning{Unknown control sequence name
5782                 '\@glxtr@attrval' supplied in glossdescfont attribute
5783                 for entry '#1'. Ignoring}%
5784                 \let\@glxtr@glossdescfont\@firstofone
5785             }%
5786         }%
5787         {\let\@glxtr@glossdescfont\@firstofone}%
5788         \glsifattribute{#1}{glossdesc}{firstuc}%
5789         {%
5790             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%

```

```

5791     }%
5792     {%
5793         \glsifattribute{#1}{glossdesc}{title}%
5794         {%
5795             \@glsxtr@do@titlecaps@warn
5796             \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5797         }%
5798         {%
5799             \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
5800         }%
5801     }%
5802 }%
5803 }
5804 }

```

`\glossentryname` If the `glossname` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5805 \ifpackageloaded{glossaries-accsupp}
5806 {
5807   \renewcommand*{\glossentryname}[1]{%
5808     \@glsdoifexistsorwarn{#1}%
5809     {%
5810       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

5811   \glsifattribute{#1}{glossnamefont}%
5812   {%
5813     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5814     \ifcsdef{\@glsxtr@attrval}%
5815     {%
5816       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5817     }%
5818     {%
5819       \GlossariesExtraWarning{Unknown control sequence name
5820         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
5821         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5822       \let\@glsxtr@glossnamefont\glsnamefont
5823     }%
5824   }%
5825   {\let\@glsxtr@glossnamefont\glsnamefont}%
5826   \glsifattribute{#1}{glossname}{firstuc}%
5827   {%
5828     \glsnameaccessdisplay
5829     {%
5830       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5831     }%
5832     {#1}%
5833   }%
5834   {%
5835     \glsifattribute{#1}{glossname}{title}%

```

```

5836      {%
5837      \@glstr@do@titlecaps@warn
5838      \glsnameaccessdisplay
5839      {%
5840      \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
5841      }%
5842      {#1}%
5843      }%
5844      {%
5845      \glsifattribute{#1}{glossname}{uc}%
5846      {%
5847      \glsnameaccessdisplay
5848      {%

```

Hide the label from the upper-casing command.

```

5849      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5850      \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5851      }%
5852      {#1}%
5853      }%
5854      {%
5855      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5856      \glsnameaccessdisplay
5857      {%
5858      \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
5859      }%
5860      {#1}%
5861      }%
5862      }%
5863      }%

```

Do post-name hook:

```

5864      \glstrpostnamehook{#1}%
5865      }%
5866    }
5867  }
5868  {
5869    \renewcommand*{\glossentryname}[1]{%
5870      \@glsdofexistsorwarn{#1}%
5871      {%
5872      \glsetabbrvfmt{\glscategory{#1}}%
5873      \glshasattribute{#1}{glossnamefont}%
5874      {%
5875      \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5876      \ifcsdef{\@glstr@attrval}%
5877      {%
5878      \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5879      }%
5880      {%
5881      \GlossariesExtraWarning{Unknown control sequence name

```

```

5882         ‘\@glstr@attrval’ supplied in glossnamefont attribute
5883         for entry ‘#1’. Reverting to default \string\glstrfont}%
5884         \let\@glstr@glossnamefont\glstrfont
5885     }%
5886 }%
5887 {\let\@glstr@glossnamefont\glstrfont}%
5888 \glstrattribute{#1}{glossname}{firstuc}%
5889 {%
5890     \@glstr@glossnamefont{\Glsentryname{#1}}%
5891 }%
5892 {%
5893     \glstrattribute{#1}{glossname}{title}%
5894     {%
5895         \@glstr@do@titlecaps@warn
5896         \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
5897     }%
5898     {%
5899         \glstrattribute{#1}{glossname}{uc}%
5900     }%

```

Hide the label from the upper-casing command.

```

5901         \letcs{\glo@name}{glo\glstrdetoklabel{#1}@name}%
5902         \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5903     }%
5904     {%

```

This little trick is used by glossaries to allow the user to redefine \glstrfont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```

5905         \letcs{\glo@name}{glo\glstrdetoklabel{#1}@name}%
5906         \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
5907     }%
5908 }%
5909 }%

```

Do post-name hook.

```

5910     \glstrpostnamehook{#1}%
5911 }%
5912 }
5913 }

```

\Glsentryname Redefine to set the abbreviation format and accessibility support.

```

5914 \@ifpackageloaded{glossaries-accsupp}
5915 {
5916     \renewcommand*{\Glsentryname}[1]{%
5917         \@glstrdoifexistsorwarn{#1}%
5918         {%
5919             \glstrsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5920     \glshasattribute{#1}{glossnamefont}%
5921     {%

```



```

5922     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5923     \ifcsdef{\@glxtr@attrval}%
5924     {%
5925         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5926     }%
5927     {%
5928         \GlossariesExtraWarning{Unknown control sequence name
5929             '\@glxtr@attrval' supplied in glossnamefont attribute
5930             for entry '#1'. Reverting to default \string\glsnamefont}%
5931         \let\@glxtr@glossnamefont\glsnamefont
5932     }%
5933 }%
5934 {\let\@glxtr@glossnamefont\glsnamefont}%
5935 \glsnameaccessdisplay
5936 {%
5937     \@glxtr@glossnamefont{\Glsentryname{#1}}%
5938 }%
5939 {#1}%

```

Do post-name hook:

```

5940     \glxtrpostnamehook{#1}%
5941 }%
5942 }
5943 }
5944 {
5945     \renewcommand*{\Glossentryname}[1]{%
5946         \@glsdoifexistsorwarn{#1}%
5947     }%
5948     \glsssetabbrvfmt{\glscategory{#1}}%
5949     \glshasattribute{#1}{glossnamefont}%
5950     {%
5951         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5952         \ifcsdef{\@glxtr@attrval}%
5953         {%
5954             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5955         }%
5956         {%
5957             \GlossariesExtraWarning{Unknown control sequence name
5958                 '\@glxtr@attrval' supplied in glossnamefont attribute
5959                 for entry '#1'. Reverting to default \string\glsnamefont}%
5960             \let\@glxtr@glossnamefont\glsnamefont
5961         }%
5962     }%
5963     {\let\@glxtr@glossnamefont\glsnamefont}%
5964     \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

5965     \glxtrpostnamehook{#1}%
5966 }%
5967 }

```

5968 }

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`\glstrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
5969 \newcommand*{\glstrpostnamehook}[1]{%
5970   \let\@glsnumberformat\@glstr@defaultnumberformat
5971   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
5972   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
5973   \csuse{glstrpostname\glscategory{#1}}%
5974 }
```

`\glsextrapostnamehook`

```
5975 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
5976 \newcommand*{\glsdefpostname}[2]{%
5977   \csdef{glstrpostname#1}{#2}%
5978 }
```

`\glssetaccessdisplay`

```
5979 \@ifpackageloaded{glossaries-accsupp}
5980 {
5981   \newcommand*{\glstr@setaccessdisplay}[1]{%
5982     \ifcsdef{gls#1accessdisplay}%
5983       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
5984       {%
```

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.

```
5985     \edef\@gls@thisval{#1}%
5986     \@for\@gls@map:=\@gls@keymap\do{%
5987       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
5988       \ifdefequal{\@this@key}{\@gls@thisval}%
5989       {%
5990         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
5991         \@endfortrue
5992       }%
5993     }%
5994   }%
5995   \ifcsdef{gls\@gls@thisval accessdisplay}%
5996   {\letcs\@glstr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```

5997      {\let\@glxtr@accessdisplay\@firstoftwo}%
5998    }%
5999  }
6000 }
6001 {%
6002 \newcommand*{\glxtr@setaccessdisplay}[1]{%
6003   \let\@glxtr@accessdisplay\@firstoftwo}
6004 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6005 \newrobustcmd*{\glossentrynameother}[2]{%
6006   \@glsdoifexistsorwarn{#1}%
6007   {%

```

Accessibility support:

```

6008   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6009   \glsssetabbrfmt{\glscategory{#1}}%
6010   \glshasattribute{#1}{glossnamefont}%
6011   {%
6012     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6013     \ifcsdef{\@glxtr@attrval}%
6014     {%
6015       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6016     }%
6017     {%
6018       \GlossariesExtraWarning{Unknown control sequence name
6019         '\@glxtr@attrval' supplied in glossnamefont attribute
6020         for entry '#1'. Reverting to default \string\glsnamefont}%
6021       \let\@glxtr@glossnamefont\glsnamefont
6022     }%
6023   }%
6024   {\let\@glxtr@glossnamefont\glsnamefont}%
6025   \glslifattribute{#1}{glossname}{firstuc}%
6026   {%
6027     \@glxtr@accessdisplay
6028     {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6029     {#1}%
6030   }%
6031   {%
6032     \glslifattribute{#1}{glossname}{title}%
6033     {%
6034       \@glxtr@do@titlecaps@warn
6035       \@glxtr@accessdisplay
6036       {\@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{#2}}}%
6037       {#1}%
6038     }%
6039     {%

```

```

6040      \glsifattribute{#1}{glossname}{uc}%
6041      {%
6042      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6043      \@glsxtr@accessdisplay
6044      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6045      {#1}%
6046      }%
6047      {%
6048      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6049      \@glsxtr@accessdisplay
6050      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6051      {#1}%
6052      }%
6053      }%
6054      }%
      Do post-name hook.
6055      \glsxtrpostnamehook{#1}%
6056      }%
6057 }

```

`format@override` Determines if the `format` key should override the indexing attribute value.

```

6058 \newif\if@glsxtr@format@override
6059 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6060 \@ifpackageloaded{hyperref}
6061 {
      If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
      don't add it.
6062   \ifHy@hyperindex
6063     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6064       \@glsxtr@format@override true
6065       \appto\theindex{\let\glshypernumber\@firstofone}%
6066     }
6067   \else
6068     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6069       \@glsxtr@format@override true
6070       \appto\theindex{\let\glshypernumber\hyperpage}%
6071     }
6072   \fi
6073 }
6074 {
6075   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6076     \@glsxtr@format@override true
6077   }

```

```

6078 }
6079 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6080 \newcommand*{\glstrdoautoindexname}[2]{%
6081   \glshasattribute{#1}{#2}%
6082   {%

```

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.

```

6083   \@glstr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

6084   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6085   \if@glstr@format@override

6086     \ifx\@glsnnumberformat\@glstr@defaultnumberformat
6087     \else
6088       \let\@glstr@attrval\@glsnnumberformat
6089     \fi
6090   \fi
6091   \ifdefstring{\@glstr@attrval}{true}%
6092   {}%
6093   {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6094   \expandafter\glstrautoindex\expandafter{\@glo@name}%
6095   }%
6096   {}%
6097 }

```

glstrautoindex

```

6098 \newcommand*{\glstrautoindex}{\index}

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6099 \newcommand*{\@glstr@autoindex@setname}[1]{%
6100   \protected@edef\@glo@name{\glstrautoindexentry{#1}}%
6101   \glstrautoindexassignsort{\@glo@sort}{#1}%
6102   \@gls@checkmkidxchars\@glo@sort
6103   \@glstr@autoindex@doextra@esc\@glo@sort
6104   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
6105 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6106 \newcommand*{\glstrautoindexentry}[1]{\string\glstryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```

6107 \newcommand*{\glstrautoindexassignsort}[2]{%
6108   \glslentryfield{#1}{#2}{sort}%
6109 }

```

dex@doextra@esc

```
6110 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6111 \ifx\@glsxtr@autoindex@esc\@gls@quotechar
6112 \else
6113 \def\@gls@checkedmkidx{}%
6114 \edef\@glsxtr@checkspch{%
6115 \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
6116 \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
6117 \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6118 \@glsxtr@checkspch
6119 \let#1\@gls@checkedmkidx\relax
6120 \fi
```

Escape actual character unless it has already been escaped.

```
6121 \ifx\@glsxtr@autoindex@at\@gls@actualchar
6122 \else
6123 \def\@gls@checkedmkidx{}%
6124 \edef\@glsxtr@checkspch{%
6125 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
6126 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
6127 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6128 \@glsxtr@checkspch
6129 \let#1\@gls@checkedmkidx\relax
6130 \fi
```

Escape level character unless it has already been escaped.

```
6131 \ifx\@glsxtr@autoindex@level\@gls@levelchar
6132 \else
6133 \def\@gls@checkedmkidx{}%
6134 \edef\@glsxtr@checkspch{%
6135 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
6136 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
6137 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6138 \@glsxtr@checkspch
6139 \let#1\@gls@checkedmkidx\relax
6140 \fi
```

Escape encap character unless it has already been escaped.

```
6141 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6142 \else
6143 \def\@gls@checkedmkidx{}%
6144 \edef\@glsxtr@checkspch{%
6145 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6146 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6147 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6148 \@glsxtr@checkspch
6149 \let#1\@gls@checkedmkidx\relax
6150 \fi
6151 }
```

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`.

```
6152 \newcommand*{\@glxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```
6153 \newcommand*{\GlsXtrSetActualChar}[1]{%
6154   \gdef\@glxtr@autoindex@at{#1}%
6155   \def\@glxtr@autoindex@escat##1##2##3\@glxtr@endescspch{%
6156     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escat}{##1}{##2}{##3}%
6157   }%
6158 }
6159 \@onlypreamble\GlsXtrSetActualChar
6160 \makeatother
6161 \GlsXtrSetActualChar{@}
6162 \makeatletter

```

`\autoindex@encap` Encap character for use with `\index`.

```
6163 \newcommand*{\@glxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```
6164 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6165   \gdef\@glxtr@autoindex@encap{#1}%
6166   \def\@glxtr@autoindex@escencap##1##2##3\@glxtr@endescspch{%
6167     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
6168   }%
6169 }
6170 \GlsXtrSetEncapChar{|}
6171 \@onlypreamble\GlsXtrSetEncapChar

```

`\autoindex@level` Level character for use with `\index`.

```
6172 \newcommand*{\@glxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```
6173 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6174   \gdef\@glxtr@autoindex@level{#1}%
6175   \def\@glxtr@autoindex@esclevel##1##2##3\@glxtr@endescspch{%
6176     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
6177   }%
6178 }
6179 \GlsXtrSetLevelChar{!}
6180 \@onlypreamble\GlsXtrSetLevelChar

```

`\r@autoindex@esc` Escape character for use with `\index`.

```
6181 \newcommand*{\@glxtr@autoindex@esc}{"}

```

`\GlsXtrSetEscChar` Set the escape character.

```
6182 \newcommand*{\GlsXtrSetEscChar}[1]{%
6183   \gdef\@glstr@autoindex@esc{#1}%
6184   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
6185     \@glstr@autoindex@escspch{#1}\@glstr@autoindex@escquote}{##1}{##2}{##3}%
6186   }%
6187 }
6188 \GlsXtrSetEscChar{"}
6189 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character `\actualchar`:

```
6190 \ifdef\actualchar
6191 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6192 {}
```

Quote character `\quotechar`:

```
6193 \ifdef\quotechar
6194 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6195 {}
```

Level character `\levelchar`:

```
6196 \ifdef\levelchar
6197 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6198 {}
```

Encap character `\encapchar`:

```
6199 \ifdef\encapchar
6200 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6201 {}
```

`\letto@endescspch`

```
6202 \def\@glstr@gobbleto@endescspch#1\@glstr@endescspch{}
```

`\toindex@esc@spch`

<code>\@glstr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}</code>
--

```
6203 \newcommand*{\@glstr@autoindex@escspch}[5]{%
6204   \@glstr@tmpb=\expandafter{\@glstr@checkedmkidx}%
6205   \toks@={#3}%
6206   \ifx\@nnil#3\relax
6207     \def\@glstr@checkspch{\@glstr@gobbleto@endescspch#5\@glstr@endescspch}%
6208   \else
6209     \ifx\@nnil#4\relax
6210       \edef\@glstr@checkedmkidx{\the\@glstr@tmpb\the\toks@}%
6211       \def\@glstr@checkspch{\@glstr@gobbleto@endescspch
6212         #4#5\@glstr@endescspch}%
6213     \else
6214       \edef\@glstr@checkedmkidx{\the\@glstr@tmpb\the\toks@
```



```

6215      \@glxtr@autoindex@esc#1}%
6216      \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6217      \fi
6218      \fi
6219      \@glxtr@checkspch
6220 }

```

`\Glossentrydesc` Redefine to set the abbreviation format and accessibility support.

```

6221 \renewcommand*{\Glossentrydesc}[1]{%
6222   \glsoifexistsorwarn{#1}%
6223   {%
6224     \glsetabbrvfmt{\glscategory{#1}}%
6225     \Glsaccessdesc{#1}%
6226   }%
6227 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

6228 \renewcommand*{\Glossentrysymbol}[1]{%
6229   \glsoifexistsorwarn{#1}%
6230   {%
6231     \glsetabbrvfmt{\glscategory{#1}}%
6232     \Glsaccesssymbol{#1}%
6233   }%
6234 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

6235 \renewcommand*{\Glossentrysymbol}[1]{%
6236   \glsoifexistsorwarn{#1}%
6237   {%
6238     \glsetabbrvfmt{\glscategory{#1}}%
6239     \Glsaccesssymbol{#1}%
6240   }%
6241 }

```

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.

```

6242 \newcommand*{\GlsXtrEnableInitialTagging}{%
6243   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6244 }
6245 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\GlsXtrEnableInitialTagging` Starred version undefines command.

```

6246 \newcommand*{\s@glxtr@enabletagging}[2]{%
6247   \undef#2%
6248   \@glxtr@enabletagging{#1}{#2}%
6249 }

```

r@enabletagging Internal command.

```
6250 \newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
6251 \@for\@glsxtr@cat:=#1\do
6252 {%
6253 \ifdefempty\@glsxtr@cat
6254 {}%
6255 {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
6256 }%
6257 \newrobustcmd*#2[1]{##1}%
6258 \def\@glsxtr@taggingcs{#2}%
6259 \renewcommand*\@glsxtr@activate@initialtagging{%
6260 \let#2\@glsxtr@tag
6261 }%
6262 \ifundef\@gls@preglossaryhook
6263 {\GlossariesExtraWarning{Initial tagging requires at least
6264 glossaries.sty v4.19 to work correctly}}%
6265 {}%
6266 }
```

Are we using an old version of mfirstuc that has a bug in \capitalisewords? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
6267 \ifundef\mfu@checkword@do
6268 {
6269 \newcommand*{\mfu@checkword@do}[1]{%
6270 \ifdefstring{\mfu@checkword@arg}{#1}%
6271 {%
6272 \let\@mfu@domakefirstuc\@firstofone
6273 \listbreak
6274 }%
6275 {}%
6276 }
```

\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.

```
6277 \ifundef\mfu@checkword
6278 {
6279 \newcommand{\@glsxtr@do@titlecaps@warn}{%
6280 \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6281 support not available}}%
```

One warning should suffice.

```
6282 \let\@glsxtr@do@titlecaps@warn\relax
6283 }
6284 }
6285 {
6286 \renewcommand*{\mfu@checkword}[1]{%
```

```

6287      \def\mfu@checkword@arg{#1}%
6288      \let\@mfu@domakefirstuc\makefirstuc
6289      \forlistloop\mfu@checkword@do\@mfu@nocaplist
6290    }
6291  }
6292 }
6293 {}% no patch required

@titlecaps@warn  Do warning if title case not supported.
6294 \newcommand*{\@glxtr@do@titlecaps@warn}{}

@initialtagging  Used in \printglossary but at least v4.19 of glossaries required.
6295 \newcommand*{\@glxtr@activate@initialtagging}{}

\@glxtr@tag  Definition of tagging command when used in glossary.
6296 \newrobustcmd*{\@glxtr@tag}[1]{%
6297   \gl@ifattribute{\glscurrententrylabel}{tagging}{true}%
6298   {\glxtrtagfont{#1}}{#1}%
6299 }

\glxtrtagfont  Used in the glossary.
6300 \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.
6301 \ifdef\@gl@preglossaryhook
6302 {
6303   \renewcommand*{\@gl@preglossaryhook}{%
6304     \@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.
6305     \ifundef\@glxtr@org@postdescription
6306     {%
6307       \let\@glxtr@org@postdescription\glspostdescription
6308       \renewcommand*{\glspostdescription}{%
6309         \ifglentryexists{\glscurrententrylabel}%
6310         {%
6311           \glxtrpostdescription
6312           \@glxtr@org@postdescription
6313         }%
6314         {}%
6315       }%
6316     }%
6317   }%

```

Enable the options used by \@glsxtrp:

```
6318 \glossxtrsetpopts
6319 }%
6320 }
6321 {}
```

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.

```
6322 \newcommand*{\glsxtrpostdescription}{%
6323 \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6324 }
```

postdescgeneral

```
6325 \newcommand*{\glsxtrpostdescgeneral}{}%
```

xtrpostdescterm

```
6326 \newcommand*{\glsxtrpostdescterm}{}%
```

postdescacronym

```
6327 \newcommand*{\glsxtrpostdescacronym}{}%
```

descabbreviation

```
6328 \newcommand*{\glsxtrpostdescabbreviation}{}%
```

\glsdefpostdesc Provide a convenient command for defining the post-description hook for the given category.

```
6329 \newcommand*{\glsdefpostdesc}[2]{%
6330 \csdef{glsxtrpostdesc#1}{#2}%
6331 }
```

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.

```
6332 \renewcommand*{\glspostlinkhook}{%
6333 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6334 }
```

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.

```
6335 \newcommand*{\glsxtrpostlinkhook}{%
6336 \glsxtrdiscardperiod{\glslabel}%
6337 {\glsxtrpostlinkendsentence}%
6338 {\glsxtrifcustomdiscardperiod
6339 {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
6340 {\glsxtrpostlink}%
6341 }%
6342 }
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6343 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}
```

`\glxtrpostlink`

```
6344 \newcommand*{\glxtrpostlink}{%
6345   \csuse{\glxtrpostlink\glscategory{\glslabel}}%
6346 }
```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glxtrpostlink`.

```
6347 \newcommand*{\glsdefpostlink}[2]{%
    \ifthenelse{\equal{#1}{}}{%
6348       \PackageError{glossaries-extra}
6349       {Invalid empty category label in \string\glsdefpostlink}{}}%
6350       {\csdef{\glxtrpostlink#1}{#2}}%
6351       {\csdef{\glxtrpostlink#1}{#2}}%
6352 }
```

`linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```
6353 \newcommand*{\glxtrpostlinkendsentence}{%
6354   \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
6355   {%
6356     \csuse{\glxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
6357   .\spacefactor\sffcode'\. \relax
6358   }%
6359   {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the spacefactor.

```
6360   \spacefactor\sffcode'\. \relax
6361   }%
6362 }
```

`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.

```
6363 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6364   \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}{}}%
6365 }
```

`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.

```
6366 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6367   \glxtrifwasfirstuse
6368   {%
```

```

6369 \ifglshassymbol{\glslabel}%
6370 {\space\glstrparen{\glsaccesssymbol{\glslabel}}}%
6371 {}%
6372 }%
6373 {}%
6374 }

```

DescOnFirstUse Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```

6375 \newcommand*{\glstrpostlinkAddSymbolDescOnFirstUse}{%
6376 \glstrifwasfirstuse
6377 {%
6378 \space\glstrparen
6379 {%
6380 \ifglshassymbol{\glslabel}%
6381 {\glsaccesssymbol{\glslabel}, }%
6382 {}%
6383 \glsaccessdesc{\glslabel}%
6384 }%
6385 }%
6386 {}%
6387 }

```

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).

```

6388 \newcommand*{\glstrdiscardperiod}[3]{%
6389 \glstrifwasfirstuse
6390 {%
6391 \glsifattribute{#1}{retainfirstuseperiod}{true}%
6392 {#3}%
6393 {%
6394 \glsifattribute{#1}{discardperiod}{true}%
6395 {%
6396 \glsifplural
6397 {%
6398 \glsifattribute{#1}{pluraldiscardperiod}{true}%
6399 {\glstrifperiod{#2}{#3}}%
6400 {#3}%
6401 }%
6402 {%
6403 \glstrifperiod{#2}{#3}%
6404 }%
6405 }%
6406 {#3}%
6407 }%
6408 }%
6409 {%

```

```

6410 \glsifattribute{#1}{discardperiod}{true}%
6411 {%
6412   \glsifplural
6413   {%
6414     \glsifattribute{#1}{pluraldiscardperiod}{true}%
6415     {\glsxtrifperiod{#2}{#3}}%
6416     {#3}%
6417   }%
6418   {%
6419     \glsxtrifperiod{#2}{#3}%
6420   }%
6421 }%
6422 {#3}%
6423 }%
6424 }

```

`\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`

```

6425 \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 ' ').

```

6426 \newcommand*{\glsxtr@punclist}{.,,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

6427 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

6428 \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`.)

```

6429 \newcommand*{\glsxtrifnextpunc}[2]{%
6430   \def\reserved@a{#1}%
6431   \def\reserved@b{#2}%
6432   \futurelet\@glsxpunc@token\glsxtr@ifnextpunc
6433 }

```

`\glsxtr@ifnextpunc`

```

6434 \newcommand*{\glsxtr@ifnextpunc}{%
6435   \glsxtr@ifpunctoken{\@glsxpunc@token}{\let\reserved@b\reserved@a}{}%
6436   \reserved@b
6437 }

```

xtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
6438 \newcommand*{\glxtr@ifpunctoken}[1]{%
6439   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6440 }
```

xtr@ifpunctoken

```
6441 \def\@glxtr@ifpunctoken#1#2{%
6442   \let\reserved@d=#2%
6443   \ifx\reserved@d\@nnil
6444     \let\glxtr@next\@glxtr@notfoundinlist
6445   \else
6446     \ifx#1\reserved@d
6447       \let\glxtr@next\@glxtr@foundinlist
6448     \else
6449       \let\glxtr@next\@glxtr@ifpunctoken
6450     \fi
6451   \fi
6452   \glxtr@next#1%
6453 }
```

xtr@foundinlist

```
6454 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6455 \def\@glxtr@notfoundinlist#1{\@secondoftwo}
```

glxtrdopostpunc

`\glxtrdopostpunc{<code>}`

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```
6456 \newcommand{\glxtrdopostpunc}[1]{%
6457   \glxtr@ifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6458 }
```

@glxtr@swaptwo

```
6459 \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.


```

6460 \define@key{glsxtrabbrv}{category}{%
6461 \edef\glscategorylabel{#1}%
6462 \ifcsdef{@glsabbrv@current@#1}%
6463 {%

```

Warning should already have been issued.

```

6464 \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6465 \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6466 \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
6467 \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
6468 }%
6469 }%
6470 }

```

Save the short plural form. This may be needed before the entry is defined.

```

6471 \define@key{glsxtrabbrv}{shortplural}{%
6472 \def\@gls@shortpl{#1}%
6473 }

```

Similarly for the long plural form.

```

6474 \define@key{glsxtrabbrv}{longplural}{%
6475 \def\@gls@longpl{#1}%
6476 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```

6477 \newtoks\glsshortpltok

```

\glslongpltok

```

6478 \newtoks\glslongpltok

```

glsxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to \newabbreviation. Note that explicitly using the short or shortplural keys will override this.

```

6479 \newcommand*{\@glsxtr@insertdots}[2]{%
6480 \def#1{%
6481 \@glsxtr@insert@dotes#1#2\@nnil
6482 }

```

glsxtr@insert@dotes

```

6483 \newcommand*{\@glsxtr@insert@dotes}[2]{%
6484 \ifx\@nnil#2\relax
6485 \let\@glsxtr@insert@dotes@next\@gobble
6486 \else
6487 \ifx\relax#2\relax

```

```

6488 \else
6489 \appto#1{#2.}%
6490 \fi
6491 \let\@glxstr@insert@dots@next\@glxstr@insert@dots
6492 \fi
6493 \@glxstr@insert@dots@next#1%
6494 }

```

Similarly provide a way of replacing spaces with `\glxstrwordsep`, which first needs to be defined:

`\glxstrwordsep`

```

6495 \newcommand*{\glxstrwordsep}{\space}

```

Each word is marked with

`\glxstrword`

```

6496 \newcommand*{\glxstrword}[1]{#1}

```

`tr@markwordseps`

```

6497 \newcommand*{\@glxstr@markwordseps}[2]{%
6498 \def#1{}%
6499 \@glxstr@mark@wordseps#1#2 \@nnil
6500 }

```

`r@mark@wordseps`

```

6501 \def\@glxstr@mark@wordseps#1#2 #3{%
6502 \ifdefempty{#1}%
6503 {\def#1{\protect\glxstrword{#2}}}%
6504 {\appto#1{\protect\glxstrwordsep\protect\glxstrword{#2}}}%
6505 \ifx\@nnil#3\relax
6506 \let\@glxstr@mark@wordseps@next\relax
6507 \else
6508 \def\@glxstr@mark@wordseps@next{%
6509 \@glxstr@mark@wordseps#1#3}%
6510 \fi
6511 \@glxstr@mark@wordseps@next
6512 }

```

`newabbreviation` Define a new generic abbreviation.

```

6513 \newcommand*{\newabbreviation}[4][ ]{%
6514 \glxstr@newabbreviation{#1}{#2}{#3}{#4}%
6515 }

```

`newabbreviation` Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This is just makes it easier to save and restore the original definition.)

```

6516 \newcommand*{\glxstr@newabbreviation}[4]{%
6517 \gliskeylisttok{#1}%
6518 \glslabeltok{#2}%

```

6519 \glsshorttok{#3}%

6520 \glslongtok{#4}%

Save the original short and long values (before attribute settings modify them).

6521 \def\glxtrorgshort{#3}%

6522 \def\glxtrorglong{#4}%

Provide extra settings for hooks (if modified, this command must end with a comma).

6523 \def\ExtraCustomAbbreviationFields{}%

Initialise accessibility settings if required.

6524 \@gls@initaccesskeys

Get the category.

6525 \def\glscategorylabel{abbreviation}%

6526 \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%

Ignore the shortplural and longplural keys.

6527 \setkeys*{glxtrabbrv}[shortplural,longplural]{#1}%

Set the default long plural

6528 \def\@gls@longpl{#4\glspluralsuffix}%

6529 \let\@gls@default@longpl\@gls@longpl

Has the markwords attribute been set?

6530 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%

6531 {%

6532 \@glxtr@markwordseps\@gls@long{#4}%

6533 \expandafter\def\expandafter\@gls@longpl\expandafter

6534 {\@gls@long\glspluralsuffix}%

6535 \let\@gls@default@longpl\@gls@longpl

Update \glslongtok.

6536 \expandafter\glslongtok\expandafter{\@gls@long}%

6537 }%

6538 {}%

Has the markshortwords attribute been set? (Not compatible with insertdots.)

6539 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%

6540 {%

6541 \@glxtr@markwordseps\@gls@short{#3}%

6542 }%

6543 {}%

Has the insertdots attribute been set?

6544 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%

6545 {%

6546 \@glxtr@insertdots\@gls@short{#3}%

6547 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%

6548 }%

6549 {\def\@gls@short{#3}}%

6550 }%

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6551 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6552 {%
6553   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6554     '\abbrvpluralsuffix}%
6555 }%
6556 {%
```

Has the noshortplural attribute been set?

```
6557 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6558 {%
6559   \let\@gls@shortpl\@gls@short
6560 }%
6561 {%
6562   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6563     \abbrvpluralsuffix}%
6564 }%
6565 }%
```

Update \glsshorttok:

```
6566 \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
6567 \glxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
6568 \setkeys*{glxtrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
6569 \ifx\@gls@default@longpl\@gls@longpl
6570 \else
```

Has the markwords attribute been set?

```
6571 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6572 {%
6573   \expandafter\@glxtr@keywordseps\expandafter\@gls@longpl\expandafter
6574     {\@gls@longpl}%
6575 }%
6576 {}%
6577 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
6578 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6579 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6580 \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6581 \newabbreviationhook
```

Define this entry:

```
6582 \protected@edef\@do@newglossaryentry{%
6583   \noexpand\newglossaryentry{\the\glslabeltok}%
6584   {%
6585     type=\glstrabbrvtype,%
6586     category=abbreviation,%
6587     short={\the\glsshorttok},%
6588     shortplural={\the\glsshortpltok},%
6589     long={\the\glslongtok},%
6590     longplural={\the\glslongpltok},%
6591     name={\the\glsshorttok},%
6592     \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6593     \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6594     \the\glskeylisttok
6595   }%
6596 }%
6597 \@do@newglossaryentry
6598 \GlsXtrPostNewAbbreviation
6599 }
```

`\evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```
6600 \newcommand*{\glstrnewabbspresetkeyhook}[3]{}%
```

`\NewAbbreviation` Hook used by abbreviation styles.

```
6601 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

`\bbreviationhook` Hook for use with `\newabbreviation`.

```
6602 \newcommand*{\newabbreviationhook}{}%
```

`\reviationFields`

```
6603 \newcommand*{\CustomAbbreviationFields}{}%
```

`\glstrparen` For the parenthetical styles.

```
6604 \newcommand*{\glstrparen}[1]{(#1)}
```

`\lsxtrfullformat` Full format without case change.

```
6605 \newcommand*{\glxtrfullformat}[2]{%
6606   \glsfirstlongfont{\glsaccesslong{#1}}#2\glxtrfullsep{#1}%
6607   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6608 }
```

`\lsxtrfullformat` Full format with case change.

```
6609 \newcommand*{\Glsxtrfullformat}[2]{%
6610   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glxtrfullsep{#1}%
```

```

6611 \glxstrparen{\protect\glsfirstabbrvfont{\glaccessshort{#1}}}%
6612 }

```

`\glxstrfullplformat` Plural full format without case change.

```

6613 \newcommand*{\glxstrfullplformat}[2]{%
6614 \glsfirstlongfont{\glaccesslongpl{#1}}#2\glxstrfullsep{#1}%
6615 \glxstrparen{\protect\glsfirstabbrvfont{\glaccessshortpl{#1}}}%
6616 }

```

`\glxstrfullformat` Plural full format with case change.

```

6617 \newcommand*{\Glsxtrfullplformat}[2]{%
6618 \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxstrfullsep{#1}%
6619 \glxstrparen{\protect\glsfirstabbrvfont{\glaccessshortpl{#1}}}%
6620 }

```

`\glxstrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```

6621 \newcommand*{\glxstrfullsep}[1]{\space}

```

In-line formats in case first use isn't compatible with `\gl Sentryfull` (for example, first use suppresses the long form or uses a footnote).

`\glxstrinlinefullformat` Full format without case change.

```

6622 \newcommand*{\glxstrinlinefullformat}{\glxstrfullformat}

```

`\Glsxtrinlinefullformat` Full format with case change.

```

6623 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}

```

`\glxstrfullplformat` Plural full format without case change.

```

6624 \newcommand*{\glxstrinlinefullplformat}{\glxstrfullplformat}

```

`\Glsxtrinlinefullplformat` Plural full format with case change.

```

6625 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}

```

Redefine `\gl Sentryfull` 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 `\glxstrfull` set of commands instead.

`\gl Sentryfull`

```

6626 \renewcommand*{\gl Sentryfull}[1]{\glxstrinlinefullformat{#1}{}}

```

`\Glsentryfull`

```

6627 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}

```

`\gl Sentryfullpl`

```

6628 \renewcommand*{\gl Sentryfullpl}[1]{\glxstrinlinefullplformat{#1}{}}

```

`\Glsentryfullpl`

```

6629 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}

```

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
6630 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
6631 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.
6632 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`
6633 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
6634 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.
6635 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.
6636 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`
6637 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.
6638 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glsbrvpluralsuffix` Default plural suffix.
6639 `\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).
6640 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\@ns@glsxtrfull}`
6641 `\newcommand*\@ns@glsxtrfull[2][\%]`
6642 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{\@glsxtr@full{#1}{#2}[]}%`
6643 `\@glsxtr@full{#1}{#2}[]}%`
6644 `}`

`\@glsxtr@full` Low-level macro:
6645 `\def\@glsxtr@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).
6646 `\@glsxtr@record{#1}{#2}{\glslink}%`
6647 `\glsdoifexists{#2}%`
6648 `{%`
6649 `\glssetabbrvfmt{\glscategory{#2}}%`
6650 `\let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper`
6651 `\let\glsifplural\@secondoftwo`

```

6652 \let\glscapscase\@firstofthree
6653 \let\glsinsert\@empty
6654 \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%

```

What should \glsxtrifwasfirstuse 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.

```

6655 \glsxtrsetupfulldefs
6656 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6657 }%
6658 \glspostlinkhook
6659 }

```

trsetupfulldefs

```

6660 \newcommand*{\glsxtrsetupfulldefs}{%
6661 \let\glsxtrifwasfirstuse\@firstoftwo
6662 }

```

\Glsxtrfull Full form (first letter uppercase).

```

6663 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
6664 \newcommand*\ns@Glsxtrfull[2][{}]{%
6665 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
6666 {\@Glsxtr@full{#1}{#2}[]}%
6667 }

```

\@Glsxtr@full Low-level macro:

```

6668 \def\@Glsxtr@full#1#2[#3]{%
6669 \glsdoifexists{#2}%
6670 {%
6671 \glssetabbrvfmt{\glscategory{#2}}%
6672 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6673 \let\glsifplural\@secondoftwo
6674 \let\glscapscase\@secondofthree
6675 \let\glsinsert\@empty
6676 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6677 \glsxtrsetupfulldefs
6678 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6679 }%
6680 \glspostlinkhook
6681 }

```

\GLSxtrfull Full form (all uppercase).

```

6682 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
6683 \newcommand*\ns@GLSxtrfull[2][{}]{%
6684 \new@ifnextchar[\@GLSxtr@full{#1}{#2}}%
6685 {\@GLSxtr@full{#1}{#2}[]}%
6686 }

```


\@GLSxtr@full Low-level macro:

```
6687 \def\@GLSxtr@full#1#2[#3]{%
6688   \glsdoifexists{#2}%
6689   {%
6690     \glsetabbrvfmt{\glscategory{#2}}%
6691     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6692     \let\glsifplural\@secondoftwo
6693     \let\glscapscase\@thirdofthree
6694     \let\glsinsert\@empty
6695     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6696     \glsxtrsetupfulldefs
6697     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6698   }%
6699   \glspostlinkhook
6700 }
```

\glsxtrfullpl Plural full form (no case-change).

```
6701 \newrobustcmd*{\glsxtrfullpl}{\@gl@hyp@opt\ns@glsxtrfullpl}
6702 \newcommand*\ns@glsxtrfullpl[2][ ]{%
6703   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
6704     {\@glsxtr@fullpl{#1}{#2}[ ]}%
6705 }
```

\@glsxtr@fullpl Low-level macro:

```
6706 \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).
6707   \@glsxtr@record{#1}{#2}{glslink}%
6708   \glsdoifexists{#2}%
6709   {%
6710     \glsetabbrvfmt{\glscategory{#2}}%
6711     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6712     \let\glsifplural\@firstoftwo
6713     \let\glscapscase\@firstofthree
6714     \let\glsinsert\@empty
6715     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6716     \glsxtrsetupfulldefs
6717     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6718   }%
6719   \glspostlinkhook
6720 }
```

\Glsxtrfullpl Plural full form (first letter uppercase).

```
6721 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\ns@Glsxtrfullpl}
6722 \newcommand*\ns@Glsxtrfullpl[2][ ]{%
6723   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6724     {\@Glsxtr@fullpl{#1}{#2}[ ]}%
6725 }
```

\@Glsxtr@fullpl Low-level macro:

```
6726 \def\@Glsxtr@fullpl#1#2[#3]{%
```

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

```
6727 \@glxtr@record{#1}{#2}{glslink}%
6728 \glstoifexists{#2}%
6729 {%
6730 \glsssetabbrvfmt{\glscategory{#2}}%
6731 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6732 \let\gl@sifplural\@firstoftwo
6733 \let\glscapscase\@secondofthree
6734 \let\gl$insert\@empty
6735 \def\glscustomtext{\@Glsxtrinlinefullplformat{#2}{#3}}%
6736 \glxtrsetupfulldefs
6737 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6738 }%
6739 \glspostlinkhook
6740 }
```

\GLSxtrfullpl Plural full form (all upper case).

```
6741 \newrobustcmd*{\GLSxtrfullpl}{\@gl@hyp@opt\@ns@GLSxtrfullpl}
6742 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
6743 \new@ifnextchar[\@GLSxtr@fullpl{#1}{#2}}%
6744 {\@GLSxtr@fullpl{#1}{#2}[]}%
6745 }
```

\@GLSxtr@fullpl Low-level macro:

```
6746 \def\@GLSxtr@fullpl#1#2[#3]{%
```

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

```
6747 \@glxtr@record{#1}{#2}{glslink}%
6748 \glstoifexists{#2}%
6749 {%
6750 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6751 \let\gl@sifplural\@firstoftwo
6752 \let\glscapscase\@thirdofthree
6753 \let\gl$insert\@empty
6754 \def\glscustomtext{%
6755 \mfirstucMakeUppercase{\@Glsxtrinlinefullplformat{#2}{#3}}}%
6756 \glxtrsetupfulldefs
6757 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6758 }%
6759 \glspostlinkhook
6760 }
```

The short and long forms work in a similar way to acronyms.

`\glxtrshort`

```
6761 \newrobustcmd*{\glxtrshort}{\@gls@hyp@opt\ns@glxtrshort}
```

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

```
6762 \newcommand*{\ns@glxtrshort}[2] [] {%
```

```
6763   \new@ifnextchar[{\@glxtrshort{#1}{#2}}{\@glxtrshort{#1}{#2} []}%
```

```
6764 }
```

Read in the final optional argument:

```
6765 \def\@glxtrshort#1#2[#3] {%
```

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

```
6766   \@glxtr@record{#1}{#2}{glslink}%
```

```
6767   \glsdoifexists{#2}%
```

```
6768   {%
```

Need to make sure `\glsabbrvfont` is set correctly.

```
6769   \glssetabrvfmt{\glscategory{#2}}%
```

```
6770   \let\do@glslink@checkfirsthyper\@glslink@nocheckfirsthyper
```

```
6771   \let\glxtrifwasfirstuse\@secondoftwo
```

```
6772   \let\glslplural\@secondoftwo
```

```
6773   \let\glscapscase\@firstofthree
```

```
6774   \let\glinsert\@empty
```

```
6775   \def\glscustomtext{%
```

```
6776     \glsabbrvfont{\glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
```

```
6777     \ifglxtrininsertinside\else#3\fi
```

```
6778   }%
```

```
6779   \@glslink[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
```

```
6780   }%
```

```
6781   \glspostlinkhook
```

```
6782 }
```

`\Glsxtrshort`

```
6783 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
```

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

```
6784 \newcommand*{\ns@Glsxtrshort}[2] [] {%
```

```
6785   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
```

```
6786 }
```

Read in the final optional argument:

```
6787 \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).

```
6788   \@glxtr@record{#1}{#2}{glslink}%
```

```
6789   \glsdoifexists{#2}%
```

```
6790   {%
```

```
6791   \glssetabrvfmt{\glscategory{#2}}%
```

```
6792   \let\do@glslink@checkfirsthyper\@glslink@nocheckfirsthyper
```

```

6793 \let\glxtrifwasfirstuse\@secondoftwo
6794 \let\glxifplural\@secondoftwo
6795 \let\glscapscase\@secondofthree
6796 \let\glxinsert\@empty
6797 \def\glscustomtext{%
6798 \glxabbrvfont{\glxaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6799 \ifglxtrinsertinside\else#3\fi
6800 }%
6801 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6802 }%
6803 \glspostlinkhook
6804 }

```

\GLSxtrshort

```

6805 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
6806 \newcommand*{\ns@GLSxtrshort}[2] [] {%
6807 \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2} []}%
6808 }

```

Read in the final optional argument:

```

6809 \def\@GLSxtrshort#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6810 \@glsxtr@record{#1}{#2}{glxlink}%
6811 \glxdoifexists{#2}%
6812 {%
6813 \glxsetabbrvfmt{\glxcategory{#2}}%
6814 \let\do@glx@link@checkfirsthyper\@glx@link@nocheckfirsthyper
6815 \let\glxtrifwasfirstuse\@secondoftwo
6816 \let\glxifplural\@secondoftwo
6817 \let\glscapscase\@thirdofthree
6818 \let\glxinsert\@empty
6819 \def\glscustomtext{%
6820 \mfirstucMakeUppercase
6821 {\glxabbrvfont{\glxaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6822 \ifglxtrinsertinside\else#3\fi
6823 }%
6824 }%
6825 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6826 }%
6827 \glspostlinkhook
6828 }

```

\glxtrlong

```

6829 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument

```

```

6830 \newcommand*{\ns@glxtrlong}[2] [] {%
6831   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
6832 }

```

Read in the final optional argument:

```

6833 \def\@glxtrlong#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).

```

6834   \@glxtr@record{#1}{#2}{glslink}%
6835   \glsoifexists{#2}%
6836   {%
6837     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6838     \let\glxtrifwasfirstuse\@secondoftwo
6839     \let\glslifplural\@secondoftwo
6840     \let\glscapscase\@firstofthree
6841     \let\glsininsert\@empty
6842     \def\glscustomtext{%
6843       \glslongfont{\glssaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6844       \ifglxtrininsertinside\else#3\fi
6845     }%
6846     \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6847   }%
6848   \glspostlinkhook
6849 }

```

\Glsxtrlong

```

6850 \newrobustcmd*{\Glsxtrlong}{\@glshyp@opt\ns@Glsxtrlong}

```

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

```

6851 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6852   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6853 }

```

Read in the final optional argument:

```

6854 \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).

```

6855   \@glxtr@record{#1}{#2}{glslink}%
6856   \glsoifexists{#2}%
6857   {%
6858     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6859     \let\glxtrifwasfirstuse\@secondoftwo
6860     \let\glslifplural\@secondoftwo
6861     \let\glscapscase\@secondofthree
6862     \let\glsininsert\@empty
6863     \def\glscustomtext{%
6864       \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6865       \ifglxtrininsertinside\else#3\fi
6866     }%

```

```

6867 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6868 }%
6869 \glspostlinkhook
6870 }

```

\GLSxtrlong

```

6871 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
6872 \newcommand*{\ns@GLSxtrlong}[2] [] {%
6873 \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
6874 }

    Read in the final optional argument:
6875 \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).
6876 \@glsxtr@record{#1}{#2}{glslink}%
6877 \glsdoifexists{#2}%
6878 {%
6879 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6880 \let\glsxtrifwasfirstuse\@secondoftwo
6881 \let\glsifplural\@secondoftwo
6882 \let\gls caps case\@thirdofthree
6883 \let\glsinsert\@empty
6884 \def\gls custom text{%
6885 \mfirstucMakeUppercase
6886 {\gls long font{\gls access long{#2}\ifglsxtrinsertinside#3\fi}%
6887 \ifglsxtrinsertinside\else#3\fi
6888 }%
6889 }%
6890 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6891 }%
6892 \glspostlinkhook
6893 }

```

Plural short forms:

\glsxtrshortpl

```

6894 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6895 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
6896 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
6897 }

    Read in the final optional argument:
6898 \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).

```

6899 \@glstr@record{#1}{#2}{glslink}%
6900 \glsoifexists{#2}%
6901 {%
6902 \glsetabbrvfmt{\glscategory{#2}}%
6903 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6904 \let\glstrifwasfirstuse\@secondoftwo
6905 \let\gl@ifplural\@firstoftwo
6906 \let\glscapscase\@firstofthree
6907 \let\glinsert\@empty
6908 \def\glscustomtext{%
6909 \glabbrvfont{\glaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6910 \ifglstrinsertinside\else#3\fi
6911 }%
6912 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6913 }%
6914 \glspostlinkhook
6915 }

```

\Glsxtrshortpl

```

6916 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\ns@Glsxtrshortpl}

```

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

```

6917 \newcommand*{\ns@Glsxtrshortpl}[2][{}]{%
6918 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
6919 }

```

Read in the final optional argument:

```

6920 \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).

```

6921 \@glstr@record{#1}{#2}{glslink}%
6922 \glsoifexists{#2}%
6923 {%
6924 \glsetabbrvfmt{\glscategory{#2}}%
6925 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6926 \let\glstrifwasfirstuse\@secondoftwo
6927 \let\gl@ifplural\@firstoftwo
6928 \let\glscapscase\@secondofthree
6929 \let\glinsert\@empty
6930 \def\glscustomtext{%
6931 \glabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6932 \ifglstrinsertinside\else#3\fi
6933 }%
6934 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6935 }%
6936 \glspostlinkhook
6937 }

```

\GLSxtrshortpl

6938 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\@ns@GLSxtrshortpl}

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

6939 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
6940 \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6941 }

Read in the final optional argument:

6942 \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).

6943 \@glsxtr@record{#1}{#2}{glslink}%
6944 \glsdoifexists{#2}%
6945 {%
6946 \glssetabbrvfmt{\glscategory{#2}}%
6947 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6948 \let\glsxtrifwasfirstuse\@secondoftwo
6949 \let\glsifplural\@firstoftwo
6950 \let\glsupcase\@thirdofthree
6951 \let\glsinsert\@empty
6952 \def\glscustomtext{%
6953 \mfirstucMakeUppercase
6954 {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
6955 \ifglsxtrininsertinside\else#3\fi
6956 }%
6957 }%
6958 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6959 }%
6960 \glspostlinkhook
6961 }

Plural long forms:

\glsxtrlongpl

6962 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\@ns@glsxtrlongpl}

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

6963 \newcommand*{\ns@glsxtrlongpl}[2] [] {%
6964 \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%
6965 }

Read in the final optional argument:

6966 \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).

6967 \@glsxtr@record{#1}{#2}{glslink}%
6968 \glsdoifexists{#2}%
6969 {%


```

6970 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6971 \let\glxtrifwasfirstuse\@secondoftwo
6972 \let\gl@sifplural\@firstoftwo
6973 \let\glscapscase\@firstofthree
6974 \let\gl$insert\@empty
6975 \def\glscustomtext{%
6976 \glslongfont{\gl@saccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
6977 \ifglxtrininsertinside\else#3\fi
6978 }%
6979 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6980 }%
6981 \glspostlinkhook
6982 }

```

\Glsxtrlongpl

```

6983 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6984 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
6985 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
6986 }
    Read in the final optional argument:
6987 \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).
6988 \glxtr@record{#1}{#2}{glslink}%
6989 \glsdoidexists{#2}%
6990 {%
6991 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6992 \let\glxtrifwasfirstuse\@secondoftwo
6993 \let\gl@sifplural\@firstoftwo
6994 \let\glscapscase\@secondofthree
6995 \let\gl$insert\@empty
6996 \def\glscustomtext{%
6997 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
6998 \ifglxtrininsertinside\else#3\fi
6999 }%
7000 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7001 }%
7002 \glspostlinkhook
7003 }

```

\GLSxtrlongpl

```

7004 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7005 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7006 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7007 }

```

Read in the final optional argument:

```
7008 \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).

```
7009 \@glstr@record{#1}{#2}{glslink}%
7010 \glsoifexists{#2}%
7011 {%
7012 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7013 \let\glstrifwasfirstuse\@secondoftwo
7014 \let\gl@ifplural\@firstoftwo
7015 \let\glscapscase\@thirdofthree
7016 \let\glinsert\@empty
7017 \def\glscustomtext{%
7018 \mfirstucMakeUppercase
7019 {\glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7020 \ifglstrinsertinside\else#3\fi
7021 }%
7022 }%
7023 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7024 }%
7025 \glspostlinkhook
7026 }
```

\glsetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```
7027 \newcommand*{\glsetabbrvfmt}[1]{%
7028 \ifcsdef{@glsabbrv@current@#1}%
7029 {\glstr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7030 {\glstr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7031 }
```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```
7032 \newrobustcmd*{\gluseabbrvfont}[2]{\glsetabbrvfmt{#2}\glsabbrvfont{#1}}
```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```
7033 \newrobustcmd*{\glsuselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}
```

sxtrgenabbrvfmt Similar to \glsgenacfmt, but for abbreviations.

```
7034 \newcommand*{\glsxtrgenabbrvfmt}{%
7035 \ifdefempty\glscustomtext
7036 {%
7037 \ifglused\glslabel
7038 {%
```

Subsequent use:

```
7039 \gl@ifplural
7040 {%
```

Subsequent plural form:

```
7041      \glscapscase
7042      {%
```

Subsequent plural form, don't adjust case:

```
7043      \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7044      }%
7045      {%
```

Subsequent plural form, make first letter upper case:

```
7046      \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7047      }%
7048      {%
```

Subsequent plural form, all caps:

```
7049      \mfirstucMakeUppercase
7050      {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7051      }%
7052      }%
7053      {%
```

Subsequent singular form

```
7054      \glscapscase
7055      {%
```

Subsequent singular form, don't adjust case:

```
7056      \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7057      }%
7058      {%
```

Subsequent singular form, make first letter upper case:

```
7059      \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7060      }%
7061      {%
```

Subsequent singular form, all caps:

```
7062      \mfirstucMakeUppercase
7063      {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7064      }%
7065      }%
7066      }%
7067      {%
```

First use:

```
7068      \glsifplural
7069      {%
```

First use plural form:

```
7070      \glscapscase
7071      {%
```

First use plural form, don't adjust case:

```
7072      \glxtrfullplformat{\glslabel}{\glsinsert}%
```

7073 }%
 7074 {%

First use plural form, make first letter upper case:

7075 \Glsxtrfullplformat{\glslabel}{\glsinsert}%
 7076 }%
 7077 {%

First use plural form, all caps:

7078 \mfirstucMakeUppercase
 7079 {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
 7080 }%
 7081 }%
 7082 {%

First use singular form

7083 \glscapscase
 7084 {%

First use singular form, don't adjust case:

7085 \glsxtrfullformat{\glslabel}{\glsinsert}%
 7086 }%
 7087 {%

First use singular form, make first letter upper case:

7088 \Glsxtrfullformat{\glslabel}{\glsinsert}%
 7089 }%
 7090 {%

First use singular form, all caps:

7091 \mfirstucMakeUppercase
 7092 {\glsxtrfullformat{\glslabel}{\glsinsert}}%
 7093 }%
 7094 }%
 7095 }%
 7096 }%
 7097 {%

User supplied text.

7098 \glscustomtext
 7099 }%
 7100 }

trsubsequentfmt Subsequent use format (singular no case change).

7101 \newcommand*{\glsxtrsubsequentfmt}[2]{%
 7102 \glsabbrvfont{\glsaccessshort{#1}}\ifglsxtrininsertinside #2\fi}%
 7103 \ifglsxtrininsertinside \else#2\fi
 7104 }
 7105 \let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt

subsequentplfmt Subsequent use format (plural no case change).

7106 \newcommand*{\glsxtrsubsequentplfmt}[2]{%

```

7107 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7108 \ifglxtrinsertinside \else#2\fi
7109 }
7110 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```

7111 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7112 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7113 \ifglxtrinsertinside \else#2\fi
7114 }
7115 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```

7116 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7117 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7118 \ifglxtrinsertinside \else#2\fi
7119 }
7120 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

breviationstyle

```

7121 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7122 \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
7123 {%
7124 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7125 }%
7126 {%

```

Have abbreviations already been defined for this category?

```

7127 \ifcsstring{@glsabbrv@current@#1}{#2}%
7128 {%

```

Style already set.

```

7129 }%
7130 {%
7131 \def\@glxtr@dostylewarn{%
7132 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7133 {%
7134 \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7135 style has been switched \MessageBreak
7136 for category ‘#1’, \MessageBreak
7137 but there have already been entries \MessageBreak
7138 defined for this category. Unwanted \MessageBreak
7139 side-effects may result}}%
7140 \@endfortrue
7141 }%
7142 \@glxtr@dostylewarn

```

Set up the style for the given category.

```

7143 \csdef{@glsabbrv@current@#1}{#2}%
7144 \glsxtr@applyabbrvstyle{#2}%
7145 }%
7146 }%
7147 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7148 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7149 \csuse{@glsabbrv@dispstyle@setup@#1}%
7150 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7151 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

7152 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7153 \csuse{@glsabbrv@dispstyle@fmts@#1}%
7154 }

```

`\newabbreviationstyle` This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```

7155 \newcommand*{\newabbreviationstyle}[3]{%
7156 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7157 {%
7158 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7159 defined}{}}%
7160 }%
7161 {%
7162 \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7163 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7164 #2}%
7165 \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7166 \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7167 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7168 \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7169 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7170 \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7171 \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7172 \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7173 \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7174 #3}%
7175 }%
7176 }

```

breivationstyle

```

7177 \newcommand*{\renewabbreviationstyle}[3]{%
7178   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7179   {%
7180     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7181   }%
7182   {%
7183     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
7184       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7185       #2}%
7186     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
7187       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7188       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7189       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7190       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7191       #3}%
7192   }%
7193 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

7194 \newcommand*{\letabbreviationstyle}[2]{%
7195   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7196   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7197 }
```

ecated@abbrstyle

```
\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

7198 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7199   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7200     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7201     \csuse{@glsabbrv@dispstyle@setup@#2}%
7202   }%
7203   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7204 }
```

ecatedAbbrStyle Generate warning for deprecated style use.

```

7205 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7206   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7207   use ‘#2’ instead}%
7208 }
```

eAbbrStyleSetup

```

7209 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7210   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7211   {%
7212     \PackageError{glossaries-extra}%
7213     {Unknown abbreviation style definitions ‘#1’}{}%
7214   }%
7215   {%
7216     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7217   }%
7218 }
```

seAbbrStyleFmts

```

7219 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7220   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7221   {%
7222     \PackageError{glossaries-extra}%
7223     {Unknown abbreviation style formats ‘#1’}{}%
7224   }%
7225   {%
7226     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7227   }%
7228 }
```

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.

```

7229 \newif\ifglsxtrinsertinside
7230 \glsxtrinsertinsidefalse
```

trlongshortname

```

7231 \newcommand*{\glsxtrlongshortname}{%
7232   \protect\glsabbrvfont{\the\glsshorttok}%
7233 }
```

long-short

```

7234 \newabbreviationstyle{long-short}%
7235 {%
```



```

7236 \renewcommand*{\CustomAbbreviationFields}{%
7237     name={\glxtrlongshortname},
7238     sort={\the\glsshorttok},
7239     first={\protect\glsfirstlongfont{\the\glslongtok}%
7240         \protect\glxtrfullsep{\the\glslabeltok}%
7241         \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7242     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7243         \protect\glxtrfullsep{\the\glslabeltok}%
7244         \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7245     plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
7246     description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7247 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7248     \glshasattribute{\the\glslabeltok}{regular}%
7249     {%
7250         \glssetattribute{\the\glslabeltok}{regular}{false}%
7251     }%
7252     {}%
7253 }%
7254 }%
7255 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7256 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7257 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7258 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7259 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7260 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7261 \renewcommand*{\glxtrfullformat}[2]{%
7262     \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7263     \ifglxtrininsertinside\else##2\fi
7264     \glxtrfullsep{##1}%
7265     \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7266 }%
7267 \renewcommand*{\glxtrfullplformat}[2]{%
7268     \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7269     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7270     \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7271 }%
7272 \renewcommand*{\Glsxtrfullformat}[2]{%
7273     \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7274     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7275     \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7276 }%
7277 \renewcommand*{\Glsxtrfullplformat}[2]{%
7278     \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7279     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7280 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7281 }%
7282 }

```

Set this as the default style for general abbreviations:

```

7283 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7284 \newcommand*{\glsxtrlongshortdescsort}{%
7285 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7286 }

```

ngshortdescname

```

7287 \newcommand*{\glsxtrlongshortdescname}{%
7288 \protect\glslongfont{\the\glslongtok}
7289 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7290 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7291 \newabbreviationstyle{long-short-desc}%
7292 {%
7293 \renewcommand*{\CustomAbbreviationFields}{%
7294 name={\glsxtrlongshortdescname},
7295 sort={\glsxtrlongshortdescsort},%
7296 first={\protect\glsfirstlongfont{\the\glslongtok}%
7297 \protect\glsxtrfullsep{\the\glslabeltok}%
7298 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7299 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7300 \protect\glsxtrfullsep{\the\glslabeltok}%
7301 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7302 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7303 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7304 }%

```

Unset the regular attribute if it has been set.

```

7305 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7306 \glshasattribute{\the\glslabeltok}{regular}%
7307 {%
7308 \glissetattribute{\the\glslabeltok}{regular}{false}%
7309 }%
7310 {}%
7311 }%
7312 }%
7313 {%
7314 \GlsXtrUseAbbrStyleFmts{long-short}%
7315 }

```

trshortlongname

```
7316 \newcommand*{\glxtrshortlongname}{%
7317   \protect\glsabbrvfont{\the\glsshorttok}%
7318 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7319 \newabbreviationstyle{short-long}%
7320 {%
7321   \renewcommand*{\CustomAbbreviationFields}{%
7322     name={\glxtrshortlongname},
7323     sort={\the\glsshorttok},
7324     description={\the\glslongtok},%
7325     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7326       \protect\glxtrfullsep{\the\glslabeltok}%
7327       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7328     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7329       \protect\glxtrfullsep{\the\glslabeltok}%
7330       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7331     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7332 }
```

Unset the regular attribute if it has been set.

```
7332 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7333   \glsattribute{\the\glslabeltok}{regular}%
7334   {%
7335     \glssattribute{\the\glslabeltok}{regular}{false}%
7336   }%
7337   {}%
7338 }%
7339 }%
7340 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7341 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7342 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7343 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7344 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7345 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7346 \renewcommand*{\glxtrfullformat}[2]{%
7347   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7348   \ifglxtrininsertinside\else##2\fi
7349   \glxtrfullsep{##1}%
7350   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7351 }%
7352 \renewcommand*{\glxtrfullplformat}[2]{%
7353   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7354   \ifglxtrininsertinside\else##2\fi
7355   \glxtrfullsep{##1}%
7356 }
```

```

7356 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7357 }%
7358 \renewcommand*{\Glsxtrfullformat}[2]{%
7359 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
7360 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7361 \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7362 }%
7363 \renewcommand*{\Glsxtrfullplformat}[2]{%
7364 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
7365 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7366 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7367 }%
7368 }

```

ortlongdescsort

```

7369 \newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7370 \newcommand*{\glsxtrshortlongdescname}{%
7371 \protect\glsabbrvfont{\the\glsshorttok}
7372 \glsxtrparen{\protect\glslongfont{\the\glslongtok}}}%
7373 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7374 \newabbreviationstyle{short-long-desc}%
7375 {%
7376 \renewcommand*{\CustomAbbreviationFields}{%
7377 name={\glsxtrshortlongdescname},
7378 sort={\glsxtrshortlongdescsort},
7379 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7380 \protect\glsxtrfullsep{\the\glslabeltok}%
7381 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7382 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7383 \protect\glsxtrfullsep{\the\glslabeltok}%
7384 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7385 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7386 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7387 }%

```

Unset the regular attribute if it has been set.

```

7388 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7389 \glshasattribute{\the\glslabeltok}{regular}%
7390 {%
7391 \glissetattribute{\the\glslabeltok}{regular}{false}%
7392 }%
7393 {}%
7394 }%

```

```

7395 }%
7396 {%
7397   \GlsXtrUseAbbrStyleFmts{short-long}%
7398 }

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7399 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7400 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`\glstrabbrvfootnote` `\glstrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

7401 \newcommand*{\glstrabbrvfootnote}[2]{\footnote{#2}}

```

`\glstrfootnotename`

```

7402 \newcommand*{\glstrfootnotename}{%
7403   \protect\glsabbrvfont{\the\glsshorttok}%
7404 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

7405 \newabbreviationstyle{footnote}%
7406 {%
7407   \renewcommand*{\CustomAbbreviationFields}{%
7408     name={\glstrfootnotename},
7409     sort={\the\glsshorttok},
7410     description={\the\glslongtok},%

7411     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7412       \protect\glstrabbrvfootnote{\the\glslabeltok}%
7413       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7414     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7415       \protect\glstrabbrvfootnote{\the\glslabeltok}%
7416       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

7417     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.

```

7418 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7419   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7420   \glsattribute{\the\glslabeltok}{regular}%

```

```

7421   {%
7422     \glsetattribute{\the\glslabeltok}{regular}{false}%
7423   }%
7424   {}%
7425 }%
7426 }%
7427 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7428 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7429 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7430 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7431 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7432 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7433 \renewcommand*\glsxtrfullformat[2]{%
7434   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7435   \ifglxtrininsertinside\else##2\fi
7436   \protect\glsxtrabbrvfootnote{##1}%
7437   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7438 }%
7439 \renewcommand*\glsxtrfullplformat[2]{%
7440   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7441   \ifglxtrininsertinside\else##2\fi
7442   \protect\glsxtrabbrvfootnote{##1}%
7443   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7444 }%
7445 \renewcommand*\Glsxtrfullformat[2]{%
7446   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7447   \ifglxtrininsertinside\else##2\fi
7448   \protect\glsxtrabbrvfootnote{##1}%
7449   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7450 }%
7451 \renewcommand*\Glsxtrfullplformat[2]{%
7452   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7453   \ifglxtrininsertinside\else##2\fi
7454   \protect\glsxtrabbrvfootnote{##1}%
7455   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7456 }%

```

The first use full form and the inline full form use the short (long) style.

```

7457 \renewcommand*\glsxtrinlinefullformat[2]{%
7458   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7459   \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7460   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7461 }%
7462 \renewcommand*\glsxtrinlinefullplformat[2]{%
7463   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7464   \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7465   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

7466 }%
7467 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7468   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7469   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7470   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7471 }%
7472 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7473   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7474   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7475   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7476 }%
7477 }

```

short-footnote

```

7478 \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.

```

7479 \newabbreviationstyle{postfootnote}%
7480 {%
7481   \renewcommand*{\CustomAbbreviationFields}{%
7482     name={\glxtrfootnotename},
7483     sort={\the\glsshorttok},
7484     description={\the\glslongtok},%
7485     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7486     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7487     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.

```

7488 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7489   \csdef{glxtrpostlink\glscategorylabel}{%
7490     \glxtrifwasfirstuse
7491     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7492       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7493       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
7494     }%
7495   }%
7496 }%
7497 \glshasattribute{\the\glslabeltok}{regular}%
7498 {%
7499   \glssetattribute{\the\glslabeltok}{regular}{false}%
7500 }%
7501 {}%
7502 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7503 \renewcommand*{\glxtrsetupfulldefs}{%
7504   \let\glxtrifwasfirstuse\@secondoftwo
7505 }%
7506 }%
7507 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7508 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7509 \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvdefaultfont{##1}}%
7510 \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvdefaultfont{##1}}%
7511 \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlongfootnotefont{##1}}%
7512 \renewcommand*{\glxtrlongfont}[1]{\glxtrlongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7513 \renewcommand*{\glxtrfullformat}[2]{%
7514   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7515   \ifglxtrininsertinside\else##2\fi
7516 }%
7517 \renewcommand*{\glxtrfullplformat}[2]{%
7518   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7519   \ifglxtrininsertinside\else##2\fi
7520 }%
7521 \renewcommand*{\Glsxtrfullformat}[2]{%
7522   \glxtrfirstabbrvfont{\Glsxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7523   \ifglxtrininsertinside\else##2\fi
7524 }%
7525 \renewcommand*{\Glsxtrfullplformat}[2]{%
7526   \glxtrfirstabbrvfont{\Glsxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7527   \ifglxtrininsertinside\else##2\fi
7528 }%
```

The first use full form and the inline full form use the short (long) style.

```
7529 \renewcommand*{\glxtrininlinefullformat}[2]{%
7530   \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7531   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7532   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslong{##1}}}%
7533 }%
7534 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7535   \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7536   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7537   \glxtrparen{\glxtrfirstlongfootnotefont{\glxtraccesslongpl{##1}}}%
7538 }%
7539 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
7540   \glxtrfirstabbrvfont{\Glsxtraccessshort{##1}\ifglxtrininsertinside##2\fi}%
7541   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7542   \glxtrparen{\glxtrfirstlongfootnotefont{\Glsxtraccesslong{##1}}}%
7543 }%
7544 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```



```

7545 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7546 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7547 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7548 }%
7549 }

```

rt-postfootnote

```

7550 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

shortnolongname

```

7551 \newcommand*{\glxtrshortnolongname}{%
7552 \protect\glsabbrvfont{\the\glsshorttok}%
7553 }

```

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.

```

7554 \newabbreviationstyle{short}%
7555 {%
7556 \renewcommand*{\CustomAbbreviationFields}{%
7557 name={\glxtrshortnolongname},
7558 sort={\the\glsshorttok},
7559 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7560 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7561 text={\protect\glsabbrvfont{\the\glsshorttok}},
7562 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7563 description={\the\glslongtok}}%
7564 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7565 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
7566 }%
7567 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7568 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7569 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7570 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7571 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7572 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7573 \renewcommand*{\glxtrinlinefullformat}[2]{%
7574 \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
7575 \ifglxtrinsertinside##2\fi}%
7576 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7577 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7578 }%
7579 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7580 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%

```

```

7581 \ifglxtrinsertinside##2\fi}%
7582 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7583 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7584 }%
7585 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7586 \protect\glsfirstabbrvfont{\glssaccessshort{##1}%
7587 \ifglxtrinsertinside##2\fi}%
7588 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7589 \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7590 }%
7591 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7592 \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}%
7593 \ifglxtrinsertinside##2\fi}%
7594 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7595 \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7596 }%

```

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.

```

7597 \renewcommand*{\glxtrfullformat}[2]{%
7598 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7599 \ifglxtrinsertinside\else##2\fi
7600 }%
7601 \renewcommand*{\glxtrfullplformat}[2]{%
7602 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7603 \ifglxtrinsertinside\else##2\fi
7604 }%
7605 \renewcommand*{\Glsxtrfullformat}[2]{%
7606 \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7607 \ifglxtrinsertinside\else##2\fi
7608 }%
7609 \renewcommand*{\Glsxtrfullplformat}[2]{%
7610 \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7611 \ifglxtrinsertinside\else##2\fi
7612 }%
7613 }

```

Set this as the default style for acronyms:

```

7614 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7615 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7616 \newabbreviationstyle{short-nolong-noreg}%
7617 {%
7618 \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7619 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

7620 \glshasattribute{\the\glslabeltok}{regular}%
7621 {%
7622 \glissetattribute{\the\glslabeltok}{regular}{false}%
7623 }%
7624 {}%
7625 }%
7626 }%
7627 {%
7628 \GlsXtrUseAbbrStyleFmts{short-nolong}%
7629 }

```

trshortdescname

```

7630 \newcommand*{\glxtrshortdescname}{%
7631 \protect\glsabbrvfont{\the\glsshorttok}%
7632 }

```

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.

```

7633 \newabbreviationstyle{short-desc}%
7634 {%
7635 \renewcommand*{\CustomAbbreviationFields}{%
7636 name={\glxtrshortdescname},
7637 sort={\the\glsshorttok},
7638 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7639 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7640 text={\protect\glsabbrvfont{\the\glsshorttok}},
7641 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7642 description={\the\glslongtok}}%
7643 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7644 \glissetattribute{\the\glslabeltok}{regular}{true}}%
7645 }%
7646 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7647 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7648 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7649 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7650 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7651 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7652 \renewcommand*{\glxtrinlinefullformat}[2]{%
7653 \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
7654 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7655 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7656 }%
7657 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7658 \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
7659 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7660 \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%

```

```

7661 }%
7662 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7663   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7664   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7665   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7666 }%
7667 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7668   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7669   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7670   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7671 }%

```

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.

```

7672 \renewcommand*{\glxtrfullformat}[2]{%
7673   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7674   \ifglxtrinsertinside\else##2\fi
7675 }%
7676 \renewcommand*{\glxtrfullplformat}[2]{%
7677   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7678   \ifglxtrinsertinside\else##2\fi
7679 }%
7680 \renewcommand*{\Glsxtrfullformat}[2]{%
7681   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7682   \ifglxtrinsertinside\else##2\fi
7683 }%
7684 \renewcommand*{\Glsxtrfullplformat}[2]{%
7685   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7686   \ifglxtrinsertinside\else##2\fi
7687 }%
7688 }

```

ort-nolong-desc

```
7689 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7690 \newabbreviationstyle{short-nolong-desc-noreg}%
7691 {%
7692   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7693 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7694   \glshasattribute{\the\glslabeltok}{regular}%
7695   {%
7696     \glissetattribute{\the\glslabeltok}{regular}{false}%
7697   }%
7698   {}%
7699 }%
7700 }%
7701 {%

```

```

7702 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7703 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7704 \newabbreviationstyle{nolong-short}%
7705 {%
7706 \GlsXtrUseAbbrStyleSetup{short-nolong}%
7707 }%
7708 {%
7709 \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7710 \renewcommand*{\glxtrinlinefullformat}[2]{%
7711 \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7712 \ifglxtrininsertinside##2\fi}%
7713 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7714 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7715 }%
7716 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7717 \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7718 \ifglxtrininsertinside##2\fi}%
7719 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7720 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7721 }%
7722 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7723 \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7724 \ifglxtrininsertinside##2\fi}%
7725 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7726 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7727 }%
7728 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7729 \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7730 \ifglxtrininsertinside##2\fi}%
7731 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7732 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7733 }%
7734 }

```

ong-short-noreg Like nolong-short but doesn't set the regular attribute.

```

7735 \newabbreviationstyle{nolong-short-noreg}%
7736 {%
7737 \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7738 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7739 \glshasattribute{\the\glslabeltok}{regular}%
7740 {%
7741 \glssetattribute{\the\glslabeltok}{regular}{false}%
7742 }%

```

```

7743     {}%
7744   }%
7745 }%
7746 {%
7747   \GlsXtrUseAbbrStyleFmts{nolong-short}%
7748 }

```

noshortdescname

```

7749 \newcommand*{\glxtrlongnoshortdescname}{%
7750   \protect\glslongfont{\the\glslongtok}%
7751 }

```

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.

```

7752 \newabbreviationstyle{long-desc}%
7753 {%
7754   \renewcommand*{\CustomAbbreviationFields}{%
7755     name={\glxtrlongnoshortdescname},
7756     sort={\the\glslongtok},
7757     first={\protect\glslongfont{\the\glslongtok}},
7758     firstplural={\protect\glslongfont{\the\glslongpltok}},
7759     text={\glslongfont{\the\glslongtok}},
7760     plural={\glslongfont{\the\glslongpltok}}}%
7761   }%
7762   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7763     \glsssetattribute{\the\glslabeltok}{regular}{true}}%
7764 }%
7765 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7766 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7767 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
7768 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
7769 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
7770 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7771 \renewcommand*{\glxtrsubsequentfmt}[2]{%
7772   \glslongfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7773   \ifglxtrinsertinside \else##2\fi
7774 }%
7775 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7776   \glslongfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7777   \ifglxtrinsertinside \else##2\fi
7778 }%
7779 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7780   \glslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7781   \ifglxtrinsertinside \else##2\fi
7782 }%

```

```

7783 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7784   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
7785   \ifglsxtrinsertinside \else##2\fi
7786 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7787 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7788   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7789   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7790   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7791 }%
7792 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7793   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7794   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7795   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7796 }%
7797 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7798   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7799   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7800   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7801 }%
7802 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7803   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7804   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7805   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7806 }%

```

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.

```

7807 \renewcommand*{\glsxtrfullformat}[2]{%
7808   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7809   \ifglsxtrinsertinside\else##2\fi
7810 }%
7811 \renewcommand*{\glsxtrfullplformat}[2]{%
7812   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7813   \ifglsxtrinsertinside\else##2\fi
7814 }%
7815 \renewcommand*{\Glsxtrfullformat}[2]{%
7816   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7817   \ifglsxtrinsertinside\else##2\fi
7818 }%
7819 \renewcommand*{\Glsxtrfullplformat}[2]{%
7820   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7821   \ifglsxtrinsertinside\else##2\fi
7822 }%
7823 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

7824 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```
7825 \newabbreviationstyle{long-noshort-desc-noreg}%
7826 {%
7827   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
7828   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7829     \glshasattribute{\the\glslabeltok}{regular}%
7830     {%
7831       \glissetattribute{\the\glslabeltok}{regular}{false}%
7832     }%
7833   }%
7834 }%
7835 }%
7836 {%
7837   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7838 }
```

longnoshortname

```
7839 \newcommand*{\glsxtrlongnoshortname}{%
7840   \protect\glsabbrvfont{\the\glsshorttok}%
7841 }
```

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.

```
7842 \newabbreviationstyle{long}%
7843 {%
7844   \renewcommand*{\CustomAbbreviationFields}{%
7845     name={\glsxtrlongnoshortname},
7846     sort={\the\glsshorttok},
7847     first={\protect\glsfirstlongfont{\the\glslongtok}},
7848     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7849     text={\glslongfont{\the\glslongtok}},
7850     plural={\glslongfont{\the\glslongpltok}},%
7851     description={\the\glslongtok}%
7852   }%
7853   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7854     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7855 }%
7856 {%
7857   \GlsXtrUseAbbrStyleFmts{long-desc}%
7858 }
```

long-noshort Provide a synonym that matches similar styles.

```
7859 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.


```

7860 \newabbreviationstyle{long-noshort-noreg}%
7861 {%
7862   \GlsXtrUseAbbrStyleSetup{long-noshort}%
       Unset the regular attribute if it has been set.
7863   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7864     \glshasattribute{\the\glslabeltok}{regular}%
7865     {%
7866       \glissetattribute{\the\glslabeltok}{regular}{false}%
7867     }%
7868     {}}%
7869   }%
7870 }%
7871 {%
7872   \GlsXtrUseAbbrStyleFmts{long-noshort}%
7873 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

```

\glsxtrscfont   Maintained for backward-compatibility.
7874 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}

\glsabbrvscfont Added for consistent naming.
7875 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}

sxtrfirstscfont Maintained for backward-compatibility.
7876 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}

irstabbrvscfont Added for consistent naming.
7877 \newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}

```

and for the default short form suffix:

```

\glsxtrscsuffix
7878 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}

long-short-sc
7879 \newabbreviationstyle{long-short-sc}%
7880 {%
7881   \renewcommand*{\CustomAbbreviationFields}{%
7882     name={\glsxtrlongshortname},
7883     sort={\the\glsshorttok},
7884     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7885       \protect\glsxtrfullsep{\the\glslabeltok}%
7886       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7887     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7888       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

7889 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7890 plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
7891 description={\the\glslongtok}}%
7892 \renewcommand*\GlsXtrPostNewAbbreviation}{%
7893 \glsattribute{\the\glslabeltok}{regular}%
7894 {%
7895 \glssetattribute{\the\glslabeltok}{regular}{false}%
7896 }%
7897 {}%
7898 }%
7899 }%
7900 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7901 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
7902 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7903 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

7904 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7905 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7906 \renewcommand*\glxtrfullformat[2]{%
7907 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7908 \ifglxtrininsertinside\else##2\fi
7909 \glxtrfullsep{##1}%
7910 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7911 }%
7912 \renewcommand*\glxtrfullplformat[2]{%
7913 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7914 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7915 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7916 }%
7917 \renewcommand*\Glsxtrfullformat[2]{%
7918 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7919 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7920 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7921 }%
7922 \renewcommand*\Glsxtrfullplformat[2]{%
7923 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7924 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7925 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7926 }%
7927 }

```

g-short-sc-desc

```

7928 \newabbreviationstyle{long-short-sc-desc}%
7929 {%
7930 \renewcommand*\CustomAbbreviationFields{%

```

```

7931   name={\glxtrlongshortdescname},
7932   sort={\glxtrlongshortdescsort},%
7933   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7934     \protect\glxtrfullsep{\the\glslabeltok}%
7935     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7936   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7937     \protect\glxtrfullsep{\the\glslabeltok}%
7938     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7939   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7940   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7941 }%

```

Unset the regular attribute if it has been set.

```

7942 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7943   \glshasattribute{\the\glslabeltok}{regular}%
7944   {%
7945     \glissetattribute{\the\glslabeltok}{regular}{false}%
7946   }%
7947   {}%
7948 }%
7949 }%
7950 {%

```

As long-short-sc style:

```

7951 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7952 }

```

Now the short (long) version

```

7953 \newabbreviationstyle{short-sc-long}%
7954 {%
7955   \renewcommand*{\CustomAbbreviationFields}{%
7956     name={\glxtrshortlongname},
7957     sort={\the\glsshorttok},
7958     description={\the\glslongtok},%
7959     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7960       \protect\glxtrfullsep{\the\glslabeltok}%
7961       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7962     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7963       \protect\glxtrfullsep{\the\glslabeltok}%
7964       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7965     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7966 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7967   \glshasattribute{\the\glslabeltok}{regular}%
7968   {%
7969     \glissetattribute{\the\glslabeltok}{regular}{false}%
7970   }%
7971   {}%
7972 }%
7973 }%

```

7974 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
7975 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7976 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7977 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7978 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7979 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7980 \renewcommand*{\glxtrfullformat}[2]{%
7981   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7982   \ifglxtrininsertinside\else##2\fi
7983   \glxtrfullsep{##1}%
7984   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7985 }%
7986 \renewcommand*{\glxtrfullplformat}[2]{%
7987   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7988   \ifglxtrininsertinside\else##2\fi
7989   \glxtrfullsep{##1}%
7990   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7991 }%
7992 \renewcommand*{\Glsxtrfullformat}[2]{%
7993   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7994   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7995   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7996 }%
7997 \renewcommand*{\Glsxtrfullplformat}[2]{%
7998   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7999   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8000   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8001 }%
8002 }
```

As before but user provides description

```
8003 \newabbreviationstyle{short-sc-long-desc}%
8004 {%
8005   \renewcommand*{\CustomAbbreviationFields}{%
8006     name={\glxtrshortlongdescname},
8007     sort={\glxtrshortlongdescsort},
8008     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
8009     \protect\glxtrfullsep{\the\glslabeltok}%
8010     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8011     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
8012     \protect\glxtrfullsep{\the\glslabeltok}%
8013     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8014     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8015     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8016   }%
```

Unset the regular attribute if it has been set.

```

8017 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8018   \glshasattribute{\the\glslabeltok}{regular}%
8019   {%
8020     \glissetattribute{\the\glslabeltok}{regular}{false}%
8021   }%
8022   {}%
8023 }%
8024 }%
8025 {%

```

As short-sc-long style:

```

8026 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8027 }

```

short-sc

```

8028 \newabbreviationstyle{short-sc}%
8029 {%
8030   \renewcommand*{\CustomAbbreviationFields}{%
8031     name={\glxtrshortnolongname},
8032     sort={\the\glsshorttok},
8033     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8034     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8035     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8036     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8037     description={\the\glslongtok}}%
8038   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8039     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8040 }%
8041 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8042 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8043 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8044 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8045 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8046 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8047 \renewcommand*{\glxtrinlinefullformat}[2]{%
8048   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8049   \ifglxtrininsertinside##2\fi}%
8050   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8051   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8052 }%
8053 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8054   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8055   \ifglxtrininsertinside##2\fi}%
8056   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8057   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8058 }%

```

```

8059 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8060   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
8061     \ifglxtrinsertinside##2\fi}%
8062   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8063   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8064 }%
8065 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8066   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
8067     \ifglxtrinsertinside##2\fi}%
8068   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8069   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8070 }%

```

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.

```

8071 \renewcommand*{\glsxtrfullformat}[2]{%
8072   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8073   \ifglxtrinsertinside\else##2\fi
8074 }%
8075 \renewcommand*{\glsxtrfullplformat}[2]{%
8076   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8077   \ifglxtrinsertinside\else##2\fi
8078 }%
8079 \renewcommand*{\Glsxtrfullformat}[2]{%
8080   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8081   \ifglxtrinsertinside\else##2\fi
8082 }%
8083 \renewcommand*{\Glsxtrfullplformat}[2]{%
8084   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8085   \ifglxtrinsertinside\else##2\fi
8086 }%
8087 }

```

short-sc-nolong

```

8088 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8089 \newabbreviationstyle{short-sc-desc}{%
8090 {%
8091   \renewcommand*{\CustomAbbreviationFields}{%
8092     name={\glsxtrshortdesname},
8093     sort={\the\glsshorttok},
8094     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8095     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8096     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8097     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8098     description={\the\glslongtok}}%
8099 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8100   \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

8101 }%
8102 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8103 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8104 \renewcommand*{\glssabrvfont}[1]{\glssabrvscfont{##1}}%
8105 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvscfont{##1}}%
8106 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8107 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8108 \renewcommand*{\glxtrinlinefullformat}[2]{%
8109   \glsfirstabrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8110   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8111   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8112 }%
8113 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8114   \glsfirstabrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8115   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8116   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8117 }%
8118 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8119   \glsfirstabrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8120   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8121   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8122 }%
8123 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8124   \glsfirstabrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8125   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8126   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8127 }%
```

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.

```
8128 \renewcommand*{\glxtrfullformat}[2]{%
8129   \glsfirstabrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8130   \ifglxtrininsertinside\else##2\fi
8131 }%
8132 \renewcommand*{\glxtrfullplformat}[2]{%
8133   \glsfirstabrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8134   \ifglxtrininsertinside\else##2\fi
8135 }%
8136 \renewcommand*{\Glsxtrfullformat}[2]{%
8137   \glsfirstabrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8138   \ifglxtrininsertinside\else##2\fi
8139 }%
8140 \renewcommand*{\Glsxtrfullplformat}[2]{%
8141   \glsfirstabrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8142   \ifglxtrininsertinside\else##2\fi
8143 }%
8144 }
```

-sc-nolong-desc

```
8145 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
8146 \newabbreviationstyle{nolong-short-sc}%
8147 {%
8148   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8149 }%
8150 {%
8151   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8152 \renewcommand*{\glxtrinlinefullformat}[2]{%
8153   \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
8154     \ifglxtrininsertinside##2\fi}%
8155   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8156   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8157 }%
8158 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8159   \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
8160     \ifglxtrininsertinside##2\fi}%
8161   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8162   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8163 }%
8164 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8165   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8166     \ifglxtrininsertinside##2\fi}%
8167   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8168   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
8169 }%
8170 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8171   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8172     \ifglxtrininsertinside##2\fi}%
8173   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8174   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8175 }%
8176 }
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```
8177 \newabbreviationstyle{long-noshort-sc}%
8178 {%
8179   \renewcommand*{\CustomAbbreviationFields}{%
8180     name={\glxtrlongnoshortname},
8181     sort={\the\glssshorttok},
8182     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8183     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8184     text={\protect\glslongdefaultfont{\the\glslongtok}},
8185     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
```



```

8186     description={\the\glslongtok}%
8187 }%
8188 \renewcommand*\GlsXtrPostNewAbbreviation{%
8189     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8190 }%
8191 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8192 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrscsuffix}%
8193 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8194 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8195 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8196 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8197 \renewcommand*\glsxtrsubsequentfmt}[2]{%
8198     \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8199     \ifglsxtrininsertinside \else##2\fi
8200 }%
8201 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
8202     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8203     \ifglsxtrininsertinside \else##2\fi
8204 }%
8205 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8206     \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8207     \ifglsxtrininsertinside \else##2\fi
8208 }%
8209 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8210     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8211     \ifglsxtrininsertinside \else##2\fi
8212 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8213 \renewcommand*\glsxtrinlinefullformat}[2]{%
8214     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8215     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8216     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8217 }%
8218 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8219     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8220     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8221     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8222 }%
8223 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8224     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8225     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8226     \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8227 }%
8228 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8229     \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8230     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8231 \glxtrparen{\protect\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8232 }%

```

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.

```

8233 \renewcommand*{\glxtrfullformat}[2]{%
8234 \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8235 \ifglxtrininsertinside\else##2\fi
8236 }%
8237 \renewcommand*{\glxtrfullplformat}[2]{%
8238 \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8239 \ifglxtrininsertinside\else##2\fi
8240 }%
8241 \renewcommand*{\Glsxtrfullformat}[2]{%
8242 \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8243 \ifglxtrininsertinside\else##2\fi
8244 }%
8245 \renewcommand*{\Glsxtrfullplformat}[2]{%
8246 \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8247 \ifglxtrininsertinside\else##2\fi
8248 }%
8249 }

```

long-sc Backward compatibility:

```

8250 \@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`.

```

8251 \newabbreviationstyle{long-noshort-sc-desc}%
8252 {%
8253 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8254 }%
8255 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8256 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8257 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8258 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8259 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8260 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8261 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8262 \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8263 \ifglxtrininsertinside \else##2\fi
8264 }%
8265 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8266 \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8267 \ifglxtrininsertinside \else##2\fi
8268 }%

```

```

8269 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8270   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8271   \ifglxtrinsertinside \else##2\fi
8272 }%
8273 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8274   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8275   \ifglxtrinsertinside \else##2\fi
8276 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8277 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8278   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8279   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8280   \Glsxtrparen{\protect\Glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8281 }%
8282 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8283   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8284   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8285   \Glsxtrparen{\protect\Glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8286 }%
8287 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8288   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8289   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8290   \Glsxtrparen{\protect\Glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8291 }%
8292 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8293   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8294   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8295   \Glsxtrparen{\protect\Glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8296 }%

```

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.

```

8297 \renewcommand*{\Glsxtrfullformat}[2]{%
8298   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8299   \ifglxtrinsertinside\else##2\fi
8300 }%
8301 \renewcommand*{\Glsxtrfullplformat}[2]{%
8302   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8303   \ifglxtrinsertinside\else##2\fi
8304 }%
8305 \renewcommand*{\Glsxtrfullformat}[2]{%
8306   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8307   \ifglxtrinsertinside\else##2\fi
8308 }%
8309 \renewcommand*{\Glsxtrfullplformat}[2]{%
8310   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8311   \ifglxtrinsertinside\else##2\fi
8312 }%
8313 }

```

long-desc-sc Backward compatibility:

```
8314 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
8315 \newabbreviationstyle{short-sc-footnote}%
8316 {%
8317   \renewcommand*{\CustomAbbreviationFields}{%
8318     name={\glxtrfootnotename},
8319     sort={\the\glsshorttok},
8320     description={\the\glslongtok},%
8321     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8322       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8323       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8324     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8325       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8326       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8327     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.

```
8328 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8329   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8330   \glshasattribute{\the\glslabeltok}{regular}%
8331   {%
8332     \glsssetattribute{\the\glslabeltok}{regular}{false}%
8333   }%
8334 }%
8335 }%
8336 }%
8337 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8338 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8339 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8340 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8341 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8342 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```
8343 \renewcommand*{\glxtrfullformat}[2]{%
8344   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8345   \ifglxtrininsertinside\else##2\fi
8346   \protect\glxtrabbrvfootnote{##1}%
8347   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8348 }%
8349 \renewcommand*{\glxtrfullplformat}[2]{%
8350   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8351   \ifglxtrininsertinside\else##2\fi
8352   \protect\glxtrabbrvfootnote{##1}%
8353   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

8354 }%
8355 \renewcommand*{\Glsxtrfullformat}[2]{%
8356   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8357   \ifglxtrinsertinside\else##2\fi
8358   \protect\glxtrabbrvfootnote{##1}%
8359   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8360 }%
8361 \renewcommand*{\Glsxtrfullplformat}[2]{%
8362   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8363   \ifglxtrinsertinside\else##2\fi
8364   \protect\glxtrabbrvfootnote{##1}%
8365   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8366 }%

```

The first use full form and the inline full form use the short (long) style.

```

8367 \renewcommand*{\glxtrinlinefullformat}[2]{%
8368   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8369   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8370   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8371 }%
8372 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8373   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8374   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8375   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8376 }%
8377 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8378   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8379   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8380   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8381 }%
8382 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8383   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8384   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8385   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8386 }%
8387 }

```

footnote-sc Backward compatibility:

```

8388 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8389 \newabbreviationstyle{short-sc-postfootnote}%
8390 {%
8391   \renewcommand*{\CustomAbbreviationFields}{%
8392     name={\glxtrfootnotename},
8393     sort={\the\glsshorttok},
8394     description={\the\glslongtok},%
8395     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8396     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8397     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.

```
8398 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8399 \csdef{glxstrpostlink\glscategorylabel}{%
8400 \glxtrifwasfirstuse
8401 {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
8402 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8403 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
8404 }%
8405 {}%
8406 }%
8407 \glshasattribute{\the\glslabeltok}{regular}%
8408 {%
8409 \glissetattribute{\the\glslabeltok}{regular}{false}%
8410 }%
8411 {}%
8412 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8413 \renewcommand*{\glxtrsetupfulldefs}{%
8414 \let\glxtrifwasfirstuse\@secondoftwo
8415 }%
8416 }%
8417 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8418 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8419 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
8420 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8421 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8422 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8423 \renewcommand*{\glxtrfullformat}[2]{%
8424 \glsfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8425 \ifglxtrininsertinside\else##2\fi
8426 }%
8427 \renewcommand*{\glxtrfullplformat}[2]{%
8428 \glsfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8429 \ifglxtrininsertinside\else##2\fi
8430 }%
8431 \renewcommand*{\Glsxtrfullformat}[2]{%
8432 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8433 \ifglxtrininsertinside\else##2\fi
8434 }%
8435 \renewcommand*{\Glsxtrfullplformat}[2]{%
```

```

8436 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8437 \ifglxtrinsertinside\else##2\fi
8438 }%

```

The first use full form and the inline full form use the short (long) style.

```

8439 \renewcommand*{\glxtrinlinefullformat}[2]{%
8440 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8441 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8442 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8443 }%
8444 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8445 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8446 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8447 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8448 }%
8449 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8450 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8451 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8452 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8453 }%
8454 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8455 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8456 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8457 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8458 }%
8459 }

```

postfootnote-sc Backward compatibility:

```

8460 \@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.
8461 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
8462 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8463 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
8464 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

```

and for the default short form suffix:

\glxtrsmsuffix

```
8465 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8466 \newabbreviationstyle{long-short-sm}%
8467 {%
8468   \renewcommand*{\CustomAbbreviationFields}{%
8469     name={\glxtrlongshortname},
8470     sort={\the\glsshorttok},
8471     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8472       \protect\glxtrfullsep{\the\glslabeltok}%
8473       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8474     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8475       \protect\glxtrfullsep{\the\glslabeltok}%
8476       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8477     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8478     description={\the\glslongtok}}%
8479   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8480     \glshasattribute{\the\glslabeltok}{regular}%
8481     {%
8482       \glissetattribute{\the\glslabeltok}{regular}{false}%
8483     }%
8484   }%
8485 }%
8486 }%
8487 {%
8488   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8489   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8490   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8491 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8492 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8493 \renewcommand*{\glxtrfullformat}[2]{%
8494   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8495   \ifglxtrininsertinside\else##2\fi
8496   \glxtrfullsep{##1}%
8497   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8498 }%
8499 \renewcommand*{\glxtrfullplformat}[2]{%
8500   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8501   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8502   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8503 }%
8504 \renewcommand*{\Glsxtrfullformat}[2]{%
8505   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8506   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8507   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%

```



```

8508 }%
8509 \renewcommand*{\Glsxtrfullplformat}[2]{%
8510   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8511   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8512   \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8513 }%
8514 }

```

g-short-sm-desc

```

8515 \newabbreviationstyle{long-short-sm-desc}%
8516 {%
8517   \renewcommand*{\CustomAbbreviationFields}{%
8518     name={\glxtrlongshortdescname},
8519     sort={\glxtrlongshortdescsort},%
8520     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8521       \protect\glxtrfullsep{\the\glslabeltok}%
8522       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8523     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8524       \protect\glxtrfullsep{\the\glslabeltok}%
8525       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8526     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8527     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8528   }%

```

Unset the regular attribute if it has been set.

```

8529 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8530   \glshasattribute{\the\glslabeltok}{regular}%
8531   {%
8532     \glissetattribute{\the\glslabeltok}{regular}{false}%
8533   }%
8534   {}%
8535 }%
8536 }%
8537 {%

```

As long-short-sm style:

```

8538 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8539 }

```

short-sm-long Now the short (long) version

```

8540 \newabbreviationstyle{short-sm-long}%
8541 {%
8542   \renewcommand*{\CustomAbbreviationFields}{%
8543     name={\glxtrshortlongname},
8544     sort={\the\glsshorttok},
8545     description={\the\glslongtok},%
8546     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8547       \protect\glxtrfullsep{\the\glslabeltok}%
8548       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8549     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%

```

```

8550 \protect\glxtrfullsep{\the\glslabeltok}%
8551 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8552 plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8553 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8554 \glshasattribute{\the\glslabeltok}{regular}%
8555 {%
8556 \glissetattribute{\the\glslabeltok}{regular}{false}%
8557 }%
8558 {}%
8559 }%
8560}%
8561{%
8562 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8563 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8564 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8565 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8566 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8567 \renewcommand*{\glxtrfullformat}[2]{%
8568 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8569 \ifglxtrininsertinside\else##2\fi
8570 \glxtrfullsep{##1}%
8571 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8572}%
8573 \renewcommand*{\glxtrfullplformat}[2]{%
8574 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8575 \ifglxtrininsertinside\else##2\fi
8576 \glxtrfullsep{##1}%
8577 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8578}%
8579 \renewcommand*{\Glsxtrfullformat}[2]{%
8580 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8581 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8582 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8583}%
8584 \renewcommand*{\Glsxtrfullplformat}[2]{%
8585 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8586 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8587 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8588}%
8589}

```

rt-sm-long-desc As before but user provides description

```

8590 \newabbreviationstyle{short-sm-long-desc}%
8591 {%
8592 \renewcommand*{\CustomAbbreviationFields}{%
8593 name={\glxtrshortlongdescname},

```

```

8594     sort={\glxtrshortlongdescsort},
8595     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8596       \protect\glxtrfullsep{\the\glslabeltok}%
8597       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8598     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8599       \protect\glxtrfullsep{\the\glslabeltok}%
8600       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8601     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8602     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8603 }%

```

Unset the regular attribute if it has been set.

```

8604 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8605   \glshasattribute{\the\glslabeltok}{regular}%
8606   {%
8607     \glissetattribute{\the\glslabeltok}{regular}{false}%
8608   }%
8609   {}%
8610 }%
8611 }%
8612 {%

```

As short-sm-long style:

```

8613 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8614 }

```

short-sm

```

8615 \newabbreviationstyle{short-sm}%
8616 {%
8617   \renewcommand*{\CustomAbbreviationFields}{%
8618     name={\glxtrshortnolongname},
8619     sort={\the\glsshorttok},
8620     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8621     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8622     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8623     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8624     description={\the\glslongtok}}%
8625   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8626     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8627 }%
8628 {%
8629   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8630   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8631   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8632   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8633   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8634 \renewcommand*{\glxtrinlinefullformat}[2]{%
8635   \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%

```

```

8636     \ifglxtrinsertinside##2\fi}%
8637     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8638     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8639 }%
8640 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8641     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
8642         \ifglxtrinsertinside##2\fi}%
8643     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8644     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8645 }%

8646 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8647     \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8648         \ifglxtrinsertinside##2\fi}%
8649     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8650     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8651 }%
8652 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8653     \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8654         \ifglxtrinsertinside##2\fi}%
8655     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8656     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8657 }%

```

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.

```

8658 \renewcommand*{\glxtrfullformat}[2]{%
8659     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8660     \ifglxtrinsertinside\else##2\fi
8661 }%
8662 \renewcommand*{\glxtrfullplformat}[2]{%
8663     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8664     \ifglxtrinsertinside\else##2\fi
8665 }%
8666 \renewcommand*{\Glsxtrfullformat}[2]{%
8667     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8668     \ifglxtrinsertinside\else##2\fi
8669 }%
8670 \renewcommand*{\Glsxtrfullplformat}[2]{%
8671     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8672     \ifglxtrinsertinside\else##2\fi
8673 }%
8674 }

```

short-sm-nolong

```
8675 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8676 \newabbreviationstyle{short-sm-desc}%

```

```

8677 {%
8678   \renewcommand*{\CustomAbbreviationFields}{%
8679     name={\glxtrshortdescname},
8680     sort={\the\glsshorttok},
8681     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8682     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8683     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8684     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8685     description={\the\glslongtok}}%
8686   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8687     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8688 }%
8689 {%
8690   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8691   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8692   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8693   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8694   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8695   \renewcommand*\glxtrinlinefullformat[2]{%
8696     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8697     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8698     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8699   }%
8700   \renewcommand*\glxtrinlinefullplformat[2]{%
8701     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8702     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8703     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8704   }%
8705   \renewcommand*\Glsxtrinlinefullformat[2]{%
8706     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8707     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8708     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8709   }%
8710   \renewcommand*\Glsxtrinlinefullplformat[2]{%
8711     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8712     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8713     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8714   }%

```

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.

```

8715   \renewcommand*\glxtrfullformat[2]{%
8716     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8717     \ifglxtrininsertinside\else##2\fi
8718   }%
8719   \renewcommand*\glxtrfullplformat[2]{%
8720     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8721     \ifglxtrininsertinside\else##2\fi

```

```

8722 }%
8723 \renewcommand*{\Glsxtrfullformat}[2]{%
8724   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8725   \ifglxtrinsertinside\else##2\fi
8726 }%
8727 \renewcommand*{\Glsxtrfullplformat}[2]{%
8728   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8729   \ifglxtrinsertinside\else##2\fi
8730 }%
8731 }

```

-sm-nolong-desc

```

8732 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8733 \newabbreviationstyle{nolong-short-sm}%
8734 {%
8735   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8736 }%
8737 {%
8738   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8739 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8740   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8741     \ifglxtrinsertinside##2\fi}%
8742   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8743   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8744 }%
8745 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8746   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8747     \ifglxtrinsertinside##2\fi}%
8748   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8749   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8750 }%
8751 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8752   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8753     \ifglxtrinsertinside##2\fi}%
8754   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8755   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8756 }%
8757 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8758   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8759     \ifglxtrinsertinside##2\fi}%
8760   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8761   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8762 }%
8763 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8764 \newabbreviationstyle{long-noshort-sm}%
8765 {%
8766   \renewcommand*{\CustomAbbreviationFields}{%
8767     name={\glsxtrlongnoshortname},
8768     sort={\the\glsshorttok},
8769     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8770     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8771     text={\protect\glslongdefaultfont{\the\glslongtok}},
8772     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8773     description={\the\glslongtok}%
8774   }%
8775   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8776     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8777 }%
8778 {%
8779   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8780   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8781   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
8782   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8783   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8784 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8785   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8786   \ifglsxtrininsertinside \else##2\fi
8787 }%
8788 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8789   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8790   \ifglsxtrininsertinside \else##2\fi
8791 }%
8792 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8793   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8794   \ifglsxtrininsertinside \else##2\fi
8795 }%
8796 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8797   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8798   \ifglsxtrininsertinside \else##2\fi
8799 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8800 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8801   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8802   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8803   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8804 }%
8805 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8806   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8807   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8808 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8809 }%
8810 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8811 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8812 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8813 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8814 }%
8815 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8816 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8817 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8818 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8819 }%

```

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.

```

8820 \renewcommand*{\glsxtrfullformat}[2]{%
8821 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8822 \ifglsxtrininsertinside\else##2\fi
8823 }%
8824 \renewcommand*{\glsxtrfullplformat}[2]{%
8825 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8826 \ifglsxtrininsertinside\else##2\fi
8827 }%
8828 \renewcommand*{\Glsxtrfullformat}[2]{%
8829 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8830 \ifglsxtrininsertinside\else##2\fi
8831 }%
8832 \renewcommand*{\Glsxtrfullplformat}[2]{%
8833 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8834 \ifglsxtrininsertinside\else##2\fi
8835 }%
8836 }

```

long-sm Backward compatibility:

```

8837 \@glsxtr@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`.

```

8838 \newabbreviationstyle{long-noshort-sm-desc}%
8839 {%
8840 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8841 }%
8842 {%
8843 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8844 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8845 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
8846 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8847 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).


```

8848 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8849   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8850   \ifglxtrinsertinside \else##2\fi
8851 }%
8852 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8853   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8854   \ifglxtrinsertinside \else##2\fi
8855 }%
8856 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8857   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8858   \ifglxtrinsertinside \else##2\fi
8859 }%
8860 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8861   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8862   \ifglxtrinsertinside \else##2\fi
8863 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8864 \renewcommand*{\glxtrinlinefullformat}[2]{%
8865   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8866   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8867   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8868 }%
8869 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8870   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8871   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8872   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8873 }%
8874 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8875   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8876   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8877   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8878 }%
8879 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8880   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8881   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8882   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8883 }%

```

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.

```

8884 \renewcommand*{\glxtrfullformat}[2]{%
8885   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8886   \ifglxtrinsertinside\else##2\fi
8887 }%
8888 \renewcommand*{\glxtrfullplformat}[2]{%
8889   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8890   \ifglxtrinsertinside\else##2\fi
8891 }%
8892 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

8893 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8894 \ifglxtrinsertinside\else##2\fi
8895 }%
8896 \renewcommand*{\Glsxtrfullplformat}[2]{%
8897 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8898 \ifglxtrinsertinside\else##2\fi
8899 }%
8900 }

```

long-desc-sm Backward compatibility:

```

8901 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8902 \newabbreviationstyle{short-sm-footnote}%
8903 {%
8904 \renewcommand*{\CustomAbbreviationFields}{%
8905 name={\glxtrfootnotename},
8906 sort={\the\glsshorttok},
8907 description={\the\glslongtok},%
8908 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8909 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8910 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8911 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8912 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8913 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8914 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.

```

8915 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8916 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8917 \glsattribute{\the\glslabeltok}{regular}%
8918 {%
8919 \glssetattribute{\the\glslabeltok}{regular}{false}%
8920 }%
8921 {}%
8922 }%
8923 }%
8924 {%
8925 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8926 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8927 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8928 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8929 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8930 \renewcommand*{\glxtrfullformat}[2]{%
8931 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8932 \ifglxtrinsertinside\else##2\fi
8933 \protect\glxtrabbrvfootnote{##1}%

```

```

8934     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8935 }%
8936 \renewcommand*{\glsxtrfullplformat}[2]{%
8937   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8938   \ifglsxtrininsertinside\else##2\fi
8939   \protect\glsxtrabbrvfootnote{##1}%
8940   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8941 }%
8942 \renewcommand*{\Glsxtrfullformat}[2]{%
8943   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8944   \ifglsxtrininsertinside\else##2\fi
8945   \protect\glsxtrabbrvfootnote{##1}%
8946   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8947 }%
8948 \renewcommand*{\Glsxtrfullplformat}[2]{%
8949   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8950   \ifglsxtrininsertinside\else##2\fi
8951   \protect\glsxtrabbrvfootnote{##1}%
8952   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8953 }%

```

The first use full form and the inline full form use the short (long) style.

```

8954 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8955   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8956   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8957   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8958 }%
8959 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8960   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8961   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8962   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8963 }%
8964 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8965   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8966   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8967   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8968 }%
8969 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8970   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8971   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8972   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8973 }%
8974 }

```

footnote-sm Backward compatibility:

```

8975 \@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

8976 \newabbreviationstyle{short-sm-postfootnote}%
8977 {}

```

```

8978 \renewcommand*{\CustomAbbreviationFields}{%
8979   name={\glxtrfootnotename},
8980   sort={\the\glsshorttok},
8981   description={\the\glslongtok},%
8982   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
8983   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
8984   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.

```

8985 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8986   \csdef{glxtrpostlink\glscategorylabel}{%
8987     \glxtrifwasfirstuse
8988     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8989       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
8990       {\glsfirstlongfootnotefont{\glentrylong{\glslabel}}}}%
8991     }%
8992   }%
8993 }%
8994 \glshasattribute{\the\glslabeltok}{regular}%
8995 {%
8996   \glissetattribute{\the\glslabeltok}{regular}{false}%
8997 }%
8998 {}%
8999 }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

9000 \renewcommand*{\glxtrsetupfulldefs}{%
9001   \let\glxtrifwasfirstuse\@secondoftwo
9002 }%
9003 }%
9004 {%
9005 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9006 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9007 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9008 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9009 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9010 \renewcommand*{\glxtrfullformat}[2]{%
9011   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9012   \ifglxtrininsertinside\else##2\fi
9013 }%
9014 \renewcommand*{\glxtrfullplformat}[2]{%
9015   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9016   \ifglxtrininsertinside\else##2\fi
9017 }%

```

```

9018 \renewcommand*{\Glsxtrfullformat}[2]{%
9019   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9020   \ifglxtrinsertinside\else##2\fi
9021 }%
9022 \renewcommand*{\Glsxtrfullplformat}[2]{%
9023   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9024   \ifglxtrinsertinside\else##2\fi
9025 }%

```

The first use full form and the inline full form use the short (long) style.

```

9026 \renewcommand*{\glxtrinlinefullformat}[2]{%
9027   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9028   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9029   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9030 }%
9031 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9032   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9033   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9034   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9035 }%
9036 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9037   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9038   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9039   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9040 }%
9041 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9042   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9043   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9044   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9045 }%
9046 }

```

postfootnote-sm Backward compatibility:

```

9047 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9048 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

9049 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

9050 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`firstlongemfont` Only used by the “long-em” styles.

```
9051 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
9052 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9053 \newabbreviationstyle{long-short-em}%
9054 {%
9055   \renewcommand*{\CustomAbbreviationFields}{%
9056     name={\glsxtrlongshortname},
9057     sort={\the\glsshorttok},
9058     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
9059       \protect\glsxtrfullsep{\the\glslabeltok}%
9060       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9061     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
9062       \protect\glsxtrfullsep{\the\glslabeltok}%
9063       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9064     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9065     description={\the\glslongtok}}%
9066   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9067     \glshasattribute{\the\glslabeltok}{regular}}%
9068   {%
9069     \glissetattribute{\the\glslabeltok}{regular}{false}%
9070   }%
9071   {}%
9072 }%
9073 }%
9074 {%
9075   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9076   \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvemfont{##1}}%
9077   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%

```

Use the default long fonts.

```
9078 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9079 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9080 \renewcommand*{\glsxtrfullformat}[2]{%
9081   \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9082   \ifglxtrininsertinside\else##2\fi
9083   \glsxtrfullsep{##1}%
9084   \glsxtrparen{\glfirstabbrvemfont{\glaccessshort{##1}}}%
9085 }%
9086 \renewcommand*{\glsxtrfullplformat}[2]{%
9087   \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9088   \ifglxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9089   \glsxtrparen{\glfirstabbrvemfont{\glaccessshortpl{##1}}}%
9090 }%
9091 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9092 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9093 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9094 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9095 }%
9096 \renewcommand*{\Glsxtrfullplformat}[2]{%
9097 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9098 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9099 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9100 }%
9101 }

```

g-short-em-desc

```

9102 \newabbreviationstyle{long-short-em-desc}%
9103 {%
9104 \renewcommand*{\CustomAbbreviationFields}{%
9105 name={\glxtrlongshortdescname},
9106 sort={\glxtrlongshortdescsort},%
9107 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9108 \protect\glxtrfullsep{\the\glslabeltok}%
9109 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9110 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9111 \protect\glxtrfullsep{\the\glslabeltok}%
9112 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9113 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9114 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9115 }%

```

Unset the regular attribute if it has been set.

```

9116 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9117 \glshasattribute{\the\glslabeltok}{regular}%
9118 {%
9119 \glissetattribute{\the\glslabeltok}{regular}{false}%
9120 }%
9121 {}%
9122 }%
9123 }%
9124 {%

```

As long-short-em style:

```

9125 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9126 }

```

ong-em-short-em

```

9127 \newabbreviationstyle{long-em-short-em}%
9128 {%
9129 \glslongemfont is used in the description since \glsdesc doesn't set the style.
9129 \renewcommand*{\CustomAbbreviationFields}{%
9130 name={\glxtrlongshortname},
9131 sort={\the\glsshorttok},

```

```

9132 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9133 \protect\glsxtrfullsep{\the\glslabeltok}%
9134 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9135 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9136 \protect\glsxtrfullsep{\the\glslabeltok}%
9137 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9138 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9139 description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9140 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9141 \glsattribute{\the\glslabeltok}{regular}%
9142 {%
9143 \glssetattribute{\the\glslabeltok}{regular}{false}%
9144 }%
9145 {}%
9146 }%
9147 }%
9148 {%
9149 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9150 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9151 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9152 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9153 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9154 \renewcommand*{\glsxtrfullformat}[2]{%
9155 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9156 \ifglsxtrininsertinside\else##2\fi
9157 \glsxtrfullsep{##1}%
9158 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9159 }%
9160 \renewcommand*{\glsxtrfullplformat}[2]{%
9161 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9162 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9163 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9164 }%
9165 \renewcommand*{\Glsxtrfullformat}[2]{%
9166 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9167 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9168 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9169 }%
9170 \renewcommand*{\Glsxtrfullplformat}[2]{%
9171 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9172 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9173 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9174 }%
9175 }

```


m-short-em-desc

```
9176 \newabbreviationstyle{long-em-short-em-desc}%
9177 {%
9178   \renewcommand*{\CustomAbbreviationFields}{%
9179     name={\glxtrlongshortdescname},
9180     sort={\glxtrlongshortdescsort},%
9181     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9182       \protect\glxtrfullsep{\the\glslabeltok}%
9183       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9184     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9185       \protect\glxtrfullsep{\the\glslabeltok}%
9186       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9187     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9188     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9189   }%
9190   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9191     \glshasattribute{\the\glslabeltok}{regular}%
9192     {%
9193       \glissetattribute{\the\glslabeltok}{regular}{false}%
9194     }%
9195   }%
9196 }%
9197 }%
9198 {%
9199   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9200 }
```

short-em-long Now the short (long) version

```
9201 \newabbreviationstyle{short-em-long}%
9202 {%
9203   \renewcommand*{\CustomAbbreviationFields}{%
9204     name={\glxtrshortlongname},
9205     sort={\the\glsshorttok},
9206     description={\the\glslongtok},%
9207     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9208       \protect\glxtrfullsep{\the\glslabeltok}%
9209       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9210     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9211       \protect\glxtrfullsep{\the\glslabeltok}%
9212       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9213     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9214   }%
9215   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9216     \glshasattribute{\the\glslabeltok}{regular}%
9217     {%
9218       \glissetattribute{\the\glslabeltok}{regular}{false}%
9219     }%
9220   }%
9221 }
```

```

9219   {}%
9220 }%
9221 }%
9222 {%

```

Mostly as short-long style:

```

9223 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9224 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9225 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9226 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9227 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9228 \renewcommand*{\glssxtrfullformat}[2]{%
9229   \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9230   \ifglssxtrininsertinside\else##2\fi
9231   \glssxtrfullsep{##1}%
9232   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9233 }%
9234 \renewcommand*{\glssxtrfullplformat}[2]{%
9235   \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9236   \ifglssxtrininsertinside\else##2\fi
9237   \glssxtrfullsep{##1}%
9238   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9239 }%
9240 \renewcommand*{\Glsxtrfullformat}[2]{%
9241   \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
9242   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9243   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslong{##1}}}%
9244 }%
9245 \renewcommand*{\Glsxtrfullplformat}[2]{%
9246   \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
9247   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9248   \glssxtrparen{\glssfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9249 }%
9250 }

```

rt-em-long-desc As before but user provides description

```

9251 \newabbreviationstyle{short-em-long-desc}%
9252 {%
9253   \renewcommand*{\CustomAbbreviationFields}{%
9254     name={\glssxtrshortlongdescname},
9255     sort={\glssxtrshortlongdescsort},
9256     first={\protect\glssfirstabbrvemfont{\the\glssshorttok}}%
9257     \protect\glssxtrfullsep{\the\glsslabeltok}}%
9258     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongtok}}},%
9259     firstplural={\protect\glssfirstabbrvemfont{\the\glssshortpltok}}%
9260     \protect\glssxtrfullsep{\the\glsslabeltok}}%
9261     \glssxtrparen{\protect\glssfirstlongdefaultfont{\the\glsslongpltok}}},%
9262     text={\protect\glssabbrvemfont{\the\glssshorttok}},%

```

```

9263     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9264 }%

```

Unset the regular attribute if it has been set.

```

9265 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9266   \glshasattribute{\the\glslabeltok}{regular}%
9267   {%
9268     \glssetattribute{\the\glslabeltok}{regular}{false}%
9269   }%
9270   {}%
9271 }%
9272 }%
9273 {%
9274   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9275 }

```

hort-em-long-em

```

9276 \newabbreviationstyle{short-em-long-em}%
9277 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9278 \renewcommand*{\CustomAbbreviationFields}{%
9279   name={\glsxtrshortlongname},
9280   sort={\the\glsshorttok},
9281   description={\protect\glslongemfont{\the\glslongtok}},%
9282   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9283   \protect\glsxtrfullsep{\the\glslabeltok}%
9284   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9285   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9286   \protect\glsxtrfullsep{\the\glslabeltok}%
9287   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9288   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9289 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9290   \glshasattribute{\the\glslabeltok}{regular}%
9291   {%
9292     \glssetattribute{\the\glslabeltok}{regular}{false}%
9293   }%
9294   {}%
9295 }%
9296 }%
9297 {%
9298   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9299   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9300   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9301   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9302   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

9303 \renewcommand*{\glxtrfullformat}[2]{%
9304   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9305   \ifglxtrinsertinside\else##2\fi
9306   \glxtrfullsep{##1}%
9307   \glxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9308 }%
9309 \renewcommand*{\glxtrfullplformat}[2]{%
9310   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9311   \ifglxtrinsertinside\else##2\fi
9312   \glxtrfullsep{##1}%
9313   \glxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9314 }%
9315 \renewcommand*{\Glsxtrfullformat}[2]{%
9316   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9317   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9318   \glxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
9319 }%
9320 \renewcommand*{\Glsxtrfullplformat}[2]{%
9321   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9322   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9323   \glxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
9324 }%
9325 }

```

em-long-em-desc

```

9326 \newabbreviationstyle{short-em-long-em-desc}%
9327 {%
9328   \renewcommand*{\CustomAbbreviationFields}{%
9329     name={\glxtrshortlongdescname},%
9330     sort={\glxtrshortlongdescsort},%
9331     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9332     \protect\glxtrfullsep{\the\glslabeltok}}%
9333     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9334     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9335     \protect\glxtrfullsep{\the\glslabeltok}}%
9336     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9337     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9338     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9339 }%

```

Unset the regular attribute if it has been set.

```

9340 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9341   \glshasattribute{\the\glslabeltok}{regular}%
9342   {%
9343     \glissetattribute{\the\glslabeltok}{regular}{false}%
9344   }%
9345   {}%
9346 }%

```

```

9347 }%
9348 {%
9349   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9350 }

```

short-em

```

9351 \newabbreviationstyle{short-em}%
9352 {%
9353   \renewcommand*{\CustomAbbreviationFields}{%
9354     name={\glstrshortnolongname},
9355     sort={\the\glsshorttok},
9356     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9357     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9358     text={\protect\glabbrvemfont{\the\glsshorttok}},
9359     plural={\protect\glabbrvemfont{\the\glsshortpltok}},
9360     description={\the\glslongtok}}%
9361   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9362     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9363 }%
9364 {%
9365   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9366   \renewcommand*{\glabbrvfnt}[1]{\glabbrvemfont{##1}}%
9367   \renewcommand*{\glfirstabbrvfnt}[1]{\glfirstabbrvemfont{##1}}%
9368   \renewcommand*{\glfirstlongfnt}[1]{\glfirstlongdefaultfont{##1}}%
9369   \renewcommand*{\glslongfnt}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9370   \renewcommand*{\glxtrinlinefullformat}[2]{%
9371     \protect\glfirstabbrvemfont{\glaccessshort{##1}}%
9372     \ifglxtrininsertinside##2\fi}%
9373   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9374   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
9375 }%
9376   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9377     \protect\glfirstabbrvemfont{\glaccessshortpl{##1}}%
9378     \ifglxtrininsertinside##2\fi}%
9379   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9380   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9381 }%
9382   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9383     \protect\glfirstabbrvemfont{\Glsaccessshort{##1}}%
9384     \ifglxtrininsertinside##2\fi}%
9385   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9386   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9387 }%
9388   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9389     \protect\glfirstabbrvemfont{\Glsaccessshortpl{##1}}%
9390     \ifglxtrininsertinside##2\fi}%
9391   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9392 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9393 }%

```

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.

```

9394 \renewcommand*{\glxtrfullformat}[2]{%
9395   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9396   \ifglxtrininsertinside\else##2\fi
9397 }%
9398 \renewcommand*{\glxtrfullplformat}[2]{%
9399   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9400   \ifglxtrininsertinside\else##2\fi
9401 }%
9402 \renewcommand*{\Glsxtrfullformat}[2]{%
9403   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9404   \ifglxtrininsertinside\else##2\fi
9405 }%
9406 \renewcommand*{\Glsxtrfullplformat}[2]{%
9407   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9408   \ifglxtrininsertinside\else##2\fi
9409 }%
9410 }

```

short-em-nolong

```

9411 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9412 \newabbreviationstyle{short-em-desc}%
9413 {%
9414   \renewcommand*{\CustomAbbreviationFields}{%
9415     name={\glxtrshortdescname},
9416     sort={\the\glssshorttok},
9417     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9418     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9419     text={\protect\glsabbrvemfont{\the\glssshorttok}},
9420     plural={\protect\glsabbrvemfont{\the\glssshortpltok}},
9421     description={\the\glslongtok}}%
9422   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9423     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9424 }%
9425 {%
9426   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9427   \renewcommand*{\glsabbrvfnt}[1]{\glsabbrvemfont{##1}}%
9428   \renewcommand*{\glsfirstabbrvfnt}[1]{\glsfirstabbrvemfont{##1}}%
9429   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9430   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9431 \renewcommand*{\glxtrinlinefullformat}[2]{%
9432   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```

```

9433     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9434     \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
9435 }%
9436 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9437     \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9438     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9439     \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9440 }%
9441 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9442     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9443     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9444     \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
9445 }%
9446 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9447     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9448     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9449     \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
9450 }%

```

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.

```

9451 \renewcommand*{\glxtrfullformat}[2]{%
9452     \glsfirstabbrvemfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9453     \ifglxtrinsertinside\else##2\fi
9454 }%
9455 \renewcommand*{\glxtrfullplformat}[2]{%
9456     \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9457     \ifglxtrinsertinside\else##2\fi
9458 }%
9459 \renewcommand*{\Glsxtrfullformat}[2]{%
9460     \glsfirstabbrvemfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9461     \ifglxtrinsertinside\else##2\fi
9462 }%
9463 \renewcommand*{\Glsxtrfullplformat}[2]{%
9464     \glsfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9465     \ifglxtrinsertinside\else##2\fi
9466 }%
9467 }

```

-em-nolong-desc

```

9468 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9469 \newabbreviationstyle{nolong-short-em}%
9470 {%
9471     \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9472 }%
9473 {%
9474     \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9475 \renewcommand*{\glxtrinlinefullformat}[2]{%
9476   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9477     \ifglxtrinsertinside##2\fi}%
9478   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9479   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9480 }%
9481 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9482   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9483     \ifglxtrinsertinside##2\fi}%
9484   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9485   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9486 }%
9487 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9488   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9489     \ifglxtrinsertinside##2\fi}%
9490   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9491   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9492 }%
9493 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9494   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9495     \ifglxtrinsertinside##2\fi}%
9496   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9497   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9498 }%
9499 }

```

`long-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```

9500 \newabbreviationstyle{long-noshort-em}%
9501 {%
9502   \renewcommand*{\CustomAbbreviationFields}{%
9503     name={\glxtrlongnoshortname},
9504     sort={\the\glsshorttok},
9505     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9506     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9507     text={\protect\glslongdefaultfont{\the\glslongtok}},
9508     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9509     description={\the\glslongtok}%
9510   }%
9511   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9512     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9513 }%
9514 {%
9515   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9516   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9517   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9518   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9519   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).


```

9520 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9521   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9522   \ifglxtrinsertinside \else##2\fi
9523 }%
9524 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9525   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9526   \ifglxtrinsertinside \else##2\fi
9527 }%
9528 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9529   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9530   \ifglxtrinsertinside \else##2\fi
9531 }%
9532 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9533   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9534   \ifglxtrinsertinside \else##2\fi
9535 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9536 \renewcommand*{\glxtrinlinefullformat}[2]{%
9537   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9538   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9539   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9540 }%
9541 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9542   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9543   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9544   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9545 }%
9546 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9547   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9548   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9549   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9550 }%
9551 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9552   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9553   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9554   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9555 }%

```

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.

```

9556 \renewcommand*{\glxtrfullformat}[2]{%
9557   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9558   \ifglxtrinsertinside\else##2\fi
9559 }%
9560 \renewcommand*{\glxtrfullplformat}[2]{%
9561   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9562   \ifglxtrinsertinside\else##2\fi
9563 }%
9564 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9565 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9566 \ifglxtrinsertinside\else##2\fi
9567 }%
9568 \renewcommand*{\Glsxtrfullplformat}[2]{%
9569 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9570 \ifglxtrinsertinside\else##2\fi
9571 }%
9572 }

```

long-em Backward compatibility:

```

9573 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9574 \newabbreviationstyle{long-em-noshort-em}%
9575 {%
9576 \renewcommand*{\CustomAbbreviationFields}{%
9577 name={\glxtrlongnoshortname},
9578 sort={\the\glsshorttok},
9579 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9580 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9581 text={\protect\glslongemfont{\the\glslongtok}},
9582 plural={\protect\glslongemfont{\the\glslongpltok}},%
9583 description={\protect\glslongemfont{\the\glslongtok}}%
9584 }%
9585 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9586 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9587 }%
9588 {%
9589 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9590 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9591 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9592 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9593 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9594 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9595 \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9596 \ifglxtrinsertinside \else##2\fi
9597 }%
9598 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9599 \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9600 \ifglxtrinsertinside \else##2\fi
9601 }%
9602 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9603 \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9604 \ifglxtrinsertinside \else##2\fi
9605 }%
9606 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9607 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9608 \ifglxtrinsertinside \else##2\fi

```

9609 }%

The inline full form displays the long format followed by the short form in parentheses.

```
9610 \renewcommand*{\glxstrinlinefullformat}[2]{%
9611   \glsfirslongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9612   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9613   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9614 }%
9615 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9616   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9617   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9618   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9619 }%
9620 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9621   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9622   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9623   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9624 }%
9625 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9626   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9627   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9628   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9629 }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
9630 \renewcommand*{\glxstrfullformat}[2]{%
9631   \glsfirslongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9632   \ifglxstrinsertinside\else##2\fi
9633 }%
9634 \renewcommand*{\glxstrfullplformat}[2]{%
9635   \glsfirslongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9636   \ifglxstrinsertinside\else##2\fi
9637 }%
9638 \renewcommand*{\Glsxtrfullformat}[2]{%
9639   \glsfirslongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9640   \ifglxstrinsertinside\else##2\fi
9641 }%
9642 \renewcommand*{\Glsxtrfullplformat}[2]{%
9643   \glsfirslongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9644   \ifglxstrinsertinside\else##2\fi
9645 }%
9646 }
```

oshort-em-noreg Like long-em-noshort-em but doesn't set the regular attribute.

```
9647 \newabbreviationstyle{long-em-noshort-em-noreg}%
9648 {%
9649   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```

9650 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9651   \glshasattribute{\the\glslabeltok}{regular}}%
9652   {%
9653     \glissetattribute{\the\glslabeltok}{regular}{false}}%
9654   }%
9655   {}%
9656 }%
9657 }%
9658 {%
9659   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}}%
9660 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9661 \newabbreviationstyle{long-noshort-em-desc}%
9662 {%
9663   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}}%
9664 }%
9665 {%
9666   \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9667   \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9668   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9669   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9670   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9671 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9672   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9673   \ifglsxtrininsertinside \else##2\fi
9674 }%
9675 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9676   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9677   \ifglsxtrininsertinside \else##2\fi
9678 }%
9679 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9680   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9681   \ifglsxtrininsertinside \else##2\fi
9682 }%
9683 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9684   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9685   \ifglsxtrininsertinside \else##2\fi
9686 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9687 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9688   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9689   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9690   \glsxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9691 }%
9692 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

```

```

9693 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9694 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9695 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9696 }%
9697 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9698 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9699 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9700 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9701 }%
9702 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9703 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9704 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9705 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9706 }%

```

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.

```

9707 \renewcommand*{\glxtrfullformat}[2]{%
9708 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9709 \ifglxtrinsertinside\else##2\fi
9710 }%
9711 \renewcommand*{\glxtrfullplformat}[2]{%
9712 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9713 \ifglxtrinsertinside\else##2\fi
9714 }%
9715 \renewcommand*{\Glsxtrfullformat}[2]{%
9716 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9717 \ifglxtrinsertinside\else##2\fi
9718 }%
9719 \renewcommand*{\Glsxtrfullplformat}[2]{%
9720 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9721 \ifglxtrinsertinside\else##2\fi
9722 }%
9723 }

```

long-desc-em Backward compatibility:

```
9724 \@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.

```

9725 \newabbreviationstyle{long-em-noshort-em-desc}%
9726 {%
9727 \renewcommand*{\CustomAbbreviationFields}{%
9728 name={\glxtrlongnoshortdescname},
9729 sort={\the\glslongtok},
9730 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9731 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9732 text={\glslongemfont{\the\glslongtok}},
9733 plural={\glslongemfont{\the\glslongpltok}}}%

```

```

9734 }%
9735 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9736   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9737 }%
9738 {%
9739 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9740 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9741 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9742 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
9743 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9744 \renewcommand*\glsxtrsubsequentfmt}[2]{%
9745   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9746   \ifglsxtrininsertinside \else##2\fi
9747 }%
9748 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
9749   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9750   \ifglsxtrininsertinside \else##2\fi
9751 }%
9752 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
9753   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9754   \ifglsxtrininsertinside \else##2\fi
9755 }%
9756 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
9757   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9758   \ifglsxtrininsertinside \else##2\fi
9759 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9760 \renewcommand*\glsxtrininlinefullformat}[2]{%
9761   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9762   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9763   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9764 }%
9765 \renewcommand*\glsxtrininlinefullplformat}[2]{%
9766   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9767   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9768   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9769 }%
9770 \renewcommand*\Glsxtrininlinefullformat}[2]{%
9771   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9772   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9773   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9774 }%
9775 \renewcommand*\Glsxtrininlinefullplformat}[2]{%
9776   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9777   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9778   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9779 }%

```

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.

```

9780 \renewcommand*{\glxtrfullformat}[2]{%
9781   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9782   \ifglxtrinsertinside\else##2\fi
9783 }%
9784 \renewcommand*{\glxtrfullplformat}[2]{%
9785   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9786   \ifglxtrinsertinside\else##2\fi
9787 }%
9788 \renewcommand*{\Glsxtrfullformat}[2]{%
9789   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9790   \ifglxtrinsertinside\else##2\fi
9791 }%
9792 \renewcommand*{\Glsxtrfullplformat}[2]{%
9793   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9794   \ifglxtrinsertinside\else##2\fi
9795 }%
9796 }

```

`t-em-desc-noreg` Like `long-em-noshort-em-desc` but doesn't set the regular attribute.

```

9797 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9798 {%
9799   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

9800 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9801   \glshasattribute{\the\glslabeltok}{regular}%
9802   {%
9803     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9804   }%
9805   {}%
9806 }%
9807 }%
9808 {%
9809   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9810 }

```

`ort-em-footnote`

```

9811 \newabbreviationstyle{short-em-footnote}%
9812 {%
9813   \renewcommand*{\CustomAbbreviationFields}{%
9814     name={\glxtrfootnotename},
9815     sort={\the\glsshorttok},
9816     description={\the\glslongtok},%
9817     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9818       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9819       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9820     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%

```

```

9821 \protect\glstrabbrvfootnote{\the\glslabeltok}%
9822 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9823 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.

```

9824 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9825 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9826 \glsattribute{\the\glslabeltok}{regular}%
9827 {%
9828 \glssetattribute{\the\glslabeltok}{regular}{false}%
9829 }%
9830 {}%
9831 }%
9832 }%
9833 {%
9834 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9835 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9836 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9837 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9838 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9839 \renewcommand*{\glstrfullformat}[2]{%
9840 \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglstrinsertinside##2\fi}%
9841 \ifglstrinsertinside\else##2\fi
9842 \protect\glstrabbrvfootnote{##1}%
9843 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9844 }%
9845 \renewcommand*{\glstrfullplformat}[2]{%
9846 \glsfirstabbrvemfont{\glsaccessshortpl{##1}}\ifglstrinsertinside##2\fi}%
9847 \ifglstrinsertinside\else##2\fi
9848 \protect\glstrabbrvfootnote{##1}%
9849 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9850 }%
9851 \renewcommand*{\GlsXtrfullformat}[2]{%
9852 \glsfirstabbrvemfont{\Glsaccessshort{##1}}\ifglstrinsertinside##2\fi}%
9853 \ifglstrinsertinside\else##2\fi
9854 \protect\glstrabbrvfootnote{##1}%
9855 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9856 }%
9857 \renewcommand*{\GlsXtrfullplformat}[2]{%
9858 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglstrinsertinside##2\fi}%
9859 \ifglstrinsertinside\else##2\fi
9860 \protect\glstrabbrvfootnote{##1}%
9861 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9862 }%

```

The first use full form and the inline full form use the short (long) style.

```

9863 \renewcommand*{\glstrinlinefullformat}[2]{%
9864 \glsfirstabbrvemfont{\glsaccessshort{##1}}\ifglstrinsertinside##2\fi}%

```



```

9865     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9866     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9867 }%
9868 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9869     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9870     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9871     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9872 }%
9873 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9874     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9875     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9876     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9877 }%
9878 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9879     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9880     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9881     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9882 }%
9883 }

```

footnote-em Backward compatibility:

```

9884 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9885 \newabbreviationstyle{short-em-postfootnote}%
9886 {%
9887   \renewcommand*{\CustomAbbreviationFields}{%
9888     name={\glxtrfootnotename},
9889     sort={\the\glssshorttok},
9890     description={\the\glslongtok},%
9891     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},%
9892     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},%
9893     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.

```

9894 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9895   \csdef{glxtrpostlink\glscategorylabel}{%
9896     \glxtrifwasfirstuse
9897     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9898     \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9899     {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
9900   }%
9901 }%
9902 }%
9903 \glshasattribute{\the\glslabeltok}{regular}%
9904 {%

```

```

9905     \glsetattribute{\the\glslabeltok}{regular}{false}%
9906   }%
9907   {}%
9908 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9909 \renewcommand*\glxtrsetupfulldefs{%
9910   \let\glxtrifwasfirstuse\@secondoftwo
9911 }%
9912}%
9913{%
9914 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
9915 \renewcommand*\glabbrvfont[1]{\glabbrvemfont{##1}}%
9916 \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvemfont{##1}}%
9917 \renewcommand*\glfirstlongfont[1]{\glfirstlongfootnotefont{##1}}%
9918 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9919 \renewcommand*\glxtrfullformat[2]{%
9920   \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9921   \ifglxtrininsertinside\else##2\fi
9922 }%
9923 \renewcommand*\glxtrfullplformat[2]{%
9924   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9925   \ifglxtrininsertinside\else##2\fi
9926 }%
9927 \renewcommand*\Glsxtrfullformat[2]{%
9928   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9929   \ifglxtrininsertinside\else##2\fi
9930 }%
9931 \renewcommand*\Glsxtrfullplformat[2]{%
9932   \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9933   \ifglxtrininsertinside\else##2\fi
9934 }%

```

The first use full form and the inline full form use the short (long) style.

```

9935 \renewcommand*\glxtrininlinefullformat[2]{%
9936   \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9937   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9938   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
9939 }%
9940 \renewcommand*\glxtrininlinefullplformat[2]{%
9941   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9942   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9943   \glxtrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
9944 }%
9945 \renewcommand*\Glsxtrininlinefullformat[2]{%
9946   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9947   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9948   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%

```

```

9949 }%
9950 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9951   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9952   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9953   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9954 }%
9955 }

```

postfootnote-em Backward compatibility:

```

9956 \@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.

```

9957 \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.

```

9958 \ifdef\glscurrentfieldvalue
9959 {
9960   \newcommand*{\glxtruserparen}[2]{%
9961     \glsxtrfullsep{#2}%
9962     \glsxtrparen
9963       {#1\ifglshasfield{\glxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9964   }
9965 }
9966 {
9967   \newcommand*{\glxtruserparen}[2]{%
9968     \glsxtrfullsep{#2}%
9969     \glsxtrparen
9970       {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9971   }
9972 }

```

Font used for short form:

lsabbrvuserfont

```

9973 \newcommand*{\glsabrvuserfont}[1]{\glsabrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

9974 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
9975 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
9976 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
9977 \newcommand*{\lsxtrusersuffix}{\lsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
9978 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
9979 \newabbreviationstyle{long-short-user}%
9980 {%
9981   \renewcommand*{\CustomAbbreviationFields}{%
9982     name={\glsxtrlongshortname},
9983     sort={\the\glsshorttok},
9984     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9985       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
9986     {\the\glslabeltok}},%
9987     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9988       \protect\glsxtruserparen
9989       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9990     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9991     description={\protect\glsuserdescription{\the\glslongtok}%
9992       {\the\glslabeltok}}}%

```

Unset the regular attribute if it has been set.

```
9993   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9994     \glshasattribute{\the\glslabeltok}{regular}%
9995     {%
9996       \glissetattribute{\the\glslabeltok}{regular}{false}%
9997     }%
9998   }%
9999 }%
10000}%
10001{%

```

In case the user wants to mix and match font styles, these are redefined here.

```
10002   \renewcommand*{\abbrvpluralsuffix}{\lsxtrusersuffix}%
10003   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10004   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10005   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10006   \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

10007 \renewcommand*{\glxtrfullformat}[2]{%
10008   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10009   \ifglxtrinsertinside\else##2\fi
10010   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10011 }%
10012 \renewcommand*{\glxtrfullplformat}[2]{%
10013   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10014   \ifglxtrinsertinside\else##2\fi
10015   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10016 }%
10017 \renewcommand*{\Glsxtrfullformat}[2]{%
10018   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10019   \ifglxtrinsertinside\else##2\fi
10020   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10021 }%
10022 \renewcommand*{\Glsxtrfullplformat}[2]{%
10023   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10024   \ifglxtrinsertinside\else##2\fi
10025   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10026 }%
10027 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10028 \newabbreviationstyle{long-postshort-user}%
10029 {%
10030   \renewcommand*{\CustomAbbreviationFields}{%
10031     name={\glxtrlongshortname},
10032     sort={\the\glsshorttok},
10033     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10034     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10035     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10036     description={\protect\glsuserdescription{\the\glslongtok}%
10037       {\the\glslabeltok}}}%
10038   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10039     \csdef{glxtrpostlink\glscategorylabel}{%
10040       \glxtrifwasfirstuse
10041       {%
10042         \glxtruserparen
10043         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10044         {\glslabel}%
10045       }%
10046       {}}%
10047   }%
10048   \glsasattribute{\the\glslabeltok}{regular}%
10049   {%
10050     \glssetattribute{\the\glslabeltok}{regular}{false}%
10051   }%

```

```

10052    {}%
10053  }%
10054 }%
10055 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10056 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10057 \renewcommand*{\glsabrvfont}[1]{\glsabrvuserfont{##1}}%
10058 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10059 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10060 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10061 \renewcommand*{\glxtrfullformat}[2]{%
10062   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10063   \ifglxtrininsertinside\else##2\fi
10064 }%
10065 \renewcommand*{\glxtrfullplformat}[2]{%
10066   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10067   \ifglxtrininsertinside\else##2\fi
10068 }%
10069 \renewcommand*{\Glsxtrfullformat}[2]{%
10070   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10071   \ifglxtrininsertinside\else##2\fi
10072 }%
10073 \renewcommand*{\Glsxtrfullplformat}[2]{%
10074   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10075   \ifglxtrininsertinside\else##2\fi
10076 }%

```

In-line format:

```

10077 \renewcommand*{\glxtrinlinefullformat}[2]{%
10078   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10079   \ifglxtrininsertinside\else##2\fi
10080   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10081 }%
10082 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10083   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10084   \ifglxtrininsertinside\else##2\fi
10085   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10086 }%
10087 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10088   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10089   \ifglxtrininsertinside\else##2\fi
10090   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10091 }%
10092 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10093   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10094   \ifglxtrininsertinside\else##2\fi
10095   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%

```

```

10096 }%
10097 }

```

ortuserdescname

```

10098 \newcommand*{\glxtrlongshortuserdescname}{%
10099   \protect\glslonguserfont{\the\glslongtok}%
10100   \protect\glxtruserparen
10101   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10102 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10103 \newabbreviationstyle{long-postshort-user-desc}%
10104 {%
10105   \renewcommand*{\CustomAbbreviationFields}{%
10106     name={\glxtrlongshortuserdescname},
10107     sort={\the\glslongtok},
10108     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10109     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10110     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10111     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10112 }%
10113 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10114   \csdef{glxtrpostlink\glscategorylabel}{%
10115     \glxtrifwasfirstuse
10116     {%
10117       \glxtruserparen
10118       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10119       {\glslabel}%
10120     }%
10121     {}%
10122   }%
10123   \glshasattribute{\the\glslabeltok}{regular}%
10124   {%
10125     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10126   }%
10127   {}%
10128 }%
10129 }%
10130 {%
10131   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10132 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10133 \newabbreviationstyle{short-postlong-user}%
10134 {%
10135   \renewcommand*{\CustomAbbreviationFields}{%
10136     name={\glxtrshortlongname},
10137     sort={\the\glsshorttok},

```

```

10138 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10139 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10140 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10141 description={\protect\glsuserdescription{\the\glslongtok}%
10142 {\the\glslabeltok}}}%
10143 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10144 \csdef{glsxtrpostlink\glscategorylabel}{%
10145 \glsxtrifwasfirstuse
10146 {%
10147 \glsxtruserparen
10148 {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10149 {\glslabel}}%
10150 }%
10151 {}%
10152 }%
10153 \glsattribute{\the\glslabeltok}{regular}%
10154 {%
10155 \glssetattribute{\the\glslabeltok}{regular}{false}%
10156 }%
10157 {}%
10158 }%
10159}%
10160{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10161 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10162 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10163 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10164 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10165 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10166 \renewcommand*{\glsxtrfullformat}[2]{%
10167 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10168 \ifglsxtrininsertinside\else##2\fi
10169}%
10170 \renewcommand*{\glsxtrfullplformat}[2]{%
10171 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10172 \ifglsxtrininsertinside\else##2\fi
10173}%
10174 \renewcommand*{\Glsxtrfullformat}[2]{%
10175 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10176 \ifglsxtrininsertinside\else##2\fi
10177}%
10178 \renewcommand*{\Glsxtrfullplformat}[2]{%
10179 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10180 \ifglsxtrininsertinside\else##2\fi
10181}%

```

In-line format:


```

10182 \renewcommand*{\glxtrinlinefullformat}[2]{%
10183   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10184   \ifglxtrininsertinside\else##2\fi
10185   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10186 }%
10187 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10188   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10189   \ifglxtrininsertinside\else##2\fi
10190   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10191 }%
10192 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10193   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10194   \ifglxtrininsertinside\else##2\fi
10195   \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10196 }%
10197 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10198   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10199   \ifglxtrininsertinside\else##2\fi
10200   \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10201 }%
10202 }

```

onguserdesname

```

10203 \newcommand*{\glxtrshortlonguserdesname}{%
10204   \protect\glssabbrvuserfont{\the\glssshorttok}%
10205   \protect\glsxtruserparen
10206     {\protect\glslonguserfont{\the\glslongpltok}}}%
10207     {\the\glslabelltok}%
10208 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10209 \newabbreviationstyle{short-postlong-user-desc}%
10210 {%
10211   \renewcommand*{\CustomAbbreviationFields}{%
10212     name={\glxtrshortlonguserdesname},
10213     sort={\the\glssshorttok},
10214     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10215     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10216     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10217     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10218 }%
10219 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10220   \csdef{glxtrpostlink\glscategorylabel}{%
10221     \glsxtrifwasfirstuse
10222     {%
10223       \glsxtruserparen
10224         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10225         {\glslabel}%

```

```

10226     }%
10227     {}%
10228     }%
10229     \glshasattribute{\the\glslabeltok}{regular}%
10230     {%
10231         \glissetattribute{\the\glslabeltok}{regular}{false}%
10232     }%
10233     {}%
10234 }%
10235 }%
10236 {%
10237     \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10238 }

```

short-user-desc

```

10239 \newabbreviationstyle{long-short-user-desc}%
10240 {%
10241     \renewcommand*{\CustomAbbreviationFields}{%
10242         name={\glsxtrlongshortuserdescname},
10243         sort={\glsxtrlongshortdescsort},%

10244         first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10245             \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10246             {\the\glslabeltok}},%
10247         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10248             \protect\glsxtruserparen
10249             {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10250         text={\protect\glsabbrvfont{\the\glsshorttok}},%
10251         plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10252     }%

```

Unset the regular attribute if it has been set.

```

10253     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10254         \glshasattribute{\the\glslabeltok}{regular}%
10255         {%
10256             \glissetattribute{\the\glslabeltok}{regular}{false}%
10257         }%
10258         {}%
10259     }%
10260 }%
10261 {%
10262     \GlsXtrUseAbbrStyleFmts{long-short-user}%
10263 }

```

short-long-user

```

10264 \newabbreviationstyle{short-long-user}%
10265 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10266 \renewcommand*{\CustomAbbreviationFields}{%
10267     name={\glxtrshortlongname},
10268     sort={\the\glsshorttok},
10269     description={\protect\gluserdescription{\the\glslongtok}%
10270         {\the\glslabeltok}},%
10271     first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10272         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}%
10273         {\the\glslabeltok}},%
10274     firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10275         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}%
10276         {\the\glslabeltok}},%
10277     plural={\protect\glabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10278 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10279     \glshasattribute{\the\glslabeltok}{regular}%
10280     {%
10281         \glissetattribute{\the\glslabeltok}{regular}{false}%
10282     }%
10283     {}%
10284 }%
10285 }%
10286 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10287 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10288 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
10289 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvuserfont{##1}}%
10290 \renewcommand*{\glfirstlongfont}[1]{\glfirstlonguserfont{##1}}%
10291 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10292 \renewcommand*{\glxtrfullformat}[2]{%
10293     \glfirstabbrvuserfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10294     \ifglxtrinsertinside\else##2\fi
10295     \glxtruserparen{\glfirstlonguserfont{\glaccesslong{##1}}}{##1}%
10296 }%
10297 \renewcommand*{\glxtrfullplformat}[2]{%
10298     \glfirstabbrvuserfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10299     \ifglxtrinsertinside\else##2\fi
10300     \glxtruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10301 }%
10302 \renewcommand*{\Glsxtrfullformat}[2]{%
10303     \glfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10304     \ifglxtrinsertinside\else##2\fi
10305     \glxtruserparen{\glfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10306 }%
10307 \renewcommand*{\Glsxtrfullplformat}[2]{%
10308     \glfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10309     \ifglxtrinsertinside\else##2\fi

```

```

10310 \glstruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10311 }%
10312 }

```

-long-user-desc

```

10313 \newabbreviationstyle{short-long-user-desc}%
10314 {%
10315 \renewcommand*{\CustomAbbreviationFields}{%
10316 name={\glstrshortlonguserdescname},
10317 sort={\glstrshortlongdescsort},%

10318 first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10319 \protect\glstruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}}%
10320 {\the\glslabeltok}},%
10321 firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10322 \protect\glstruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}}%
10323 {\the\glslabeltok}},%
10324 text={\protect\glabbrvfont{\the\glsshorttok}},%
10325 plural={\protect\glabbrvfont{\the\glsshortpltok}}}%
10326 }%

```

Unset the regular attribute if it has been set.

```

10327 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10328 \glshasattribute{\the\glslabeltok}{regular}%
10329 {%
10330 \glissetattribute{\the\glslabeltok}{regular}{false}%
10331 }%
10332 {}%
10333 }%
10334 }%
10335 {%
10336 \GlsXtrUseAbbrStyleFmts{short-long-user}%
10337 }

```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the markwords attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glstrlong` set `\glinsert` to empty with the entire link-text stored in `\glscustomtext`.

`\trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glinsert` so check for that and expand.

```

10338 \newrobustcmd*{\glstrifhyphenstart}[3]{%
10339 \ifx\glinsert#1\relax
10340 \expandafter\@glstrifhyphenstart#1\relax\relax
10341 \@end@glstrifhyphenstart{#2}{#3}%
10342 \else
10343 \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%

```

```
10344 \fi
10345 }
```

trifhyphenstart

```
10346 \def\@glxtrifhyphenstart#1#2\end@glxtrifhyphenstart#3#4{%
10347 \ifx-#1\relax#3\else #4\fi
10348 }
```

rlonghyphenshort

```
\glxtrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10349 \newcommand*\glxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10350 {%
```

If *<insert>* starts with a hyphen, redefine `\glxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxtrwordsep` if *<insert>* doesn't start with a hyphen.

```
10351 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
10352 \glsfirslonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10353 \ifglxtrininsertinside\else{#4}\fi
10354 \glxtrfullsep{#1}%
10355 \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10356 \ifglxtrininsertinside\else{#4}\fi}%
10357 }%
10358 }
```

abbrvhyphenfont

```
10359 \newcommand*\glsabbrvhyphenfont{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10360 \newcommand*\glsfirstabbrvhyphenfont{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10361 \newcommand*\glslonghyphenfont{\glslongdefaultfont}%
```

tlonghyphenfont

```
10362 \newcommand*\glsfirslonghyphenfont{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10363 \newcommand*\glxtrhyphensuffix{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the keywords attribute.

```
10364 \newabbreviationstyle{long-hyphen-short-hyphen}%
10365 {%
10366   \renewcommand*{\CustomAbbreviationFields}{%
10367     name={\glxtrlongshortname},
10368     sort={\the\glsshorttok},
10369     first={\protect\glxtrfirstlonghyphenfont{\the\glslongtok}%
10370       \protect\glxtrfullsep{\the\glslabeltok}%
10371       \glxtrparen{\protect\glxtrfirstabbrvhyphenfont{\the\glsshorttok}}},%
10372     firstplural={\protect\glxtrfirstlonghyphenfont{\the\glslongpltok}%
10373       \protect\glxtrfullsep{\the\glslabeltok}%
10374       \glxtrparen{\protect\glxtrfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10375     plural={\protect\glxtrabbrvhyphenfont{\the\glsshortpltok}},%
10376     description={\protect\glxtrlonghyphenfont{\the\glslongtok}}}%
10377 }
```

Unset the regular attribute if it has been set.

```
10377 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10378   \glshasattribute{\the\glslabeltok}{regular}%
10379   {%
10380     \glissetattribute{\the\glslabeltok}{regular}{false}%
10381   }%
10382 }%
10383 }%
10384 }%
10385 {%
10386   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10387   \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvhyphenfont{##1}}%
10388   \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvhyphenfont{##1}}%
10389   \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlonghyphenfont{##1}}%
10390   \renewcommand*{\glxtrlongfont}[1]{\glxtrlonghyphenfont{##1}}%
10391 }
```

The first use full form and the inline full form are the same for this style.

```
10391 \renewcommand*{\glxtrfullformat}[2]{%
10392   \glxtrlonghyphenshort{##1}{\glxtraccesslong{##1}}{\glxtraccessshort{##1}}{##2}%
10393 }%
10394 \renewcommand*{\glxtrfullplformat}[2]{%
10395   \glxtrlonghyphenshort{##1}{\glxtraccesslongpl{##1}}%
10396   {\glxtraccessshortpl{##1}}{##2}%
10397 }%
10398 \renewcommand*{\GlsXtrfullformat}[2]{%
10399   \glxtrlonghyphenshort{##1}{\glxtraccesslong{##1}}{\glxtraccessshort{##1}}{##2}%
10400 }%
10401 \renewcommand*{\GlsXtrfullplformat}[2]{%
10402   \glxtrlonghyphenshort{##1}{\glxtraccesslongpl{##1}}%
10403   {\glxtraccessshortpl{##1}}{##2}%
10404 }%
10405 }
```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```
10406 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10407 }
```

```

10407 {%
10408   \renewcommand*{\CustomAbbreviationFields}{%
10409     name={\glxtrlongshortdescname},
10410     sort={\glxtrlongshortdescsort},
10411     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10412       \protect\glxtrfullsep{\the\glslabeltok}%
10413       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10414     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10415       \protect\glxtrfullsep{\the\glslabeltok}%
10416       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10417     text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
10418     plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
10419   }%

```

Unset the regular attribute if it has been set.

```

10420   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10421     \glshasattribute{\the\glslabeltok}{regular}%
10422     {%
10423       \glissetattribute{\the\glslabeltok}{regular}{false}%
10424     }%
10425   }%
10426 }%
10427 }%
10428 {%
10429   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10430 }

```

onghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```

10431 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10432 {%

```

If *<insert>* starts with a hyphen, redefine `\glxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxtrwordsep` if *<insert>* doesn't start with a hyphen.

```

10433   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10434     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10435     \ifglxtrininsertinsideelse{#3}\fi
10436   }%
10437 }

```

short-desc-noreg

This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```

10438 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10439 {%
10440   \renewcommand*{\CustomAbbreviationFields}{%
10441     name={\glxtrlongnoshortdescname},
10442     sort={\expandonce\glxtrorglong},
10443     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10444     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10445     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10446   }%

```

Unset the regular attribute if it has been set.

```

10447   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10448     \glshasattribute{\the\glslabeltok}{regular}%
10449     {%
10450       \glissetattribute{\the\glslabeltok}{regular}{false}%
10451     }%
10452   }%
10453 }%
10454 }%
10455 {%
10456   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10457   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10458   \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
10459   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10460   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10461   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10462   \renewcommand*{\glxtrsubsequentfmt}[2]{%
10463     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10464   }%
10465   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10466     \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10467   }%
10468   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10469     \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10470   }%
10471   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10472     \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10473   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10474   \renewcommand*{\glxtrinilinefullformat}[2]{%
10475     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10476     \glxtrfullsep{##1}%
10477     \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10478   }%
10479   \renewcommand*{\glxtrinilinefullplformat}[2]{%
10480     \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%

```



```

10481 \glstrfullsep{##1}%
10482 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10483 }%
10484 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10485 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10486 \glstrfullsep{##1}%
10487 \glstrparen{\protect\glfirstabbrvfont{\glaccessshort{##1}}}%
10488 }%
10489 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10490 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10491 \glstrfullsep{##1}%
10492 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10493 }%

```

The first use full form only displays the long form.

```

10494 \renewcommand*{\glstrfullformat}[2]{%
10495 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10496 }%
10497 \renewcommand*{\glstrfullplformat}[2]{%
10498 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10499 }%
10500 \renewcommand*{\Glsxtrfullformat}[2]{%
10501 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10502 }%
10503 \renewcommand*{\Glsxtrfullplformat}[2]{%
10504 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10505 }%
10506 }

```

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.

```

10507 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10508 {%
10509 \renewcommand*{\CustomAbbreviationFields}{%
10510 name={\glstrlongnoshortname},
10511 sort={\the\glsshorttok},
10512 first={\protect\glfirstlonghyphenfont{\the\glslongtok}},%
10513 firstplural={\protect\glfirstlonghyphenfont{\the\glslongpltok}},%
10514 text={\protect\glslonghyphenfont{\the\glslongtok}},%
10515 plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10516 description={\the\glslongtok}%
10517 }%

```

Unset the regular attribute if it has been set.

```

10518 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10519 \glshasattribute{\the\glslabeltok}{regular}%
10520 {%
10521 \glissetattribute{\the\glslabeltok}{regular}{false}%
10522 }%

```

```

10523     {}%
10524   }%
10525 }%
10526 {%
10527   \GlsXtrUseAbbrStyleFmts{long-desc}%
10528 }

```

glsxtrlonghyphen `\glsxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10529 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10530  {%
10531    \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10532    \glsfirstlonghyphenfont{#1}%
10533  }%
10534 }

```

rpostthyphenshort `\glsxtrpostthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glsxtrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```

10535 \newcommand*{\glsxtrpostthyphenshort}[2]{%
10536  {%
10537    \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10538    \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10539    \glsxtrfullsep{#1}%
10540    \glsxtrparen
10541    {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10542    \ifglsxtrininsertinside\else{#2}\fi
10543  }%
10544 }%
10545 }

```

hyphensubsequent `\glsxtrpostthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10546 \newcommand*{\glsxtrpostthyphensubsequent}[2]{%
10547   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%

```

```

10548 \ifglxtrinsertinside \else{#2}\fi
10549 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10550 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10551 {%
10552   \renewcommand*{\CustomAbbreviationFields}{%
10553     name={\glxtrlongshortname},
10554     sort={\the\glsshorttok},
10555     first={\protect\glxtrfirstlonghyphenfont{\the\glslongtok}},%
10556     firstplural={\protect\glxtrfirstlonghyphenfont{\the\glslongpltok}},%
10557     plural={\protect\glxtrabbrvhyphenfont{\the\glsshortpltok}},%
10558     description={\protect\glxtrlonghyphenfont{\the\glslongtok}}}%
10559   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10560     \csdef{glxtrpostlink\glscategorylabel}{%
10561       \glxtrifwasfirstuse
10562       {%
10563         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10564       }%
10565     }%

```

Put the insertion into the post-link:

```

10566       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10567     }%
10568   }%
10569   \glshasattribute{\the\glslabeltok}{regular}%
10570   {%
10571     \glissetattribute{\the\glslabeltok}{regular}{false}%
10572   }%
10573   {}%
10574 }%
10575 }%
10576 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10577 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10578 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10579 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10580 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10581 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10582 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10583   \glsabbrvfont{\glsaccessshort{##1}}%
10584 }%
10585 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10586   \glsabbrvfont{\glsaccessshortpl{##1}}%
10587 }%
10588 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

```

```

10589 \glsabbrvfont{\Glsaccessshort{##1}}%
10590 }%
10591 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10592 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10593 }%

First use full form:

10594 \renewcommand*{\glsxtrfullformat}[2]{%
10595 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10596 }%
10597 \renewcommand*{\glsxtrfullplformat}[2]{%
10598 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10599 }%
10600 \renewcommand*{\Glsxtrfullformat}[2]{%
10601 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10602 }%
10603 \renewcommand*{\Glsxtrfullplformat}[2]{%
10604 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10605 }%

```

In-line format.

```

10606 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10607 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10608 \ifglsxtrininsertinside{##2}\fi}%
10609 \ifglsxtrininsertinside \else{##2}\fi
10610 }%
10611 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10612 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10613 \ifglsxtrininsertinside{##2}\fi}%
10614 \ifglsxtrininsertinside \else{##2}\fi
10615 }%
10616 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10617 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10618 \ifglsxtrininsertinside{##2}\fi}%
10619 \ifglsxtrininsertinside \else{##2}\fi
10620 }%
10621 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10622 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10623 \ifglsxtrininsertinside{##2}\fi}%
10624 \ifglsxtrininsertinside \else{##2}\fi
10625 }%
10626 }

```

`ort-hyphen-desc` Like `long-hyphen-postshort-hyphen` but the description must be supplied by the user.

```

10627 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10628 {%
10629 \renewcommand*{\CustomAbbreviationFields}{%
10630 name={\glsxtrlongshortdescname},
10631 sort={\glsxtrlongshortdescsort},%
10632 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%

```

```

10633   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10634   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10635   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10636 }%
10637 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10638   \csdef{glxtrpostlink\glscategorylabel}{%
10639     \glxtrifwasfirstuse
10640     {%
10641       \glxtrposthyphenshort{\glslabel}{\glsinsert}}%
10642     }%
10643     {%

```

Put the insertion into the post-link:

```

10644       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
10645     }%
10646   }%
10647   \glshasattribute{\the\glslabeltok}{regular}%
10648   {%
10649     \glissetattribute{\the\glslabeltok}{regular}{false}%
10650   }%
10651   {}%
10652 }%
10653 }%
10654 {%
10655   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10656 }

```

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.

```
10657 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10658 {%
```

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.)

```

10659   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}}%
10660   \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10661   \ifglxtrininsertinside\else{#4}\fi
10662   \glxtrfullsep{#1}%
10663   \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10664     \ifglxtrininsertinside\else{#4}\fi}%
10665 }%
10666 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```

10667 \newabbreviationstyle{short-hyphen-long-hyphen}%
10668 {%
10669   \renewcommand*{\CustomAbbreviationFields}{%
10670     name={\glxstrshortlongname},
10671     sort={\the\glsshorttok},
10672     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10673       \protect\glxstrfullsep{\the\glslabeltok}%
10674       \glxstrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10675     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10676       \protect\glxstrfullsep{\the\glslabeltok}%
10677       \glxstrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10678     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10679     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10680 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10681   \glshasattribute{\the\glslabeltok}{regular}%
10682   {%
10683     \glissetattribute{\the\glslabeltok}{regular}{false}%
10684   }%
10685   {}%
10686 }%
10687 }%
10688 {%
10689   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10690   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10691   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10692   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10693   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10694 \renewcommand*{\glxstrfullformat}[2]{%
10695   \glxstrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10696 }%
10697 \renewcommand*{\glxstrfullplformat}[2]{%
10698   \glxstrshorthyphenlong{##1}%
10699   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10700 }%
10701 \renewcommand*{\Glsxtrfullformat}[2]{%
10702   \glxstrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10703 }%
10704 \renewcommand*{\Glsxtrfullplformat}[2]{%
10705   \glxstrshorthyphenlong{##1}%
10706   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10707 }%
10708 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10709 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%

```

```

10710 {%
10711   \renewcommand*{\CustomAbbreviationFields}{%
10712     name={\glxtrshortlongdescname},
10713     sort={\glxtrshortlongdescsort},
10714     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10715       \protect\glxtrfullsep{\the\glslabeltok}%
10716       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10717     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10718       \protect\glxtrfullsep{\the\glslabeltok}%
10719       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10720     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10721     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10722   }%

```

Unset the regular attribute if it has been set.

```

10723   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10724     \glshasattribute{\the\glslabeltok}{regular}%
10725     {%
10726       \glissetattribute{\the\glslabeltok}{regular}{false}%
10727     }%
10728   }%
10729 }%
10730 }%
10731 {%
10732   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10733 }

```

`\glxtrshorthyphen`

```
\glxtrshorthyphen{<short>}{<label>}{<insert>}
```

Used by short-hyphen-postlong-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
10734 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

10735   {%
10736     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}}%
10737     \glsfirstabbrvhyphenfont{#1}%
10738   }%
10739 }

```

`\glxtrposthyphenlong`

```
\glxtrposthyphenlong{<label>}{<insert>}
```

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the *<short>* part. This always uses the singular long form.

```

10740 \newcommand*{\glxtrposthyphenlong}[2]{%
10741  {%
10742   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{%
10743    \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10744    \glxtrfullsep{#1}%
10745    \glxtrparen
10746    {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
10747    \ifglxtrininsertinside\else{#2}\fi
10748   }%
10749 }%
10750 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10751 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10752 {%
10753  \renewcommand*{\CustomAbbreviationFields}{%
10754   name={\glxtrshortlongname},
10755   sort={\the\glsshorttok},
10756   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10757   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10758   plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
10759   description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10760  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10761   \csdef{glxtrpostlink\glscategorylabel}{%
10762     \glxtrifwasfirstuse
10763     {%
10764       \glxtrposthyphenlong{\glslabel}{\glinsert}%
10765     }%
10766     {%

```

Put the insertion into the post-link:

```

10767       \glxtrposthyphenlong{\glslabel}{\glinsert}%
10768     }%
10769   }%
10770   \glshasattribute{\the\glslabeltok}{regular}%
10771   {%
10772     \glissetattribute{\the\glslabeltok}{regular}{false}%
10773   }%
10774   {}%
10775 }%
10776 }%
10777 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10778 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10779 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
10780 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10781 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10782 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```


Subsequent use needs to omit the insertion:

```

10783 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10784   \glsabbrvfont{\glsaccessshort{##1}}%
10785 }%
10786 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10787   \glsabbrvfont{\glsaccessshortpl{##1}}%
10788 }%
10789 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10790   \glsabbrvfont{\Glsaccessshort{##1}}%
10791 }%
10792 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10793   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10794 }%

```

First use full form:

```

10795 \renewcommand*{\glxtrfullformat}[2]{%
10796   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10797 }%
10798 \renewcommand*{\glxtrfullplformat}[2]{%
10799   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10800 }%
10801 \renewcommand*{\Glsxtrfullformat}[2]{%
10802   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10803 }%
10804 \renewcommand*{\Glsxtrfullplformat}[2]{%
10805   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10806 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10807 \renewcommand*{\glxtrinlinefullformat}[2]{%
10808   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10809   \ifglxtrininsertinside{##2}\fi}%
10810   \ifglxtrininsertinside \else{##2}\fi
10811 }%
10812 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10813   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10814   \ifglxtrininsertinside{##2}\fi}%
10815   \ifglxtrininsertinside \else{##2}\fi
10816 }%
10817 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10818   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10819   \ifglxtrininsertinside{##2}\fi}%
10820   \ifglxtrininsertinside \else{##2}\fi
10821 }%
10822 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10823   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10824   \ifglxtrininsertinside{##2}\fi}%
10825   \ifglxtrininsertinside \else{##2}\fi
10826 }%

```

10827 }

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

10828 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}{%

10829 {%

10830 \renewcommand*{\CustomAbbreviationFields}{%

10831 name={\glstrshortlongdescname},

10832 sort={\glstrshortlongdescsort},%

10833 first={\protect\glfirstabbrvhyphenfont{\the\glsshorttok}},%

10834 firstplural={\protect\glfirstabbrvhyphenfont{\the\glsshortpltok}},%

10835 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%

10836 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%

10837 }%

10838 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

10839 \csdef{glstrpostlink\glscategorylabel}{%

10840 \glstrifwasfirstuse

10841 {%

10842 \glstrposthyphenlong{\glslabel}{\glsinsert}%

10843 }%

10844 {%

Put the insertion into the post-link:

10845 \glstrposthyphensubsequent{\glslabel}{\glsinsert}%

10846 }%

10847 }%

10848 \glshasattribute{\the\glslabeltok}{regular}%

10849 {%

10850 \glissetattribute{\the\glslabeltok}{regular}{false}%

10851 }%

10852 {}}%

10853 }%

10854 }%

10855 {%

10856 \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%

10857 }

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

10858 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%

stabbrvonlyfont

10859 \newcommand*{\glfirstabbrvonlyfont}{\glsabbrvonlyfont}%

glslongonlyfont

10860 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

rstlongonlyfont

```
10861 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10862 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
10863 \newcommand*{\glsxtronlyname}{%  
10864   \protect\glsabbrvonlyfont{\the\glsshorttok}}%  
10865 }
```

only-short-only

```
10866 \newabbreviationstyle{long-only-short-only}%  
10867 {%  
10868   \renewcommand*{\CustomAbbreviationFields}{%  
10869     name={\glsxtronlyname},  
10870     sort={\the\glsshorttok},  
10871     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%  
10872     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%  
10873     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%  
10874     description={\protect\glslongonlyfont{\the\glslongtok}}}%  
10875 }
```

Unset the regular attribute if it has been set.

```
10875 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
10876   \glshasattribute{\the\glslabeltok}{regular}}%  
10877   {%  
10878     \glissetattribute{\the\glslabeltok}{regular}{false}}%  
10879   }%  
10880   {}%  
10881 }%  
10882 }%  
10883 {%  
10884   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%  
10885   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%  
10886   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%  
10887   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%  
10888   \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%  
10889 }
```

The first use full form doesn't show the short form.

```
10889 \renewcommand*{\glsxtrfullformat}[2]{%  
10890   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%  
10891   \ifglsxtrinsertinside\else##2\fi  
10892 }%  
10893 \renewcommand*{\glsxtrfullplformat}[2]{%  
10894   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%  
10895   \ifglsxtrinsertinside\else##2\fi  
10896 }%  
10897 \renewcommand*{\Glsxtrfullformat}[2]{%  
10898   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%  
10899   \ifglsxtrinsertinside\else##2\fi  
10900 }%
```

```

10898 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10899 \ifglxtrinsertinside\else##2\fi
10900 }%
10901 \renewcommand*{\Glsxtrfullplformat}[2]{%
10902 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10903 \ifglxtrinsertinside\else##2\fi
10904 }%

```

The inline full form does show the short form.

```

10905 \renewcommand*{\glxtrinlinefullformat}[2]{%
10906 \glsfirstlongonlyfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10907 \ifglxtrinsertinside\else##2\fi
10908 \glxtrfullsep{##1}%
10909 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshort{##1}}}%
10910 }%
10911 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10912 \glsfirstlongonlyfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10913 \ifglxtrinsertinside\else##2\fi
10914 \glxtrfullsep{##1}%
10915 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
10916 }%
10917 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10918 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10919 \ifglxtrinsertinside\else##2\fi
10920 \glxtrfullsep{##1}%
10921 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10922 }%
10923 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10924 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10925 \ifglxtrinsertinside\else##2\fi
10926 \glxtrfullsep{##1}%
10927 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10928 }%
10929 }

```

xtonlydescsort

```

10930 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtonlydescname

```

10931 \newcommand*{\glxtronlydescname}{%
10932 \protect\glslongfont{\the\glslongtok}%
10933 }

```

short-only-desc

```

10934 \newabbreviationstyle{long-only-short-only-desc}%
10935 {%
10936 \renewcommand*{\CustomAbbreviationFields}{%
10937 name={\glxtronlydescname},
10938 sort={\glxtronlydescsort},%

```

```

10939   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10940   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10941   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
10942   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
10943 }%

```

Unset the regular attribute if it has been set.

```

10944 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10945   \glshasattribute{\the\glslabeltok}{regular}%
10946   {%
10947     \glssetattribute{\the\glslabeltok}{regular}{false}%
10948   }%
10949   {}%
10950 }%
10951 }%
10952 {%
10953   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10954 }

```

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:

```

10955 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```
10956 \renewcommand*{\markright}[1]{%
10957   \glxtrmarkhook
10958   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10959   \glxtrrestoremarkhook
10960 }
```

\markboth Save original definition:

```
10961 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10962 \renewcommand*{\markboth}[2]{%
10963   \glxtrmarkhook
10964   \@glxtr@org@markboth
10965   {\@glxtrinmark#1\@glxtrnotinmark}%
10966   {\@glxtrinmark#2\@glxtrnotinmark}%
10967   \glxtrrestoremarkhook
10968 }
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
10969 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
10970 \renewcommand*{\@starttoc}[1]{%
10971   \glxtrmarkhook
10972   \@glxtrinmark
10973   \@glxtr@org@@starttoc{#1}%
10974   \@glxtrnotinmark
10975   \glxtrrestoremarkhook
10976 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

glxtrRevertMarks

```
10977 \newcommand*{\glxtrRevertMarks}{%
10978   \let\markright\@glxtr@org@markright
10979   \let\markboth\@glxtr@org@markboth
10980   \let\@starttoc\@glxtr@org@@starttoc
10981 }
```

glxtrRevertTocMarks Just restores \@starttoc.

```
10982 \newcommand*{\glxtrRevertTocMarks}{%
10983   \let\@starttoc\@glxtr@org@@starttoc
10984 }
```

\glxtrifinmark

```
10985 \newcommand*{\glxtrifinmark}[2]{#2}
```

```

\@glsxtrinmark
10986 \newrobustcmd*{\@glsxtrinmark}{%
10987   \let\glsxtrifinmark\@firstoftwo
10988 }

glsxtrnotinmark
10989 \newrobustcmd*{\@glsxtrnotinmark}{%
10990   \let\glsxtrifinmark\@secondoftwo
10991 }

eorpdpforheading
10992 \ifdef\texorpdfstring
10993 {
10994   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
10995 }
10996 {
10997   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
10998 }

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply
to the marks:
10999 \newcommand*{\glsxtrmarkhook}{%
  Save current definitions:
11000   \let\@glsxtr@org@MakeUppercase\MakeUppercase
11001   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11002   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11003   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11004   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11005   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11006   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11007   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11008   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11009   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11010   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11011   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11012   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11013   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11014   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11015   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11016   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11017   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11018   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11019   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11020   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11021   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11022   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11023   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

11024 \let\glsxtrifinmark\@firstoftwo
11025 \let\MakeUppercase\MakeTextUppercase
11026 \let\glsxtrtitleorpdforheading\@thirdofthree
11027 \let\glsxtrtitleshort\glsxtrheadshort
11028 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11029 \let\Glsxtrtitleshort\Glsxtrheadshort
11030 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11031 \let\glsxtrtitlename\glsxtrheadname
11032 \let\Glsxtrtitlename\Glsxtrheadname
11033 \let\glsxtrtitletext\glsxtrheadtext
11034 \let\Glsxtrtitletext\Glsxtrheadtext
11035 \let\glsxtrtitleplural\glsxtrheadplural
11036 \let\Glsxtrtitleplural\Glsxtrheadplural
11037 \let\glsxtrtitlefirst\glsxtrheadfirst
11038 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11039 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11040 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11041 \let\glsxtrtitlelong\glsxtrheadlong
11042 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11043 \let\Glsxtrtitlelong\Glsxtrheadlong
11044 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11045 \let\glsxtrtitlefull\glsxtrheadfull
11046 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11047 \let\Glsxtrtitlefull\Glsxtrheadfull
11048 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11049 }

```

`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.)

```

11050 \newcommand*{\glsxtrrestoremarkhook}{%
11051   \let\glsxtrifinmark\@secondoftwo
11052   \let\MakeUppercase\@glsxtr@org@MakeUppercase
11053   \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
11054   \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
11055   \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
11056   \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
11057   \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
11058   \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
11059   \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
11060   \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
11061   \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
11062   \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
11063   \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
11064   \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
11065   \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
11066   \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural

```



```

11067 \let\Glsxtrtitlefirstplural\@glxstr@org@Glsxtrtitlefirstplural
11068 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
11069 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
11070 \let\Glsxtrtitlelong\@glxstr@org@Glsxtrtitlelong
11071 \let\Glsxtrtitlelongpl\@glxstr@org@Glsxtrtitlelongpl
11072 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
11073 \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
11074 \let\Glsxtrtitlefull\@glxstr@org@Glsxtrtitlefull
11075 \let\Glsxtrtitlefullpl\@glxstr@org@Glsxtrtitlefullpl
11076 }

```

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.

```

11077 \newcommand*{\glxstrheadshort}[1]{%
11078   \protect\NoCaseChange
11079   {%
11080     \gl@ifattribute{#1}{headuc}{true}%
11081     {%
11082       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11083     }%
11084     {%
11085       \glxstrshort[noindex,hyper=false]{#1}[]%
11086     }%
11087   }%
11088 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

11089 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11090   \glxstrshort[noindex,hyper=false]{#1}[]%
11091 }

```

lsxtrheadshortpl 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.

```

11092 \newcommand*{\glxstrheadshortpl}[1]{%
11093   \protect\NoCaseChange
11094   {%
11095     \gl@ifattribute{#1}{headuc}{true}%
11096     {%
11097       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11098     }%
11099     {%
11100       \glxstrshortpl[noindex,hyper=false]{#1}[]%
11101     }%
11102   }%
11103 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11104 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11105   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11106 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11107 \newcommand*{\Glsxtrheadshort}[1]{%
11108   \protect\NoCaseChange
11109   {%
11110     \glsifattribute{#1}{headuc}{true}%
11111     {%
11112       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11113     }%
11114     {%
11115       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11116     }%
11117   }%
11118 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11119 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11120   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11121 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11122 \newcommand*{\Glsxtrheadshortpl}[1]{%
11123   \protect\NoCaseChange
11124   {%
11125     \glsifattribute{#1}{headuc}{true}%
11126     {%
11127       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11128     }%
11129     {%
11130       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11131     }%
11132   }%
11133 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11134 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11135   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11136 }
```

`\glxtrheadname` As above but for the name value.

```

11137 \newcommand*{\glxtrheadname}[1]{%
11138   \protect\NoCaseChange
11139   {%
11140     \gl@ifattribute{#1}{headuc}{true}%
11141     {%
11142       \GLSname[noindex,hyper=false]{#1}[]%
11143     }%
11144     {%
11145       \glSname[noindex,hyper=false]{#1}[]%
11146     }%
11147   }%
11148 }

```

glxstrtitlename Command to display name value in section title and table of contents.

```

11149 \newrobustcmd*{\glxstrtitlename}[1]{%
11150   \glSname[noindex,hyper=false]{#1}[]%
11151 }

```

\Glsxtrheadname First letter converted to upper case

```

11152 \newcommand*{\Glsxtrheadname}[1]{%
11153   \protect\NoCaseChange
11154   {%
11155     \gl@ifattribute{#1}{headuc}{true}%
11156     {%
11157       \GLSname[noindex,hyper=false]{#1}[]%
11158     }%
11159     {%
11160       \glSname[noindex,hyper=false]{#1}[]%
11161     }%
11162   }%
11163 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11164 %\changes{1.21}{2017-11-03}{new}
11165 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11166   \glSname[noindex,hyper=false]{#1}[]%
11167 }

```

\glxtrheadtext As above but for the text value.

```

11168 \newcommand*{\glxtrheadtext}[1]{%
11169   \protect\NoCaseChange
11170   {%
11171     \gl@ifattribute{#1}{headuc}{true}%
11172     {%
11173       \GLStext[noindex,hyper=false]{#1}[]%
11174     }%
11175     {%
11176       \glstext[noindex,hyper=false]{#1}[]%

```

```

11177 }%
11178 }%
11179 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

11180 \newrobustcmd*{\glsxtrtitletext}[1]{%
11181   \glstext[noindex,hyper=false]{#1}[]%
11182 }

```

\Glsxtrheadtext First letter converted to upper case

```

11183 \newcommand*{\Glsxtrheadtext}[1]{%
11184   \protect\NoCaseChange
11185   {%
11186     \glsifattribute{#1}{headuc}{true}%
11187     {%
11188       \GLStext[noindex,hyper=false]{#1}[]%
11189     }%
11190     {%
11191       \Glstext[noindex,hyper=false]{#1}[]%
11192     }%
11193   }%
11194 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

11195 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11196   \Glstext[noindex,hyper=false]{#1}[]%
11197 }

```

lsxtrheadplural As above but for the plural value.

```

11198 \newcommand*{\lsxtrheadplural}[1]{%
11199   \protect\NoCaseChange
11200   {%
11201     \glsifattribute{#1}{headuc}{true}%
11202     {%
11203       \GLSplural[noindex,hyper=false]{#1}[]%
11204     }%
11205     {%
11206       \glsplural[noindex,hyper=false]{#1}[]%
11207     }%
11208   }%
11209 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

11210 \newrobustcmd*{\lsxtrtitleplural}[1]{%
11211   \glsplural[noindex,hyper=false]{#1}[]%
11212 }

```

lsxtrheadplural Convert first letter to upper case.

```
11213 \newcommand*{\Glsxtrheadplural}[1]{%
11214   \protect\NoCaseChange
11215   {%
11216     \glsifattribute{#1}{headuc}{true}%
11217     {%
11218       \GLSplural[noindex,hyper=false]{#1}[]%
11219     }%
11220     {%
11221       \Glsplural[noindex,hyper=false]{#1}[]%
11222     }%
11223   }%
11224 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11225 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11226   \Glsplural[noindex,hyper=false]{#1}[]%
11227 }
```

glsxtrheadfirst As above but for the first value.

```
11228 \newcommand*{\glsxtrheadfirst}[1]{%
11229   \protect\NoCaseChange
11230   {%
11231     \glsifattribute{#1}{headuc}{true}%
11232     {%
11233       \GLSfirst[noindex,hyper=false]{#1}[]%
11234     }%
11235     {%
11236       \glsfirst[noindex,hyper=false]{#1}[]%
11237     }%
11238   }%
11239 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
11240 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
11241   \glsfirst[noindex,hyper=false]{#1}[]%
11242 }
```

Glsxtrheadfirst First letter converted to upper case

```
11243 \newcommand*{\Glsxtrheadfirst}[1]{%
11244   \protect\NoCaseChange
11245   {%
11246     \glsifattribute{#1}{headuc}{true}%
11247     {%
11248       \GLSfirst[noindex,hyper=false]{#1}[]%
11249     }%
11250     {%
```

```

11251      \Glsfirst[noindex,hyper=false]{#1}[]%
11252    }%
11253  }%
11254 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11255 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11256   \Glsfirst[noindex,hyper=false]{#1}[]%
11257 }

```

headfirstplural As above but for the firstplural value.

```

11258 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11259   \protect\NoCaseChange
11260   {%
11261     \glsifattribute{#1}{headuc}{true}%
11262     {%
11263       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11264     }%
11265     {%
11266       \glsfirstplural[noindex,hyper=false]{#1}[]%
11267     }%
11268   }%
11269 }

```

itlefirstplural Command to display firstplural value in section title and table of contents.

```

11270 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11271   \glsfirstplural[noindex,hyper=false]{#1}[]%
11272 }

```

headfirstplural First letter converted to upper case

```

11273 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11274   \protect\NoCaseChange
11275   {%
11276     \glsifattribute{#1}{headuc}{true}%
11277     {%
11278       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11279     }%
11280     {%
11281       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11282     }%
11283   }%
11284 }

```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11285 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11286   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11287 }

```

`\glsxtrheadlong` Command used to display long form in the page header.

```
11288 \newcommand*{\glsxtrheadlong}[1]{%
11289   \protect\NoCaseChange
11290   {%
11291     \glsifattribute{#1}{headuc}{true}%
11292     {%
11293       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11294     }%
11295     {%
11296       \glsxtrlong[noindex,hyper=false]{#1}[]%
11297     }%
11298   }%
11299 }
```

`glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
11300 \newrobustcmd*{\glsxtrtitlelong}[1]{%
11301   \glsxtrlong[noindex,hyper=false]{#1}[]%
11302 }
```

`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.

```
11303 \newcommand*{\glsxtrheadlongpl}[1]{%
11304   \protect\NoCaseChange
11305   {%
11306     \glsifattribute{#1}{headuc}{true}%
11307     {%
11308       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11309     }%
11310     {%
11311       \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11312     }%
11313   }%
11314 }
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
11315 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%
11316   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11317 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
11318 \newcommand*{\Glsxtrheadlong}[1]{%
11319   \protect\NoCaseChange
11320   {%
11321     \glsifattribute{#1}{headuc}{true}%
11322     {%
11323       \GLSxtrlong[noindex,hyper=false]{#1}[]%
```

```

11324 }%
11325 {%
11326     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11327 }%
11328 }%
11329 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11330 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11331     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11332 }

```

lgsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

11333 \newcommand*{\Glsxtrheadlongpl}[1]{%
11334     \protect\NoCaseChange
11335     {%
11336         \glsifattribute{#1}{headuc}{true}%
11337         {%
11338             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11339         }%
11340         {%
11341             \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11342         }%
11343     }%
11344 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11345 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11346     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11347 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

11348 \newcommand*{\glsxtrheadfull}[1]{%
11349     \protect\NoCaseChange
11350     {%
11351         \glsifattribute{#1}{headuc}{true}%
11352         {%
11353             \GLSxtrfull[noindex,hyper=false]{#1}[]%
11354         }%
11355         {%
11356             \glsxtrfull[noindex,hyper=false]{#1}[]%
11357         }%
11358     }%
11359 }

```


`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
11360 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11361   \glsxtrfull[noindex,hyper=false]{#1}[]%
11362 }
```

`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.

```
11363 \newcommand*{\glsxtrheadfullpl}[1]{%
11364   \protect\NoCaseChange
11365   {%
11366     \glsifattribute{#1}{headuc}{true}%
11367     {%
11368       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11369     }%
11370     {%
11371       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11372     }%
11373   }%
11374 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
11375 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
11376   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11377 }
```

`\GLsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
11378 \newcommand*{\GLsxtrheadfull}[1]{%
11379   \protect\NoCaseChange
11380   {%
11381     \glsifattribute{#1}{headuc}{true}%
11382     {%
11383       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11384     }%
11385     {%
11386       \GLsxtrfull[noindex,hyper=false]{#1}[]%
11387     }%
11388   }%
11389 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11390 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11391   \GLsxtrfull[noindex,hyper=false]{#1}[]%
11392 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11393 \newcommand*{\Glsxtrheadfullpl}[1]{%
11394   \protect\NoCaseChange
11395   {%
11396     \glsifattribute{#1}{headuc}{true}%
11397     {%
11398       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11399     }%
11400     {%
11401       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11402     }%
11403   }%
11404 }

```

`\sxttrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11405 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11406   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11407 }

```

`\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.

```

11408 \ifdef\texorpdfstring
11409 {
11410   \newcommand*{\glsfmtshort}[1]{%
11411     \texorpdfstring
11412       {\glsxtrtitleshort{#1}}%
11413       {\glsentryshort{#1}}%
11414   }
11415 }
11416 {
11417   \newcommand*{\glsfmtshort}[1]{%
11418     \glsxtrtitleshort{#1}}
11419 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11420 \ifdef\texorpdfstring
11421 {
11422   \newcommand*{\glsfmtshortpl}[1]{%
11423     \texorpdfstring
11424       {\glsxtrtitleshortpl{#1}}%
11425       {\glsentryshortpl{#1}}%
11426   }
11427 }
11428 {
11429   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11430 \glstrtitleshortpl{#1}}
11431 }

```

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).

```

11432 \ifdef\texorpdfstring
11433 {
11434 \newcommand*\Glsfmtshort}[1]{%
11435 \texorpdfstring
11436 {\Glsxtrtitleshort{#1}}%
11437 {\glsentryshort{#1}}%
11438 }
11439 }
11440 {
11441 \newcommand*\Glsfmtshort}[1]{%
11442 \Glsxtrtitleshort{#1}}
11443 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11444 \ifdef\texorpdfstring
11445 {
11446 \newcommand*\Glsfmtshortpl}[1]{%
11447 \texorpdfstring
11448 {\Glsxtrtitleshortpl{#1}}%
11449 {\glsentryshortpl{#1}}%
11450 }
11451 }
11452 {
11453 \newcommand*\Glsfmtshortpl}[1]{%
11454 \Glsxtrtitleshortpl{#1}}
11455 }

```

`\glsfmtname` As above but for the name value.

```

11456 \ifdef\texorpdfstring
11457 {
11458 \newcommand*\glsfmtname}[1]{%
11459 \texorpdfstring
11460 {\glsxtrtitlename{#1}}%
11461 {\glsentryname{#1}}%
11462 }
11463 }
11464 {
11465 \newcommand*\glsfmtname}[1]{%
11466 \glsxtrtitlename{#1}}
11467 }

```

`\Glsfmtname` First letter converted to upper case.

```

11468 \ifdef\texorpdfstring
11469 {
11470   \newcommand*{\Glsfmtname}[1]{%
11471     \texorpdfstring
11472     {\Glsxtrtitlename{#1}}%
11473     {\glsentryname{#1}}%
11474   }
11475 }
11476 {
11477   \newcommand*{\Glsfmtname}[1]{%
11478     \Glsxtrtitlename{#1}}
11479 }

```

`\glsfmttext` As above but for the text value.

```

11480 \ifdef\texorpdfstring
11481 {
11482   \newcommand*{\glsfmttext}[1]{%
11483     \texorpdfstring
11484     {\glsxtrtitletext{#1}}%
11485     {\glsentrytext{#1}}%
11486   }
11487 }
11488 {
11489   \newcommand*{\glsfmttext}[1]{%
11490     \glsxtrtitletext{#1}}
11491 }

```

`\Glsfmttext` First letter converted to upper case.

```

11492 \ifdef\texorpdfstring
11493 {
11494   \newcommand*{\Glsfmttext}[1]{%
11495     \texorpdfstring
11496     {\Glsxtrtitletext{#1}}%
11497     {\glsentrytext{#1}}%
11498   }
11499 }
11500 {
11501   \newcommand*{\Glsfmttext}[1]{%
11502     \Glsxtrtitletext{#1}}
11503 }

```

`\glsfmtplural` As above but for the plural value.

```

11504 \ifdef\texorpdfstring
11505 {
11506   \newcommand*{\glsfmtplural}[1]{%
11507     \texorpdfstring
11508     {\glsxtrtitleplural{#1}}%
11509     {\glsentryplural{#1}}%
11510   }

```

```

11511 }
11512 {
11513   \newcommand*{\glsfmtplural}[1]{%
11514     \glsxtrtitleplural{#1}}
11515 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11516 \ifdef\texorpdfstring
11517 {
11518   \newcommand*{\Glsfmtplural}[1]{%
11519     \texorpdfstring
11520     {\Glsxtrtitleplural{#1}}%
11521     {\glsentryplural{#1}}%
11522   }
11523 }
11524 {
11525   \newcommand*{\Glsfmtplural}[1]{%
11526     \Glsxtrtitleplural{#1}}
11527 }

```

`\glsfmtfirst` As above but for the first value.

```

11528 \ifdef\texorpdfstring
11529 {
11530   \newcommand*{\glsfmtfirst}[1]{%
11531     \texorpdfstring
11532     {\glsxtrtitlefirst{#1}}%
11533     {\glsentryfirst{#1}}%
11534   }
11535 }
11536 {
11537   \newcommand*{\glsfmtfirst}[1]{%
11538     \glsxtrtitlefirst{#1}}
11539 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11540 \ifdef\texorpdfstring
11541 {
11542   \newcommand*{\Glsfmtfirst}[1]{%
11543     \texorpdfstring
11544     {\Glsxtrtitlefirst{#1}}%
11545     {\glsentryfirst{#1}}%
11546   }
11547 }
11548 {
11549   \newcommand*{\Glsfmtfirst}[1]{%
11550     \Glsxtrtitlefirst{#1}}
11551 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11552 \ifdef\teorpdfstring
11553 {
11554   \newcommand*\glsfmtfirstpl}[1]{%
11555     \teorpdfstring
11556     {\glstrtitlefirstplural{#1}}%
11557     {\glstryfirstplural{#1}}%
11558   }
11559 }
11560 {
11561   \newcommand*\glsfmtfirstpl}[1]{%
11562     \glstrtitlefirstplural{#1}}
11563 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11564 \ifdef\teorpdfstring
11565 {
11566   \newcommand*\Glsfmtfirstpl}[1]{%
11567     \teorpdfstring
11568     {\Glsxtrtitlefirstplural{#1}}%
11569     {\glstryfirstplural{#1}}%
11570   }
11571 }
11572 {
11573   \newcommand*\Glsfmtfirstpl}[1]{%
11574     \Glsxtrtitlefirstplural{#1}}
11575 }

```

`\glsfmtlong` As above but for the long value.

```

11576 \ifdef\teorpdfstring
11577 {
11578   \newcommand*\glsfmtlong}[1]{%
11579     \teorpdfstring
11580     {\glstrtitlelong{#1}}%
11581     {\glstrylong{#1}}%
11582   }
11583 }
11584 {
11585   \newcommand*\glsfmtlong}[1]{%
11586     \glstrtitlelong{#1}}
11587 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11588 \ifdef\teorpdfstring
11589 {
11590   \newcommand*\Glsfmtlong}[1]{%
11591     \teorpdfstring
11592     {\Glsxtrtitlelong{#1}}%
11593     {\glstrylong{#1}}%
11594   }

```

```

11595 }
11596 {
11597   \newcommand*{\Glsfmtlong}[1]{%
11598     \Glsxtrtitlelong{#1}}
11599 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11600 \ifdef\texorpdfstring
11601 {
11602   \newcommand*{\glsfmtlongpl}[1]{%
11603     \texorpdfstring
11604     {\Glsxtrtitlelongpl{#1}}%
11605     {\Glsentrylongpl{#1}}%
11606   }
11607 }
11608 {
11609   \newcommand*{\glsfmtlongpl}[1]{%
11610     \Glsxtrtitlelongpl{#1}}
11611 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11612 \ifdef\texorpdfstring
11613 {
11614   \newcommand*{\Glsfmtlongpl}[1]{%
11615     \texorpdfstring
11616     {\Glsxtrtitlelongpl{#1}}%
11617     {\Glsentrylongpl{#1}}%
11618   }
11619 }
11620 {
11621   \newcommand*{\Glsfmtlongpl}[1]{%
11622     \Glsxtrtitlelongpl{#1}}
11623 }

```

`\glsfmtfull` In-line full format.

```

11624 \ifdef\texorpdfstring
11625 {
11626   \newcommand*{\glsfmtfull}[1]{%
11627     \texorpdfstring
11628     {\Glsxtrtitlefull{#1}}%
11629     {\Glsxtrinlinefullformat{#1}{}}%
11630   }
11631 }
11632 {
11633   \newcommand*{\glsfmtfull}[1]{%
11634     \Glsxtrtitlefull{#1}}
11635 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11636 \ifdef\txorpdfstring
11637 {
11638   \newcommand*{\Glsfmtfull}[1]{%
11639     \txorpdfstring
11640     {\Glsxtrtitlefull{#1}}%
11641     {\Glsxtrinlinefullformat{#1}{}}}%
11642   }
11643 }
11644 {
11645   \newcommand*{\Glsfmtfull}[1]{%
11646     \Glsxtrtitlefull{#1}}
11647 }

```

`\glsfmtfullpl` In-line full plural format.

```

11648 \ifdef\txorpdfstring
11649 {
11650   \newcommand*{\glsfmtfullpl}[1]{%
11651     \txorpdfstring
11652     {\glsxtrtitlefullpl{#1}}%
11653     {\glsxtrinlinefullplformat{#1}{}}}%
11654   }
11655 }
11656 {
11657   \newcommand*{\glsfmtfullpl}[1]{%
11658     \glsxtrtitlefullpl{#1}}
11659 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11660 \ifdef\txorpdfstring
11661 {
11662   \newcommand*{\Glsfmtfullpl}[1]{%
11663     \txorpdfstring
11664     {\Glsxtrtitlefullpl{#1}}%
11665     {\Glsxtrinlinefullplformat{#1}{}}}%
11666   }
11667 }
11668 {
11669   \newcommand*{\Glsfmtfullpl}[1]{%
11670     \Glsxtrtitlefullpl{#1}}
11671 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11672 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```



```

11673 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11674 }

```

sariesExtraLang

```

11675 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11676   \ProvidesFile{glossariesxtr-#1.ldf}%
11677 }

```

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.

```

11678 \newcommand{\glxtr@loaddialect}{%
11679   \IfTrackedLanguageFileExists{\this@dialect}%
11680   {glossariesxtr-}% prefix
11681   {.ldf}%
11682   {%
11683     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11684   }%
11685   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11686 \@glxtrdialecthook
11687 }

```

```

11688 \ifpackageloaded{tracklang}
11689 {%
11690   \AnyTrackedLanguages
11691   {%
11692     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11693   }%
11694   {}%
11695 }
11696 {}

```

Load `glossaries-extra-stylemods` if required.

```

11697 \@glxtr@redefstyles

```

and set the style:

```

11698 \@glxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11699 \NeedsTeXFormat{LaTeX2e}
11700 \ProvidesPackage{glossaries-extra-bib2gls}[2018/07/26 v1.33 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11701 \newcommand*{\glshex}{\string\u}

```

`\glscapturedgroup`

```

11702 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```

11703 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11704   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11705 }

```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11706 \newcommand*{\GlsXtrProvideCommand}{\providecommand}

```

`\GlsXtrWrglossaryLocation` For use with indexcounter and bib2gls.

```

11707 \newcommand*{\GlsXtrWrglossaryLocation}[2]{#1}

```

`\GlsXtrIndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with indexcounter and bib2gls.

```

11708 \ifdef\hyperref
11709 {%
11710   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
11711     \GlsXtrIfHasField{indexcounter}{#2}%
11712     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11713     {#1}%
11714   }
11715 }
11716 {
11717   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
11718 }

```

`\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.

```

11719 \newcommand*{\GlsXtrDualField}{dual}

```

sXtrDualBackLink

<code>\GlsXtrDualBackLink{<text>}{<label>}</code>

Adds a hyperlink to the dual entry.

```
11720 \newcommand*{\GlsXtrDualBackLink}[2]{%
11721   \glstrifhasfield{\GlsXtrDualField}{#2}%
11722   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11723   {#2}%
11724 }
```

TeXEntryAliases

Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
11725 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
11726   article=bibtexentry,
11727   book=bibtexentry,
11728   booklet=bibtexentry,
11729   conference=bibtexentry,
11730   inbook=bibtexentry,
11731   incollection=bibtexentry,
11732   inproceedings=bibtexentry,
11733   manual=bibtexentry,
11734   mastersthesis=bibtexentry,
11735   misc=bibtexentry,
11736   phdthesis=bibtexentry,
11737   proceedings=bibtexentry,
11738   techreport=bibtexentry,
11739   unpublished=bibtexentry
11740 }
```

ideBibTeXFields

Convenient shortcut to define the standard BibTeX fields.

```
11741 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11742   \glsaddstoragekey{address}{\glstrbibaddress}%
11743   \glsaddstoragekey{author}{\glstrbibauthor}%
11744   \glsaddstoragekey{booktitle}{\glstrbibbooktitle}%
11745   \glsaddstoragekey{chapter}{\glstrbibchapter}%
11746   \glsaddstoragekey{edition}{\glstrbibedition}%
11747   \glsaddstoragekey{howpublished}{\glstrbibhowpublished}%
11748   \glsaddstoragekey{institution}{\glstrbibinstitution}%
11749   \glsaddstoragekey{journal}{\glstrbibjournal}%
11750   \glsaddstoragekey{month}{\glstrbibmonth}%
11751   \glsaddstoragekey{note}{\glstrbibnote}%
11752   \glsaddstoragekey{number}{\glstrbibnumber}%
11753   \glsaddstoragekey{organization}{\glstrbiborganization}%
11754   \glsaddstoragekey{pages}{\glstrbibpages}%
11755   \glsaddstoragekey{publisher}{\glstrbibpublisher}%
11756   \glsaddstoragekey{school}{\glstrbibschooll}%
11757   \glsaddstoragekey{series}{\glstrbibseries}%
11758   \glsaddstoragekey{title}{\glstrbibtitle}%

```

```

11759 \glsaddstoragekey{bibtextype}{-}{\glsxtrbibtype}%
11760 \glsaddstoragekey{volume}{-}{\glsxtrbibvolume}%
11761 }

```

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

```
11762 \providecommand*{\Alpha}{\mathrm{A}}
```

\backslash Beta

```
11763 \providecommand*{\Beta}{\mathrm{B}}
```

\backslash Epsilon

```
11764 \providecommand*{\Epsilon}{\mathrm{E}}
```

\backslash Zeta

```
11765 \providecommand*{\Zeta}{\mathrm{Z}}
```

\backslash Eta

```
11766 \providecommand*{\Eta}{\mathrm{H}}
```

\backslash Iota

```
11767 \providecommand*{\Iota}{\mathrm{I}}
```

\backslash Kappa

```
11768 \providecommand*{\Kappa}{\mathrm{K}}
```

\backslash Mu

```
11769 \providecommand*{\Mu}{\mathrm{M}}
```

\backslash Nu

```
11770 \providecommand*{\Nu}{\mathrm{N}}
```

\backslash Omicron

```
11771 \providecommand*{\Omicron}{\mathrm{O}}
```

\backslash Rho

```
11772 \providecommand*{\Rho}{\mathrm{P}}
```

\backslash Tau

```
11773 \providecommand*{\Tau}{\mathrm{T}}
```

```

\Chi
11774 \providecommand*\Chi{\mathrm{X}}

\Digamma
11775 \providecommand*\Digamma{\mathrm{F}}

\omicron
11776 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11777 \@ifpackageloaded{upgreek}%
11778 {

\Upalpha
11779   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11780   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11781   \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11782   \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11783   \providecommand*\Upeta{\mathrm{H}}

\Upiota
11784   \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11785   \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11786   \providecommand*\Upmu{\mathrm{M}}

\Upnu
11787   \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11788   \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11789   \providecommand*\Uprho{\mathrm{P}}

```

`\Uptau`

```
11790 \providecommand*\Uptau{\mathrm{T}}
```

`\Upchi`

```
11791 \providecommand*\Upchi{\mathrm{X}}
```

`\upomicron`

```
11792 \providecommand*\upomicron{\mathrm{o}}
```

```
11793 }%
```

```
11794 {}% 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={\glxstrcontrolrules
; \glxstrspacerules
; \glxstrnonprintablerules
; \glxstrcombiningdiacriticrules
, \glxstrhyphenrules
< \glxstrgeneralpuncrules
< \glxstrdigitrules
< \glxstrfractionrules
< \glxstrGeneralLatinIVrules
< \glxstrMathItalicGreekIrules
}
```

`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.

```
11795 \newcommand*\glxstrcontrolrules{%
```

```
11796 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
```

```
11797 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
```

```
11798 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
```

```
11799 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
```

```
11800 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
```

```
11801 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
```

```

11802 \string'\string=\glshex 0011
11803 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11804 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11805 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11806 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11807 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11808 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11809 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11810 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11811 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11812 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11813 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11814 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11815 }

```

lsxtrspacerules These are space characters.

```

11816 \newcommand*{\glsxtrspacerules}{%
11817 \string' \string'\string;
11818 \string'\glshex 00A0\string'\string;
11819 \string'\glshex 2000\string'\string;
11820 \string'\glshex 2001\string'\string;
11821 \string'\glshex 2002\string'\string;
11822 \string'\glshex 2003\string'\string;
11823 \string'\glshex 2004\string'\string;
11824 \string'\glshex 2005\string'\string;
11825 \string'\glshex 2006\string'\string;
11826 \string'\glshex 2007\string'\string;
11827 \string'\glshex 2008\string'\string;
11828 \string'\glshex 2009\string'\string;
11829 \string'\glshex 200A\string'\string;
11830 \string'\glshex 3000\string'
11831 }

```

nonprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11832 \newcommand*{\glsxtrnonprintablerules}{%
11833 \string'\glshex FEFF\string'\string;
11834 \string'\glshex 000A\string'\string;
11835 \string'\glshex 0009\string'\string;
11836 \string'\glshex 000C\string'\string;
11837 \string'\glshex 000B\string'
11838 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

11839 \newcommand*{\glsxtrcombiningdiacriticrules}{%
11840 \glsxtrcombiningdiacriticIrules\string;
11841 \glsxtrcombiningdiacriticIIrules\string;
11842 \glsxtrcombiningdiacriticIIIrules\string;
11843 \glsxtrcombiningdiacriticIVrules
11844 }

```

diacriticIrules First set of combining diacritic marks.

```
11845 \newcommand*{\glxtrcombingdiacriticIrules}{%
11846 \glshex 0301\string;% combining acute
11847 \glshex 0300\string;% combining grave
11848 \glshex 0306\string;% combining breve
11849 \glshex 0302\string;% combining circumflex
11850 \glshex 030C\string;% combining caron
11851 \glshex 030A\string;% combining ring
11852 \glshex 030D\string;% combining vertical line above
11853 \glshex 0308\string;% combining diaeresis
11854 \glshex 030B\string;% combining double acute
11855 \glshex 0303\string;% combining tilde
11856 \glshex 0307\string;% combining dot above
11857 \glshex 0304% combining macron
11858 }
```

diacriticIIrules Second set of combining diacritic marks.

```
11859 \newcommand*{\glxtrcombingdiacriticIIrules}{%
11860 \glshex 0337\string;% combining short solidus overlay
11861 \glshex 0327\string;% combining cedilla
11862 \glshex 0328\string;% combining ogonek
11863 \glshex 0323\string;% combining dot below
11864 \glshex 0332\string;% combining low line
11865 \glshex 0305\string;% combining overline
11866 \glshex 0309\string;% combining hook above
11867 \glshex 030E\string;% combining double vertical line above
11868 \glshex 030F\string;% combining double grave accent
11869 \glshex 0310\string;% combining candrabindu
11870 \glshex 0311\string;% combining inverted breve
11871 \glshex 0312\string;% combining turned comma above
11872 \glshex 0313\string;% combining comma above
11873 \glshex 0314\string;% combining reversed comma above
11874 \glshex 0315\string;% combining comma above right
11875 \glshex 0316\string;% combining grave accent below
11876 \glshex 0317% combining acute accent below
11877 }
```

diacriticIIIrules Third set of combining diacritic marks.

```
11878 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
11879 \glshex 0318\string;% combining left tack below
11880 \glshex 0319\string;% combining right tack below
11881 \glshex 031A\string;% combining left angle above
11882 \glshex 031B\string;% combining horn
11883 \glshex 031C\string;% combining left half ring below
11884 \glshex 031D\string;% combining up tack below
11885 \glshex 031E\string;% combining down tack below
11886 \glshex 031F\string;% combining plus sign below
11887 \glshex 0320\string;% combining minus sign below
11888 \glshex 0321\string;% combining palatalized hook below
```


11889 \glshex 0322\string;% combining retroflex hook below
 11890 \glshex 0324\string;% combining diaresis below
 11891 \glshex 0325\string;% combining ring below
 11892 \glshex 0326\string;% combining comma below
 11893 \glshex 0329\string;% combining vertical line below
 11894 \glshex 032A\string;% combining bridge below
 11895 \glshex 032B\string;% combining inverted double arch below
 11896 \glshex 032C\string;% combining caron below
 11897 \glshex 032D\string;% combining circumflex accent below
 11898 \glshex 032E\string;% combining breve below
 11899 \glshex 032F\string;% combining inverted breve below
 11900 \glshex 0330\string;% combining tilde below
 11901 \glshex 0331\string;% combining macron below
 11902 \glshex 0333\string;% combining double low line
 11903 \glshex 0334\string;% combining tilde overlay
 11904 \glshex 0335\string;% combining short stroke overlay
 11905 \glshex 0336\string;% combining long stroke overlay
 11906 \glshex 0338\string;% combining long solidus overlay
 11907 \glshex 0339\string;% combining combining right half ring below
 11908 \glshex 033A\string;% combining inverted bridge below
 11909 \glshex 033B\string;% combining square below
 11910 \glshex 033C\string;% combining seagull below
 11911 \glshex 033D\string;% combining x above
 11912 \glshex 033E\string;% combining vertical tilde
 11913 \glshex 033F\string;% combining double overline
 11914 \glshex 0342\string;% combining Greek perispomeni
 11915 \glshex 0344\string;% combining Greek dialytika tonos
 11916 \glshex 0345\string;% combining Greek ypogegrammeni
 11917 \glshex 0360\string;% combining double tilde
 11918 \glshex 0361\string;% combining double inverted breve
 11919 \glshex 0483\string;% combining Cyrillic titlo
 11920 \glshex 0484\string;% combining Cyrillic palatalization
 11921 \glshex 0485\string;% combining Cyrillic dasia pneumata
 11922 \glshex 0486% combining Cyrillic psili pneumata
 11923 }

iacriticIVrules Fourth set of combining diacritic marks.

11924 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
 11925 \glshex 20D0\string;% combining left harpoon above
 11926 \glshex 20D1\string;% combining right harpoon above
 11927 \glshex 20D2\string;% combining long vertical line overlay
 11928 \glshex 20D3\string;% combining short vertical line overlay
 11929 \glshex 20D4\string;% combining anticlockwise arrow above
 11930 \glshex 20D5\string;% combining clockwise arrow above
 11931 \glshex 20D6\string;% combining left arrow above
 11932 \glshex 20D7\string;% combining right arrow above
 11933 \glshex 20D8\string;% combining ring overlay
 11934 \glshex 20D9\string;% combining clockwise ring overlay
 11935 \glshex 20DA\string;% combining anticlockwise ring overlay

```

11936 \glshex 20DB\string;% combining three dots above
11937 \glshex 20DC\string;% combining four dots above
11938 \glshex 20DD\string;% combining enclosing circle
11939 \glshex 20DE\string;% combining enclosing square
11940 \glshex 20DF\string;% combining enclosing diamond
11941 \glshex 20E0\string;% combining enclosing circle backslash
11942 \glshex 20E1% combining left right arrow above
11943 }

```

sxtrhyphenrules Hyphens.

```

11944 \newcommand*{\glsxtrhyphenrules}{%
11945 \string'\string-\string'\string;% ASCII hyphen
11946 \glshex 00AD\string;% soft hyphen
11947 \glshex 2010\string;% hyphen
11948 \glshex 2011\string;% non-breaking hyphen
11949 \glshex 2012\string;% figure dash
11950 \glshex 2013\string;% en dash
11951 \glshex 2014\string;% em dash
11952 \glshex 2015\string;% horizontal bar
11953 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11954 }

```

eneralpuncrules General punctuation.

```

11955 \newcommand*{\glsxtrgeneralpuncrules}{%
11956 \glsxtrgeneralpuncIrules
11957 \string<\glsxtrcurrencyrules
11958 \string<\glsxtrgeneralpuncIIrules
11959 }

```

neralpuncIrules First set of general punctuation.

```

11960 \newcommand*{\glsxtrgeneralpuncIrules}{%
11961 \string'\glshex 005F\string'% underscore
11962 \string<\glshex 00AF% macron
11963 \string<\string'\glshex 002C\string'% comma
11964 \string<\string'\glshex 003B\string'% semi-colon
11965 \string<\string'\glshex 003A\string'% colon
11966 \string<\string'\glshex 0021\string'% exclamation mark
11967 \string<\glshex 00A1% inverted exclamation mark
11968 \string<\string'\glshex 003F\string'% question mark
11969 \string<\glshex 00BF% inverted question mark
11970 \string<\string'\glshex 002F\string'% solidus
11971 \string<\string'\glshex 002E\string'% full stop
11972 \string<\glshex 00B4% acute accent
11973 \string<\string'\glshex 0060\string'% grave accent
11974 \string<\string'\glshex 005E\string'% circumflex accent
11975 \string<\glshex 00A8% diaeresis
11976 \string<\string'\glshex 007E\string'% tilde
11977 \string<\glshex 00B7% middle dot
11978 \string<\glshex 00B8% cedilla

```

```

11979 \string<\string'\glshex 0027\string'% straight apostrophe
11980 \string<\string'\glshex 0022\string'% straight double quote
11981 \string<\glshex 00AB% left guillemet
11982 \string<\glshex 00BB% right guillemet
11983 \string<\string'\glshex 0028\string'% left parenthesis
11984 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
11985 \string<\string'\glshex 0029\string'% right parenthesis
11986 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
11987 \string<\string'\glshex 005B\string'% left square bracket
11988 \string<\string'\glshex 005D\string'% right square bracket
11989 \string<\string'\glshex 007B\string'% left curly bracket
11990 \string<\string'\glshex 007D\string'% right curly bracket
11991 \string<\glshex 00A7% section sign
11992 \string<\glshex 00B6% pilcrow sign
11993 \string<\glshex 00A9% copyright sign
11994 \string<\glshex 00AE% registered sign
11995 \string<\string'\glshex 0040\string'% at sign
11996 }

```

trcurrencyrules General punctuation.

```

11997 \newcommand*{\glxtrcurrencyrules}{%
11998 \glshex 00A4% currency sign
11999 \string<\glshex 0E3F% Thai currency symbol baht
12000 \string<\glshex 00A2% cent sign
12001 \string<\glshex 20A1% colon sign
12002 \string<\glshex 20A2% cruzeiro sign
12003 \string<\string'\glshex 0024\string'% dollar sign
12004 \string<\glshex 20AB% dong sign
12005 \string<\glshex 20AC% euro sign
12006 \string<\glshex 20A3% French franc sign
12007 \string<\glshex 20A4% lira sign
12008 \string<\glshex 20A5% mill sign
12009 \string<\glshex 20A6% naira sign
12010 \string<\glshex 20A7% peseta sign
12011 \string<\glshex 00A3% pound sign
12012 \string<\glshex 20A8% rupee sign
12013 \string<\glshex 20AA% new sheqel sign
12014 \string<\glshex 20A9% won sign
12015 \string<\glshex 00A5% yen sign
12016 }

```

eralpuncIIrules Second set of general punctuation.

```

12017 \newcommand*{\glxtrgeneralpuncIIrules}{%
12018 \string'\glshex 002A\string'% asterisk
12019 \string<\string'\glshex 005C\string'% backslash
12020 \string<\string'\glshex 0026\string'% ampersand
12021 \string<\string'\glshex 0023\string'% hash sign
12022 \string<\string'\glshex 0025\string'% percent sign
12023 \string<\string'\glshex 002B\string'% plus sign

```

```

12024 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12025 \string<\glshex 00B1% plus-minus sign
12026 \string<\glshex 00F7% division sign
12027 \string<\glshex 00D7% multiplication sign
12028 \string<\string'\glshex 003C\string'% less-than sign
12029 \string<\string'\glshex 003D\string'% equals sign
12030 \string<\string'\glshex 003E\string'% greater-than sign
12031 \string<\glshex 00AC% not sign
12032 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12033 \string<\glshex 00A6% broken bar
12034 \string<\glshex 00B0% degree sign
12035 \string<\glshex 00B5% micron sign
12036 }

```

eralLatinIrules Basic Latin alphabet.

```

12037 \newcommand*{\glxtrGeneralLatinIrules}{%
12038 \glxtrLatinA
12039 \string<b,B%
12040 \string<c,C%
12041 \string<d,D%
12042 \string<\glxtrLatinE
12043 \string<f,F%
12044 \string<g,G%
12045 \string<\glxtrLatinH
12046 \string<\glxtrLatinI
12047 \string<j,J%
12048 \string<\glxtrLatinK
12049 \string<\glxtrLatinL
12050 \string<\glxtrLatinM
12051 \string<\glxtrLatinN
12052 \string<\glxtrLatinO
12053 \string<\glxtrLatinP
12054 \string<q,Q%
12055 \string<r,R%
12056 \string<\glxtrLatinS
12057 \string<\glxtrLatinT
12058 \string<u,U%
12059 \string<v,V%
12060 \string<w,W%
12061 \string<\glxtrLatinX
12062 \string<y,Y%
12063 \string<z,Z
12064 }

```

eralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12065 \newcommand*{\glxtrGeneralLatinIIrules}{%
12066 \glxtrLatinA
12067 \string<b,B%
12068 \string<c,C%

```

```

12069 \string<d,D%
12070 \string<\glxtrLatinEth
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& SS \string, \glxtrLatinEszettSs
12087 \string<\glxtrLatinT
12088 \string<u,U%
12089 \string<v,V%
12090 \string<w,W%
12091 \string<\glxtrLatinX
12092 \string<y,Y%
12093 \string<z,Z%
12094 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12095 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12096 \glxtrLatinA
12097 \string<b,B%
12098 \string<c,C%
12099 \string<d,D%
12100 \string<\glxtrLatinEth
12101 \string<\glxtrLatinE
12102 \string<f,F%
12103 \string<g,G%
12104 \string<\glxtrLatinH
12105 \string<\glxtrLatinI
12106 \string<j,J%
12107 \string<\glxtrLatinK
12108 \string<\glxtrLatinL
12109 \string<\glxtrLatinM
12110 \string<\glxtrLatinN
12111 \string<\glxtrLatinO
12112 \string<\glxtrLatinP
12113 \string<q,Q%
12114 \string<r,R%
12115 \string<\glxtrLatinS

```

```

12116 \string& SZ, \glxtrLatinEszettSz
12117 \string<\glxtrLatinT
12118 \string<u,U%
12119 \string<v,V%
12120 \string<w,W%
12121 \string<\glxtrLatinX
12122 \string<y,Y%
12123 \string<z,Z%
12124 }

```

GeneralLatinIVrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12125 \newcommand*{\glxtrGeneralLatinIVrules}{%
12126 \glxtrLatinA
12127 \string& AE , \glxtrLatinAELigature
12128 \string<b,B%
12129 \string<c,C%
12130 \string<d,D%
12131 \string<\glxtrLatinEth
12132 \string<\glxtrLatinE
12133 \string<f,F%
12134 \string<g,G%
12135 \string<\glxtrLatinH
12136 \string<\glxtrLatinI
12137 \string<j,J%
12138 \string<\glxtrLatinK
12139 \string<\glxtrLatinL
12140 \string<\glxtrLatinM
12141 \string<\glxtrLatinN
12142 \string<\glxtrLatinO
12143 \string& OE , \glxtrLatinOELigature
12144 \string<\glxtrLatinP
12145 \string<q,Q%
12146 \string<r,R%
12147 \string<\glxtrLatinS
12148 \string& SS , \glxtrLatinEszettSs
12149 \string<\glxtrLatinT
12150 \string& th =\glshex 00DE
12151 \string& TH =\glshex 00FE
12152 \string<u,U%
12153 \string<v,V%
12154 \string<w,W%
12155 \string<\glxtrLatinX
12156 \string<y,Y%
12157 \string<z,Z%
12158 }

```

GeneralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12159 \newcommand*{\glxtrGeneralLatinVrules}{%

```

```

12160 \glxtrLatinA
12161 \string<b,B%
12162 \string<c,C%
12163 \string<d,D%
12164 \string<\glxtrLatinEth
12165 \string<\glxtrLatinE
12166 \string<f,F%
12167 \string<g,G%
12168 \string<\glxtrLatinH
12169 \string<\glxtrLatinI
12170 \string<j,J%
12171 \string<\glxtrLatinK
12172 \string<\glxtrLatinL
12173 \string<\glxtrLatinM
12174 \string<\glxtrLatinN
12175 \string<\glxtrLatinO
12176 \string<\glxtrLatinP
12177 \string<q,Q%
12178 \string<r,R%
12179 \string<\glxtrLatinS
12180 \string& SS , \glxtrLatinEszettSs
12181 \string<\glxtrLatinT
12182 \string& th =\glshex 00DE
12183 \string& TH =\glshex 00FE
12184 \string<u,U%
12185 \string<v,V%
12186 \string<w,W%
12187 \string<\glxtrLatinX
12188 \string<y,Y%
12189 \string<z,Z%
12190 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12191 \newcommand*{\glxtrGeneralLatinVIrules}{%
12192 \glxtrLatinA
12193 \string<b,B%
12194 \string<c,C%
12195 \string<d,D%
12196 \string<\glxtrLatinEth
12197 \string<\glxtrLatinE
12198 \string<f,F%
12199 \string<g,G%
12200 \string<\glxtrLatinH
12201 \string<\glxtrLatinI
12202 \string<j,J%
12203 \string<\glxtrLatinK
12204 \string<\glxtrLatinL
12205 \string<\glxtrLatinM
12206 \string<\glxtrLatinN

```

```

12207 \string<\glxtrLatinO
12208 \string<\glxtrLatinP
12209 \string<q,Q%
12210 \string<r,R%
12211 \string<\glxtrLatinS
12212 \string& SZ , \glxtrLatinEszettSz
12213 \string<\glxtrLatinT
12214 \string& th =\glshex 00DE
12215 \string& TH =\glshex 00FE
12216 \string<u,U%
12217 \string<v,V%
12218 \string<w,W%
12219 \string<\glxtrLatinX
12220 \string<y,Y%
12221 \string<z,Z%
12222 }

```

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).

```

12223 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12224 \glxtrLatinA
12225 \string<\glxtrLatinAELigature
12226 \string<b,B%
12227 \string<c,C%
12228 \string<d,D%
12229 \string<\glxtrLatinEth
12230 \string<\glxtrLatinE
12231 \string<f,F%
12232 \string<\glxtrLatinInsularG
12233 \string<\glxtrLatinH
12234 \string<\glxtrLatinI
12235 \string<j,J%
12236 \string<\glxtrLatinK
12237 \string<\glxtrLatinL
12238 \string<\glxtrLatinM
12239 \string<\glxtrLatinN
12240 \string<\glxtrLatinO
12241 \string<\glxtrLatinOELigature
12242 \string<\glxtrLatinP
12243 \string<q,Q%
12244 \string<r,R%
12245 \string<\glshex 017F=\glxtrLatinS % s and long s
12246 \string<\glxtrLatinT
12247 \string<\glxtrLatinThorn
12248 \string<u,U%
12249 \string<v,V%
12250 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12251 \string<\glxtrLatinX
12252 \string<y,Y%

```



```

12253 \string<z,Z%
12254 }

```

LatinVIIIrules 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).

```

12255 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
12256 \glxtrLatinA
12257 \string& AE , \glxtrLatinAELigature
12258 \string<b,B%
12259 \string<c,C%
12260 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12261 \string<\glxtrLatinE
12262 \string<f,F%
12263 \string<g,G%
12264 \string<\glxtrLatinH
12265 \string<\glxtrLatinI
12266 \string<j,J%
12267 \string<\glxtrLatinK
12268 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12269 \string<\glxtrLatinM
12270 \string<\glxtrLatinN
12271 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12272 \string& OE , \glxtrLatinOELigature
12273 \string<\glxtrLatinP
12274 \string<q,Q%
12275 \string<r,R%
12276 \string<\glxtrLatinS
12277 \string& SS , \glxtrLatinEszettSs
12278 \string<\glxtrLatinT
12279 \string& th =\glshex 00DE
12280 \string& TH =\glshex 00FE
12281 \string<u,U%
12282 \string<v,V%
12283 \string<w,W%
12284 \string<\glxtrLatinX
12285 \string<y,Y%
12286 \string<z,Z%
12287 }

```

\glxtrLatinA

```

12288 \newcommand*{\glxtrLatinA}{%
12289 a\string=\glshex 00AA\string=\glshex 2090,A
12290 }

```

\glxtrLatinE

```

12291 \newcommand*{\glxtrLatinE}{%
12292 e\string=\glshex 2091,E
12293 }

```

\glsxtrLatinH

```
12294 \newcommand*{\glsxtrLatinH}{%  
12295   h\string=\glshex 2095,H  
12296 }
```

\glsxtrLatinI

```
12297 \newcommand*{\glsxtrLatinI}{%  
12298   i\string=\glshex 2071,I  
12299 }
```

\glsxtrLatinK

```
12300 \newcommand*{\glsxtrLatinK}{%  
12301   k\string=\glshex 2096,K  
12302 }
```

\glsxtrLatinL

```
12303 \newcommand*{\glsxtrLatinL}{%  
12304   l\string=\glshex 2097,L  
12305 }
```

\glsxtrLatinM

```
12306 \newcommand*{\glsxtrLatinM}{%  
12307   m\string=\glshex 2098,M  
12308 }
```

\glsxtrLatinN

```
12309 \newcommand*{\glsxtrLatinN}{%  
12310   n\string=\glshex 207F\string=\glshex 2099,N  
12311 }
```

\glsxtrLatinO

```
12312 \newcommand*{\glsxtrLatinO}{%  
12313   o\string=\glshex 00BA\string=\glshex 2092,O  
12314 }
```

\glsxtrLatinP

```
12315 \newcommand*{\glsxtrLatinP}{%  
12316   p\string=\glshex 209A,P  
12317 }
```

\glsxtrLatinS

```
12318 \newcommand*{\glsxtrLatinS}{%  
12319   s\string=\glshex 209B,S  
12320 }
```

\glsxtrLatinT

```
12321 \newcommand*{\glsxtrLatinT}{%  
12322   t\string=\glshex 209C,T  
12323 }
```

```

\glxtrLatinX
12324 \newcommand*{\glxtrLatinX}{%
12325   x\string=\glshex 2093,X
12326 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
12327 \newcommand*{\glxtrLatinSchwa}{%
12328   \glshex 0259\string=\glshex 2094,\glshex 018F
12329 }

trLatinEszettSs
12330 \newcommand*{\glxtrLatinEszettSs}{%
12331   \glshex 00DF% eszett
12332   \string=\glshex 017Fs % long S s
12333 }

trLatinEszettSz
12334 \newcommand*{\glxtrLatinEszettSz}{%
12335   \glshex 00DF% eszett
12336   \string= \glshex 017Fz % long S z
12337 }

\glxtrLatinEth
12338 \newcommand*{\glxtrLatinEth}{%
12339   \glshex 00F0,\glshex 00D0% eth
12340 }

lsxtrLatinThorn
12341 \newcommand*{\glxtrLatinThorn}{%
12342   \glshex 00FE,\glshex 00DE% thorn
12343 }

LatinAELigature
12344 \newcommand*{\glxtrLatinAELigature}{%
12345   \glshex 00E6,\glshex 00C6% AE-ligature
12346 }

LatinOELigature
12347 \newcommand*{\glxtrLatinOELigature}{%
12348   \glshex 0153,\glshex 0152% OE-ligature
12349 }

\glxtrLatinAA
12350 \newcommand*{\glxtrLatinAA}{%
12351   \glshex 00E5=a\glshex 030A,% \aa
12352   \glshex 00C5=A\glshex 030A% \AA
12353 }

```

glsxtrLatinWynn

```
12354 \newcommand*{\glsxtrLatinWynn}{%  
12355   \glsheX 01BF,\glsheX 01F7% wynn  
12356 }
```

trLatinInsularG

```
12357 \newcommand*{\glsxtrLatinInsularG}{%  
12358   \glsheX 1D79,\glsheX A77D% insular G  
12359   \string; g, G  
12360 }
```

sxtrLatinOslash

```
12361 \newcommand*{\glsxtrLatinOslash}{%  
12362   \glsheX 00F8,\glsheX 00D8% \o, \O  
12363 }
```

sxtrLatinLslash

```
12364 \newcommand*{\glsxtrLatinLslash}{%  
12365   \glsheX 0142,\glsheX 0141% \l, \L  
12366 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12367 \newcommand*{\glsxtrMathUpGreekIrules}{%  
12368   \glsxtrUpAlpha  
12369   \string<\glsxtrUpBeta  
12370   \string<\glsxtrUpGamma  
12371   \string<\glsxtrUpDelta  
12372   \string<\glsxtrUpEpsilon  
12373   \string<\glsxtrUpDigamma  
12374   \string<\glsxtrUpZeta  
12375   \string<\glsxtrUpEta  
12376   \string<\glsxtrUpTheta  
12377   \string<\glsxtrUpIota  
12378   \string<\glsxtrUpKappa  
12379   \string<\glsxtrUpLambda  
12380   \string<\glsxtrUpMu  
12381   \string<\glsxtrUpNu  
12382   \string<\glsxtrUpXi  
12383   \string<\glsxtrUpOmicron  
12384   \string<\glsxtrUpPi  
12385   \string<\glsxtrUpRho  
12386   \string<\glsxtrUpSigma  
12387   \string<\glsxtrUpTau  
12388   \string<\glsxtrUpUpsilon  
12389   \string<\glsxtrUpPhi  
12390   \string<\glsxtrUpChi  
12391   \string<\glsxtrUpPsi  
12392   \string<\glsxtrUpOmega  
12393 }
```

hUpGreekIIrules Doesn't include digamma.

```
12394 \newcommand*{\glxtrMathUpGreekIIrules}{%
12395   \glxtrUpAlpha
12396   \string<\glxtrUpBeta
12397   \string<\glxtrUpGamma
12398   \string<\glxtrUpDelta
12399   \string<\glxtrUpEpsilon
12400   \string<\glxtrUpZeta
12401   \string<\glxtrUpEta
12402   \string<\glxtrUpTheta
12403   \string<\glxtrUpIota
12404   \string<\glxtrUpKappa
12405   \string<\glxtrUpLambda
12406   \string<\glxtrUpMu
12407   \string<\glxtrUpNu
12408   \string<\glxtrUpXi
12409   \string<\glxtrUpOmicron
12410   \string<\glxtrUpPi
12411   \string<\glxtrUpRho
12412   \string<\glxtrUpSigma
12413   \string<\glxtrUpTau
12414   \string<\glxtrUpUpsilon
12415   \string<\glxtrUpPhi
12416   \string<\glxtrUpChi
12417   \string<\glxtrUpPsi
12418   \string<\glxtrUpOmega
12419 }
```

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.

```
12420 \newcommand*{\glxtrMathItalicGreekIrules}{%
12421   \glxtrMathItalicAlpha
12422   \string<\glxtrMathItalicBeta
12423   \string<\glxtrMathItalicGamma
12424   \string<\glxtrMathItalicDelta
12425   \string<\glxtrMathItalicEpsilon
12426   \string<\glxtrUpDigamma
12427   \string<\glxtrMathItalicZeta
12428   \string<\glxtrMathItalicEta
12429   \string<\glxtrMathItalicTheta
12430   \string<\glxtrMathItalicIota
12431   \string<\glxtrMathItalicKappa
12432   \string<\glxtrMathItalicLambda
12433   \string<\glxtrMathItalicMu
12434   \string<\glxtrMathItalicNu
12435   \string<\glxtrMathItalicXi
12436   \string<\glxtrMathItalicOmicron
12437   \string<\glxtrMathItalicPi
12438   \string<\glxtrMathItalicRho
```

```

12439 \string<\glxtrMathItalicSigma
12440 \string<\glxtrMathItalicTau
12441 \string<\glxtrMathItalicUpsilon
12442 \string<\glxtrMathItalicPhi
12443 \string<\glxtrMathItalicChi
12444 \string<\glxtrMathItalicPsi
12445 \string<\glxtrMathItalicOmega
12446 }

```

licGreekIIrules Doesn't include digamma.

```

12447 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12448 \glxtrMathItalicAlpha
12449 \string<\glxtrMathItalicBeta
12450 \string<\glxtrMathItalicGamma
12451 \string<\glxtrMathItalicDelta
12452 \string<\glxtrMathItalicEpsilon
12453 \string<\glxtrMathItalicZeta
12454 \string<\glxtrMathItalicEta
12455 \string<\glxtrMathItalicTheta
12456 \string<\glxtrMathItalicIota
12457 \string<\glxtrMathItalicKappa
12458 \string<\glxtrMathItalicLambda
12459 \string<\glxtrMathItalicMu
12460 \string<\glxtrMathItalicNu
12461 \string<\glxtrMathItalicXi
12462 \string<\glxtrMathItalicOmicron
12463 \string<\glxtrMathItalicPi
12464 \string<\glxtrMathItalicRho
12465 \string<\glxtrMathItalicSigma
12466 \string<\glxtrMathItalicTau
12467 \string<\glxtrMathItalicUpsilon
12468 \string<\glxtrMathItalicPhi
12469 \string<\glxtrMathItalicChi
12470 \string<\glxtrMathItalicPsi
12471 \string<\glxtrMathItalicOmega
12472 }

```

pperGreekIrules Upper case only (includes upright digamma).

```

12473 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12474 \glshex 1D6E2% upper case alpha (maths italic)
12475 \string<\glshex 1D6E3% upper case beta (maths italic)
12476 \string<\glshex 1D6E4% upper case gamma (maths italic)
12477 \string<\glshex 1D6E5% upper case delta (maths italic)
12478 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12479 \string<\glshex 03DC% upper case digamma
12480 \string<\glshex 1D6E7% upper case zeta (maths italic)
12481 \string<\glshex 1D6E8% upper case eta (maths italic)
12482 \string<\glshex 1D6E9% upper case theta (maths italic)
12483 \string=<\glshex 1D6F3% upper case theta variant (maths italic)

```

```

12484 \string<\glshex 1D6EA% upper case iota (maths italic)
12485 \string<\glshex 1D6EB% upper case kappa (maths italic)
12486 \string<\glshex 1D6EC% upper case lambda (maths italic)
12487 \string<\glshex 1D6ED% upper case mu (maths italic)
12488 \string<\glshex 1D6EE% upper case nu (maths italic)
12489 \string<\glshex 1D6EF% upper case xi (maths italic)
12490 \string<\glshex 1D6F0% upper case omicron (maths italic)
12491 \string<\glshex 1D6F1% upper case pi (maths italic)
12492 \string<\glshex 1D6F2% upper case rho (maths italic)
12493 \string<\glshex 1D6F4% upper case sigma (maths italic)
12494 \string<\glshex 1D6F5% upper case tau (maths italic)
12495 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12496 \string<\glshex 1D6F7% upper case phi (maths italic)
12497 \string<\glshex 1D6F8% upper case chi (maths italic)
12498 \string<\glshex 1D6F9% upper case psi (maths italic)
12499 \string<\glshex 1D6FA% upper case omega (maths italic)
12500 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

12501 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
12502 \glshex 1D6E2% upper case alpha (maths italic)
12503 \string<\glshex 1D6E3% upper case beta (maths italic)
12504 \string<\glshex 1D6E4% upper case gamma (maths italic)
12505 \string<\glshex 1D6E5% upper case delta (maths italic)
12506 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12507 \string<\glshex 1D6E7% upper case zeta (maths italic)
12508 \string<\glshex 1D6E8% upper case eta (maths italic)
12509 \string<\glshex 1D6E9% upper case theta (maths italic)
12510 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12511 \string<\glshex 1D6EA% upper case iota (maths italic)
12512 \string<\glshex 1D6EB% upper case kappa (maths italic)
12513 \string<\glshex 1D6EC% upper case lambda (maths italic)
12514 \string<\glshex 1D6ED% upper case mu (maths italic)
12515 \string<\glshex 1D6EE% upper case nu (maths italic)
12516 \string<\glshex 1D6EF% upper case xi (maths italic)
12517 \string<\glshex 1D6F0% upper case omicron (maths italic)
12518 \string<\glshex 1D6F1% upper case pi (maths italic)
12519 \string<\glshex 1D6F2% upper case rho (maths italic)
12520 \string<\glshex 1D6F4% upper case sigma (maths italic)
12521 \string<\glshex 1D6F5% upper case tau (maths italic)
12522 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12523 \string<\glshex 1D6F7% upper case phi (maths italic)
12524 \string<\glshex 1D6F8% upper case chi (maths italic)
12525 \string<\glshex 1D6F9% upper case psi (maths italic)
12526 \string<\glshex 1D6FA% upper case omega (maths italic)
12527 }

```

lowerGreekIrules Lower case only (includes upright digamma).

```

12528 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%

```

```

12529 \glshex 1D6FC% lower case alpha (maths italic)
12530 \string<\glshex 1D6FD% lower case beta (maths italic)
12531 \string<\glshex 1D6FE% lower case gamma (maths italic)
12532 \string<\glshex 1D6FF% lower case delta (maths italic)
12533 \string<\glshex 1D700% lower case epsilon (maths italic)
12534 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12535 \string<\glshex 03DD% lower case digamma
12536 \string<\glshex 1D701% lower case zeta (maths italic)
12537 \string<\glshex 1D702% lower case eta (maths italic)
12538 \string<\glshex 1D703% lower case theta (maths italic)
12539 \string=\glshex 1D717% lower case theta variant (maths italic)
12540 \string<\glshex 1D704% lower case iota (maths italic)
12541 \string<\glshex 1D705% lower case kappa (maths italic)
12542 \string=\glshex 1D718% lower case kappa variant (maths italic)
12543 \string<\glshex 1D706% lower case lambda (maths italic)
12544 \string<\glshex 1D707% lower case mu (maths italic)
12545 \string<\glshex 1D708% lower case nu (maths italic)
12546 \string<\glshex 1D709% lower case xi (maths italic)
12547 \string<\glshex 1D70A% lower case omicron (maths italic)
12548 \string<\glshex 1D70B% lower case pi (maths italic)
12549 \string=\glshex 1D71B% lower case pi variant (maths italic)
12550 \string<\glshex 1D70C% lower case rho (maths italic)
12551 \string=\glshex 1D71A% lower case rho variant (maths italic)
12552 \string<\glshex 1D70D% lower case final sigma (maths italic)
12553 \string=\glshex 1D70E% lower case sigma (maths italic)
12554 \string<\glshex 1D70F% lower case tau (maths italic)
12555 \string<\glshex 1D710% lower case upsilon (maths italic)
12556 \string<\glshex 1D711% lower case phi (maths italic)
12557 \string=\glshex 1D719% lower case phi variant (maths italic)
12558 \string<\glshex 1D712% lower case chi (maths italic)
12559 \string<\glshex 1D713% lower case psi (maths italic)
12560 \string<\glshex 1D714% lower case omega (maths italic)
12561 }

```

LowerGreeklIrules Lower case only (doesn't includes upright digamma).

```

12562 \newcommand*{\glxtrMathItalicLowerGreeklIrules}{%
12563 \glshex 1D6FC% lower case alpha (maths italic)
12564 \string<\glshex 1D6FD% lower case beta (maths italic)
12565 \string<\glshex 1D6FE% lower case gamma (maths italic)
12566 \string<\glshex 1D6FF% lower case delta (maths italic)
12567 \string<\glshex 1D700% lower case epsilon (maths italic)
12568 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12569 \string<\glshex 1D701% lower case zeta (maths italic)
12570 \string<\glshex 1D702% lower case eta (maths italic)
12571 \string<\glshex 1D703% lower case theta (maths italic)
12572 \string=\glshex 1D717% lower case theta variant (maths italic)
12573 \string<\glshex 1D704% lower case iota (maths italic)
12574 \string<\glshex 1D705% lower case kappa (maths italic)
12575 \string=\glshex 1D718% lower case kappa variant (maths italic)

```



```

12576 \string<\glshex 1D706% lower case lambda (maths italic)
12577 \string<\glshex 1D707% lower case mu (maths italic)
12578 \string<\glshex 1D708% lower case nu (maths italic)
12579 \string<\glshex 1D709% lower case xi (maths italic)
12580 \string<\glshex 1D70A% lower case omicron (maths italic)
12581 \string<\glshex 1D70B% lower case pi (maths italic)
12582 \string=<\glshex 1D71B% lower case pi variant (maths italic)
12583 \string<\glshex 1D70C% lower case rho (maths italic)
12584 \string=<\glshex 1D71A% lower case rho variant (maths italic)
12585 \string<\glshex 1D70D% lower case final sigma (maths italic)
12586 \string=<\glshex 1D70E% lower case sigma (maths italic)
12587 \string<\glshex 1D70F% lower case tau (maths italic)
12588 \string<\glshex 1D710% lower case upsilon (maths italic)
12589 \string<\glshex 1D711% lower case phi (maths italic)
12590 \string=<\glshex 1D719% lower case phi variant (maths italic)
12591 \string<\glshex 1D712% lower case chi (maths italic)
12592 \string<\glshex 1D713% lower case psi (maths italic)
12593 \string<\glshex 1D714% lower case omega (maths italic)
12594 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12595 \newcommand*{\glxtrMathGreekIrules}{%
12596 \glxtrMathItalicAlpha
12597 \string;\glxtrUpAlpha
12598 \string<\glxtrMathItalicBeta
12599 \string;\glxtrUpBeta
12600 \string<\glxtrMathItalicGamma
12601 \string;\glxtrUpGamma
12602 \string<\glxtrMathItalicDelta
12603 \string;\glxtrUpDelta
12604 \string<\glxtrMathItalicEpsilon
12605 \string;\glxtrUpEpsilon
12606 \string<\glxtrUpDigamma
12607 \string<\glxtrMathItalicZeta
12608 \string;\glxtrUpZeta
12609 \string<\glxtrMathItalicEta
12610 \string;\glxtrUpEta
12611 \string<\glxtrMathItalicTheta
12612 \string;\glxtrUpTheta
12613 \string<\glxtrMathItalicIota
12614 \string;\glxtrUpIota
12615 \string<\glxtrMathItalicKappa
12616 \string;\glxtrUpKappa
12617 \string<\glxtrMathItalicLambda
12618 \string;\glxtrUpLambda
12619 \string<\glxtrMathItalicMu
12620 \string;\glxtrUpMu
12621 \string<\glxtrMathItalicNu
12622 \string;\glxtrUpNu

```

```

12623 \string<\glxtrMathItalicXi
12624 \string;\glxtrUpXi
12625 \string<\glxtrMathItalicOmicron
12626 \string;\glxtrUpOmicron
12627 \string<\glxtrMathItalicPi
12628 \string;\glxtrUpPi
12629 \string<\glxtrMathItalicRho
12630 \string;\glxtrUpRho
12631 \string<\glxtrMathItalicSigma
12632 \string;\glxtrUpSigma
12633 \string<\glxtrMathItalicTau
12634 \string;\glxtrUpTau
12635 \string<\glxtrMathItalicUpsilon
12636 \string;\glxtrUpUpsilon
12637 \string<\glxtrMathItalicPhi
12638 \string;\glxtrUpPhi
12639 \string<\glxtrMathItalicChi
12640 \string;\glxtrUpChi
12641 \string<\glxtrMathItalicPsi
12642 \string;\glxtrUpPsi
12643 \string<\glxtrMathItalicOmega
12644 \string;\glxtrUpOmega
12645 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

12646 \newcommand*{\glxtrMathGreekIIrules}{%
12647 \glxtrMathItalicAlpha
12648 \string;\glxtrUpAlpha
12649 \string<\glxtrMathItalicBeta
12650 \string;\glxtrUpBeta
12651 \string<\glxtrMathItalicGamma
12652 \string;\glxtrUpGamma
12653 \string<\glxtrMathItalicDelta
12654 \string;\glxtrUpDelta
12655 \string<\glxtrMathItalicEpsilon
12656 \string;\glxtrUpEpsilon
12657 \string<\glxtrMathItalicZeta
12658 \string;\glxtrUpZeta
12659 \string<\glxtrMathItalicEta
12660 \string;\glxtrUpEta
12661 \string<\glxtrMathItalicTheta
12662 \string;\glxtrUpTheta
12663 \string<\glxtrMathItalicIota
12664 \string;\glxtrUpIota
12665 \string<\glxtrMathItalicKappa
12666 \string;\glxtrUpKappa
12667 \string<\glxtrMathItalicLambda
12668 \string;\glxtrUpLambda
12669 \string<\glxtrMathItalicMu

```

```

12670 \string;\glxtrUpMu
12671 \string<\glxtrMathItalicNu
12672 \string;\glxtrUpNu
12673 \string<\glxtrMathItalicXi
12674 \string;\glxtrUpXi
12675 \string<\glxtrMathItalicOmicron
12676 \string;\glxtrUpOmicron
12677 \string<\glxtrMathItalicPi
12678 \string;\glxtrUpPi
12679 \string<\glxtrMathItalicRho
12680 \string;\glxtrUpRho
12681 \string<\glxtrMathItalicSigma
12682 \string;\glxtrUpSigma
12683 \string<\glxtrMathItalicTau
12684 \string;\glxtrUpTau
12685 \string<\glxtrMathItalicUpsilon
12686 \string;\glxtrUpUpsilon
12687 \string<\glxtrMathItalicPhi
12688 \string;\glxtrUpPhi
12689 \string<\glxtrMathItalicChi
12690 \string;\glxtrUpChi
12691 \string<\glxtrMathItalicPsi
12692 \string;\glxtrUpPsi
12693 \string<\glxtrMathItalicOmega
12694 \string;\glxtrUpOmega
12695 }

```

\glxtrUpAlpha

```

12696 \newcommand*{\glxtrUpAlpha}{%
12697 \glshex 03B1,% lower case alpha
12698 \glshex 0391% upper case alpha
12699 }

```

\glxtrUpBeta

```

12700 \newcommand*{\glxtrUpBeta}{%
12701 \glshex 03B2,% lower case beta
12702 \glshex 0392% upper case beta
12703 }

```

\glxtrUpGamma

```

12704 \newcommand*{\glxtrUpGamma}{%
12705 \glshex 03B3,% lower case gamma
12706 \glshex 0393% upper case gamma
12707 }

```

\glxtrUpDelta

```

12708 \newcommand*{\glxtrUpDelta}{%
12709 \glshex 03B4,% lower case delta
12710 \glshex 0394% upper case delta

```

12711 }

glsxtrUpEpsilon

```
12712 \newcommand*{\glsxtrUpEpsilon}{%
12713   \glsheX 03B5% lower case epsilon
12714   \string=\glsheX 03F5,% lower case epsilon variant
12715   \glsheX 0395% upper case epsilon
12716 }
```

glsxtrUpDigamma

```
12717 \newcommand*{\glsxtrUpDigamma}{%
12718   \glsheX 03DD,% lower case digamma
12719   \glsheX 03DC% upper case digamma
12720 }
```

\glsxtrUpZeta

```
12721 \newcommand*{\glsxtrUpZeta}{%
12722   \glsheX 03B6,% lower case zeta
12723   \glsheX 0396% upper case zeta
12724 }
```

\glsxtrUpEta

```
12725 \newcommand*{\glsxtrUpEta}{%
12726   \glsheX 03B7,% lower case eta
12727   \glsheX 0397% upper case eta
12728 }
```

\glsxtrUpTheta

```
12729 \newcommand*{\glsxtrUpTheta}{%
12730   \glsheX 03B8% lower case theta
12731   \string=\glsheX 03D1,% lower case theta variant
12732   \glsheX 0398% upper case theta
12733 }
```

\glsxtrUpIota

```
12734 \newcommand*{\glsxtrUpIota}{%
12735   \glsheX 03B9,% lower case iota
12736   \glsheX 0399% upper case iota
12737 }
```

\glsxtrUpKappa

```
12738 \newcommand*{\glsxtrUpKappa}{%
12739   \glsheX 03BA% lower case kappa
12740   \string=\glsheX 03F0,% lower case kappa variant
12741   \glsheX 039A% upper case kappa
12742 }
```

\glxtrUpLambda

```
12743 \newcommand*{\glxtrUpLambda}{%
12744   \glshex 03BB,% lower lambda
12745   \glshex 039B% upper case lambda
12746 }
```

\glxtrUpMu

```
12747 \newcommand*{\glxtrUpMu}{%
12748   \glshex 03BC,% lower case mu
12749   \glshex 039C% upper case mu
12750 }
```

\glxtrUpNu

```
12751 \newcommand*{\glxtrUpNu}{%
12752   \glshex 03BD,% lower case nu
12753   \glshex 039D% upper case nu
12754 }
```

\glxtrUpXi

```
12755 \newcommand*{\glxtrUpXi}{%
12756   \glshex 03BE,% lower case xi
12757   \glshex 039E% upper case xi
12758 }
```

\glxtrUpOmicron

```
12759 \newcommand*{\glxtrUpOmicron}{%
12760   \glshex 03BF,% lower case omicron
12761   \glshex 039F% upper case omicron
12762 }
```

\glxtrUpPi

```
12763 \newcommand*{\glxtrUpPi}{%
12764   \glshex 03C0% lower case pi
12765   \string=\glshex 03D6,% lower case pi variant
12766   \glshex 03A0% upper case pi
12767 }
```

\glxtrUpRho

```
12768 \newcommand*{\glxtrUpRho}{%
12769   \glshex 03C1% lower case rho
12770   \string=\glshex 03F1,% lower case rho variant
12771   \glshex 03A1% upper case rho
12772 }
```

\glxtrUpSigma

```
12773 \newcommand*{\glxtrUpSigma}{%
12774   \glshex 03C2% lower case sigma
12775   \string=\glshex 03C3,% lower case sigma
```

```
12776 \glshex 03A3% upper case sigma
12777 }
```

`\glxtrUpTau`

```
12778 \newcommand*{\glxtrUpTau}{%
12779 \glshex 03C4,% lower case tau
12780 \glshex 03A4% upper case tau
12781 }
```

`glxtrUpUpsilon`

```
12782 \newcommand*{\glxtrUpUpsilon}{%
12783 \glshex 03C5,% lower case upsilon
12784 \glshex 03A5% upper case upsilon
12785 }
```

`\glxtrUpPhi`

```
12786 \newcommand*{\glxtrUpPhi}{%
12787 \glshex 03C6% lower case phi
12788 \string=\glshex 03D5,% lower case phi variant
12789 \glshex 03A6% upper case phi
12790 }
```

`\glxtrUpChi`

```
12791 \newcommand*{\glxtrUpChi}{%
12792 \glshex 03C7,% lower case chi
12793 \glshex 03A7% upper case chi
12794 }
```

`\glxtrUpPsi`

```
12795 \newcommand*{\glxtrUpPsi}{%
12796 \glshex 03C8,% lower case psi
12797 \glshex 03A8% upper case psi
12798 }
```

`\glxtrUpOmega`

```
12799 \newcommand*{\glxtrUpOmega}{%
12800 \glshex 03C9,% lower case omega
12801 \glshex 03A9% upper case omega
12802 }
```

`MathItalicAlpha`

```
12803 \newcommand*{\glxtrMathItalicAlpha}{%
12804 \glshex 1D6FC,% lower case alpha (maths italic)
12805 \glshex 1D6E2% upper case alpha (maths italic)
12806 }
```

`rMathItalicBeta`

```
12807 \newcommand*{\glxtrMathItalicBeta}{%
```

```

12808 \glshex 1D6FD,% lower case beta (maths italic)
12809 \glshex 1D6E3% upper case beta (maths italic)
12810 }

```

MathItalicGamma

```

12811 \newcommand*{\glxtrMathItalicGamma}{%
12812 \glshex 1D6FE,% lower case gamma (maths italic)
12813 \glshex 1D6E4% upper case gamma (maths italic)
12814 }

```

MathItalicDelta

```

12815 \newcommand*{\glxtrMathItalicDelta}{%
12816 \glshex 1D6FF,% lower case delta (maths italic)
12817 \glshex 1D6E5% upper case delta (maths italic)
12818 }

```

thItalicEpsilon

```

12819 \newcommand*{\glxtrMathItalicEpsilon}{%
12820 \glshex 1D700% lower case epsilon (maths italic)
12821 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12822 \glshex 1D6E6% upper case epsilon (maths italic)
12823 }

```

rMathItalicZeta

```

12824 \newcommand*{\glxtrMathItalicZeta}{%
12825 \glshex 1D701,% lower case zeta (maths italic)
12826 \glshex 1D6E7% upper case zeta (maths italic)
12827 }

```

trMathItalicEta

```

12828 \newcommand*{\glxtrMathItalicEta}{%
12829 \glshex 1D702,% lower case eta (maths italic)
12830 \glshex 1D6E8% upper case eta (maths italic)
12831 }

```

MathItalicTheta

```

12832 \newcommand*{\glxtrMathItalicTheta}{%
12833 \glshex 1D703% lower case theta (maths italic)
12834 \string=\glshex 1D717,% lower case theta variant (maths italic)
12835 \glshex 1D6E9% upper case theta (maths italic)
12836 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12837 }

```

rMathItalicIota

```

12838 \newcommand*{\glxtrMathItalicIota}{%
12839 \glshex 1D704,% lower case iota (maths italic)
12840 \glshex 1D6EA% upper case iota (maths italic)
12841 }

```

MathItalicKappa

```
12842 \newcommand*{\glxtrMathItalicKappa}{%
12843   \glshex 1D705% lower case kappa (maths italic)
12844   \string=\glshex 1D718,% lower case kappa variant (maths italic)
12845   \glshex 1D6EB% upper case kappa (maths italic)
12846 }
```

athItalicLambda

```
12847 \newcommand*{\glxtrMathItalicLambda}{%
12848   \glshex 1D706,% lower case lambda (maths italic)
12849   \glshex 1D6EC% upper case lambda (maths italic)
12850 }
```

xtrMathItalicMu

```
12851 \newcommand*{\glxtrMathItalicMu}{%
12852   \glshex 1D707,% lower case mu (maths italic)
12853   \glshex 1D6ED% upper case mu (maths italic)
12854 }
```

xtrMathItalicNu

```
12855 \newcommand*{\glxtrMathItalicNu}{%
12856   \glshex 1D708,% lower case nu (maths italic)
12857   \glshex 1D6EE% upper case nu (maths italic)
12858 }
```

xtrMathItalicXi

```
12859 \newcommand*{\glxtrMathItalicXi}{%
12860   \glshex 1D709,% lower case xi (maths italic)
12861   \glshex 1D6EF% upper case xi (maths italic)
12862 }
```

thItalicOmicron

```
12863 \newcommand*{\glxtrMathItalicOmicron}{%
12864   \glshex 1D70A,% lower case omicron (maths italic)
12865   \glshex 1D6F0% upper case omicron (maths italic)
12866 }
```

xtrMathItalicPi

```
12867 \newcommand*{\glxtrMathItalicPi}{%
12868   \glshex 1D70B% lower case pi (maths italic)
12869   \string=\glshex 1D71B,% lower case pi variant (maths italic)
12870   \glshex 1D6F1% upper case pi (maths italic)
12871 }
```

trMathItalicRho

```
12872 \newcommand*{\glxtrMathItalicRho}{%
12873   \glshex 1D70C% lower case rho (maths italic)
12874   \string=\glshex 1D71A,% lower case rho variant (maths italic)
```



```

12875 \glshex 1D6F2% upper case rho (maths italic)
12876 }

```

MathItalicSigma

```

12877 \newcommand*{\glsxtrMathItalicSigma}{%
12878 \glshex 1D70D% lower case final sigma (maths italic)
12879 \string=\glshex 1D70E,% lower case sigma (maths italic)
12880 \glshex 1D6F4% upper case sigma (maths italic)
12881 }

```

trMathItalicTau

```

12882 \newcommand*{\glsxtrMathItalicTau}{%
12883 \glshex 1D70F,% lower case tau (maths italic)
12884 \glshex 1D6F5% upper case tau (maths italic)
12885 }

```

thItalicUpsilon

```

12886 \newcommand*{\glsxtrMathItalicUpsilon}{%
12887 \glshex 1D710,% lower case upsilon (maths italic)
12888 \glshex 1D6F6% upper case upsilon (maths italic)
12889 }

```

trMathItalicPhi

```

12890 \newcommand*{\glsxtrMathItalicPhi}{%
12891 \glshex 1D711% lower case phi (maths italic)
12892 \string=\glshex 1D719,% lower case phi variant (maths italic)
12893 \glshex 1D6F7% upper case phi (maths italic)
12894 }

```

trMathItalicChi

```

12895 \newcommand*{\glsxtrMathItalicChi}{%
12896 \glshex 1D712,% lower case chi (maths italic)
12897 \glshex 1D6F8% upper case chi (maths italic)
12898 }

```

trMathItalicPsi

```

12899 \newcommand*{\glsxtrMathItalicPsi}{%
12900 \glshex 1D713,% lower case psi (maths italic)
12901 \glshex 1D6F9% upper case psi (maths italic)
12902 }

```

MathItalicOmega

```

12903 \newcommand*{\glsxtrMathItalicOmega}{%
12904 \glshex 1D714,% lower case omega (maths italic)
12905 \glshex 1D6FA% upper case omega (maths italic)
12906 }

```

thItalicPartial

```
12907 \newcommand*{\glxtrMathItalicPartial}{%
12908   \glshex 1D715% partial differential (maths italic)
12909 }
```

MathItalicNabla

```
12910 \newcommand*{\glxtrMathItalicNabla}{%
12911   \glshex 1D6FB% nabla (maths italic)
12912 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12913 \newcommand*{\glxtrdigitrules}{%
12914   0\string=\glshex 2080\string=\glshex 2070
12915   \string<1\string=\glshex 2081\string=\glshex 00B9
12916   \string<2\string=\glshex 2082\string=\glshex 00B2
12917   \string<3\string=\glshex 2083\string=\glshex 00B3
12918   \string<4\string=\glshex 2084\string=\glshex 2074
12919   \string<5\string=\glshex 2085\string=\glshex 2075
12920   \string<6\string=\glshex 2086\string=\glshex 2076
12921   \string<7\string=\glshex 2087\string=\glshex 2077
12922   \string<8\string=\glshex 2088\string=\glshex 2078
12923   \string<9\string=\glshex 2089\string=\glshex 2079
12924 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12925 \newcommand*{\glxtrBasicDigitrules}{%
12926   0\string<1\string<2\string<3\string<4%
12927   \string<5\string<6\string<7\string<8\string<9%
12928 }
```

criptDigitrules Subscript digits.

```
12929 \newcommand*{\glxtrSubScriptDigitrules}{%
12930   \glshex 2080% subscript 0
12931   \string<\glshex 2081% subscript 1
12932   \string<\glshex 2082% subscript 2
12933   \string<\glshex 2083% subscript 3
12934   \string<\glshex 2084% subscript 4
12935   \string<\glshex 2085% subscript 5
12936   \string<\glshex 2086% subscript 6
12937   \string<\glshex 2087% subscript 7
12938   \string<\glshex 2088% subscript 8
12939   \string<\glshex 2089% subscript 9
12940 }
```

criptDigitrules Superscript digits.

```
12941 \newcommand*{\glxtrSuperScriptDigitrules}{%
12942   \glshex 2070% superscript 0
12943   \string<\glshex 00B9% superscript 1
```

```

12944 \string<\glshex 00B2% superscript 2
12945 \string<\glshex 00B3% superscript 3
12946 \string<\glshex 2074% superscript 4
12947 \string<\glshex 2075% superscript 5
12948 \string<\glshex 2076% superscript 6
12949 \string<\glshex 2077% superscript 7
12950 \string<\glshex 2078% superscript 8
12951 \string<\glshex 2079% superscript 9
12952 }

```

trfractionrules Vulgar fractions.

```

12953 \newcommand*{\glxtrfractionrules}{%
12954 \glshex 215F% fraction numerator one (1/)
12955 \string<\glshex 2189% zero thirds (0/3 = 0)
12956 \string<\glshex 2152% one tenth (1/10 = 0.1)
12957 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12958 \string<\glshex 215B% one eighth (1/8 = 0.125)
12959 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12960 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12961 \string<\glshex 2155% one fifth (1/5 = 0.2)
12962 \string<\glshex 00BC% one quarter (1/4 = 0.25)
12963 \string<\glshex 2153% one third (1/3 ~ 0.333)
12964 \string<\glshex 215C% three eighths (3/8 = 0.375)
12965 \string<\glshex 2156% two fifths (2/5 = 0.4)
12966 \string<\glshex 00BD% one half (1/2 = 0.5)
12967 \string<\glshex 2157% three fifths (3/5 = 0.6)
12968 \string<\glshex 215D% five eighths (5/8 = 0.625)
12969 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
12970 \string<\glshex 00BE% three quarters (3/4 = 0.75)
12971 \string<\glshex 2158% four fifths (4/5 = 0.8)
12972 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
12973 \string<\glshex 215E% seven eighths (7/8 = 0.875)
12974 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

12975 \renewcommand{\@glxtrdialecthook}{%
12976 \ifundef\CurrentTrackedScript
12977 {%
12978 \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
12979 {%
12980 \edef\CurrentTrackedScript{%
12981 \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
12982 }%
12983 {}%
12984 }%
12985 {}%
12986 \ifdef\CurrentTrackedScript
12987 {%
12988 \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix

```

```

12989 \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
12990 \let\CurrentTrackedTag\CurrentTrackedScript
12991 \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.1df}
12992 {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
12993 {}%
12994 \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
12995 }%
12996 {}%
12997 }

```

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.

```

12998 \ifdef\glsxtr@loaddialect
12999 {%
13000 \@ifpackageloaded{tracklang}
13001 {%
13002 \AnyTrackedLanguages
13003 {%
13004 \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13005 }%
13006 {}%
13007 }
13008 {}
13009 }
13010 {}

```

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:

```
13011 \NeedsTeXFormat{LaTeX2e}
13012 \ProvidesPackage{glossaries-extra-stylemods}[2018/07/26 v1.33 (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`

```
13013 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13014 \DeclareOption{all}{%
13015   \appto\@glxtr@loadstyles{%
13016     \RequirePackage{glossary-inline}%
13017     \RequirePackage{glossary-list}%
13018     \RequirePackage{glossary-tree}%
13019     \RequirePackage{glossary-mcols}%
13020     \RequirePackage{glossary-long}%
13021     \RequirePackage{glossary-longragged}%
13022     \RequirePackage{glossary-longbooktabs}%
13023     \RequirePackage{glossary-super}%
13024     \RequirePackage{glossary-superragged}%
13025     \RequirePackage{glossary-bookindex}%
13026   }
13027 }

13028 \DeclareOption*{%
13029   \IfFileExists{glossary-\CurrentOption.sty}
13030   {\eappto\@glxtr@loadstyles{%
13031     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13032   }%
13033   {%
13034     \PackageError{glossaries-extra-styles}%

```

```

13035     {Unknown option ‘\CurrentOption’}{}%
13036   }%
13037 }

```

Process the package options:

```
13038 \ProcessOptions
```

Load the required packages:

```
13039 \@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`.

```
13040 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

13041 \providecommand{\renewglossarystyle}[2]{%
13042   \ifcsundef{@glsstyle@#1}%
13043   {%
13044     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13045   }%
13046   {%
13047     \csdef{@glsstyle@#1}{#2}%
13048   }%
13049 }

```

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.

```

13050 \ifdef{@glsstyle@listdotted}
13051 {%
13052   \renewglossarystyle{listdotted}{%
13053     \setglossarystyle{list}%
13054     \renewcommand*{\glossentry}[2]{%
13055       \item[]\makebox[\glslistdottedwidth][l]{%
13056         \glsentryitem{##1}%
13057         \glstarget{##1}{\glossentryname{##1}}%
13058         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13059         \glossentrydesc{##1}\glspostdescription}%
13060     \renewcommand*{\subglossentry}[3]{%
13061       \item[]\makebox[\glslistdottedwidth][l]{%
13062         \glssubentryitem{##2}%
13063         \glstarget{##2}{\glossentryname{##2}}%
13064         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13065         \glossentrydesc{##2}\glspostdescription}%
13066   }

```

```
13067 }
13068 {%
```

Assume the style isn't required if it hasn't already been defined.

```
13069 }
```

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.

```
13070 \ifdef{\@glsstyle@list}
13071 {%
```

listprelocation Space before number list for top-level entries.

```
13072 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13073 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13074 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13075 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
13076 \renewglossarystyle{list}{%
13077   \renewenvironment{theglossary}%
13078     {\begin{description}}{\end{description}}%
13079   \renewcommand*\{glossaryheader}[1]{%
13080     \renewcommand*\{glsgroupheading}[1]{%
13081       \renewcommand*\{glossentry}[2]{%
13082         \item[\glssentryitem{##1}]%
13083           \glstarget{##1}{\glossentryname{##1}}]
13084         \glslistdesc{##1}\glslistprelocation ##2}%
13085     \renewcommand*\{subglossentry}[3]{%
13086       \glssubentryitem{##2}%
13087       \glstarget{##2}{\strut}\space
13088       \glslistdesc{##2}%
13089       \glslistchildprelocation ##3\glslistchildpostlocation}%
13090   \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13091 }
13092 }
13093 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13094 \ifdef{\@glsstyle@altlist}
13095 {%
13096   \renewglossarystyle{altlist}{%
```

```

13097 \setglossarystyle{list}%
13098 \renewcommand*{\glossentry}[2]{%
13099   \item[\glsentryitem{##1}%
13100     \glstarget{##1}{\glossentryname{##1}}]%
13101   \mbox{}\par\nobreak\@afterheading
13102   \glslistdesc{##1}\glslistprelocation ##2}%
13103 \renewcommand{\subglossentry}[3]{%
13104   \par
13105   \glssubentryitem{##2}%
13106   \glstarget{##2}{\strut}\glslistdesc{##2}%
13107   \glslistchildprelocation ##3}%
13108 }
13109 }
13110 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13111 \ifdef{\@glsstyle@listgroup}
13112 {%
13113   \renewglossarystyle{listgroup}{%
13114     \setglossarystyle{list}%
13115     \renewcommand*{\glsgroupheading}[1]{%
13116       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13117       \mbox{}\par\nobreak\@afterheading
13118     }%
13119   }
13120 }
13121 {}

```

Similarly for listhypergroup.

```

13122 \ifdef{\@glsstyle@listhypergroup}
13123 {%
13124   \renewglossarystyle{listhypergroup}{%
13125     \setglossarystyle{list}%
13126     \renewcommand*{\glossaryheader}{%
13127       \glslistnavigationitem{\glsnavigation}}%
13128     \renewcommand*{\glsgroupheading}[1]{%
13129       \item[\glslistgroupheaderfmt
13130         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13131       \mbox{}\par\nobreak\@afterheading
13132     }%
13133   }
13134 }
13135 {}

```

Similarly for altlistgroup.

```

13136 \ifdef{\@glsstyle@altlistgroup}
13137 {%
13138   \renewglossarystyle{altlistgroup}{%
13139     \setglossarystyle{altlist}%
13140     \renewcommand*{\glsgroupheading}[1]{%
13141       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```



```

13142     \mbox{}\par\nobreak\@afterheading
13143   }%
13144 }
13145 }
13146 {}

```

Similarly for altlisthypergroup.

```

13147 \ifdef{\@glsstyle@altlisthypergroup}
13148 {%
13149   \renewglossarystyle{altlisthypergroup}{%
13150     \setglossarystyle{altlist}%
13151     \renewcommand*\glossaryheader{%
13152       \glslistnavigationitem{\glsnavigation}}%
13153     \renewcommand*\glsgroupheading[1]{%
13154       \item[\glslistgroupheaderfmt
13155         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13156       \mbox{}\par\nobreak\@afterheading
13157     }%
13158   }
13159 }
13160 {}

```

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`.

```

13161 \ifcsdef{@glsstyle@long}
13162 {%
13163   \renewglossarystyle{long}{%
13164     \renewenvironment{theglossary}%
13165       {\begin{longtable}{lp{\glsdescwidth}}}%
13166       {\end{longtable}}%
13167     \renewcommand*\glossaryheader{}%
13168     \renewcommand*\glsgroupheading[1]{}%
13169     \renewcommand{\glossentry}[2]{%
13170       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13171       \glossentrydesc{##1}\glspostdescription
13172       \glxtrprelocation ##2\tabularnewline
13173     }%
13174     \renewcommand{\subglossentry}[3]{%
13175       &
13176       \glssubentryitem{##2}%
13177       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13178       \glxtrprelocation ##3\tabularnewline
13179     }%
13180     \ifglsnogroupskip
13181       \renewcommand*\glsgroupskip{}%
13182     \else

```

```

13183     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13184     \fi
13185   }
13186 }
13187 {}

```

Three column style:

```

13188 \ifcsdef{@glsstyle@long3col}
13189 {%
13190   \renewglossarystyle{long3col}{%
13191     \renewenvironment{theglossary}%
13192       {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13193       {\end{longtable}}}%
13194     \renewcommand*{\glossaryheader}{}%
13195     \renewcommand*{\glsgroupheading}[1]{}%
13196     \renewcommand{\glossentry}[2]{%
13197       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13198       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13199     }%
13200     \renewcommand{\subglossentry}[3]{%
13201       &
13202       \glssubentryitem{##2}%
13203       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13204       ##3\tabularnewline
13205     }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

13206   \ifglsnogroupskip
13207     \renewcommand*{\glsgroupskip}{}%
13208   \else
13209     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13210   \fi
13211 }
13212 }
13213 {}

```

Four column style:

```

13214 \ifcsdef{@glsstyle@long4col}
13215 {%
13216   \renewglossarystyle{long4col}{%
13217     \renewenvironment{theglossary}%
13218       {\begin{longtable}{llll}}}%
13219       {\end{longtable}}}%
13220     \renewcommand*{\glossaryheader}{}%
13221     \renewcommand*{\glsgroupheading}[1]{}%
13222     \renewcommand{\glossentry}[2]{%
13223       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13224       \glossentrydesc{##1}\glspostdescription &
13225       \glossentrysymbol{##1} &
13226       ##2\tabularnewline

```

```

13227 }%
13228 \renewcommand{\subglossentry}[3]{%
13229     &
13230     \glssubentryitem{##2}%
13231     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13232     \glossentrysymbol{##2} & ##3\tabularnewline
13233 }%

13234 \ifglsgroupskip
13235     \renewcommand*\{glsgroupskip}{}%
13236 \else
13237     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13238 \fi
13239 }
13240 }
13241 {}

```

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`.

```

13242 \ifcsdef{@glstyle@longragged}
13243 {%
13244     \renewglossarystyle{longragged}{%
13245         \renewenvironment{theglossary}%
13246             {\begin{longtable}[l>\raggedright]p{\glsgdescwidth}}}%
13247             {\end{longtable}}}%
13248     \renewcommand*\{glossaryheader}{}%
13249     \renewcommand*\{glsgroupheading}[1]{}%
13250     \renewcommand{\glossentry}[2]{%
13251         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13252         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13253         \tabularnewline
13254     }%
13255     \renewcommand{\subglossentry}[3]{%
13256         &
13257         \glssubentryitem{##2}%
13258         \glstarget{##2}{\strut}\glossentrydesc{##2}%
13259         \glspostdescription\glxtrprelocation ##3%
13260         \tabularnewline
13261     }%
13262     \ifglsgroupskip
13263         \renewcommand*\{glsgroupskip}{}%
13264     \else
13265         \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13266 \fi
13267 }
13268 }
13269 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13270 \ifcsdef{@glsstyle@longragged3col}
13271 {%
13272 \renewglossarystyle{longragged3col}{%
13273 \renewenvironment{theglossary}%
13274 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13275 >{\raggedright}p{\glspagelistwidth}}}%
13276 {\end{longtable}}}%
13277 \renewcommand*{\glossaryheader}{}%
13278 \renewcommand*{\glsgroupheading}[1]{}%
13279 \renewcommand{\glossentry}[2]{%
13280 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13281 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13282 }%
13283 \renewcommand{\subglossentry}[3]{%
13284 &
13285 \glssubentryitem{##2}%
13286 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13287 ##3\tabularnewline
13288 }%
13289 \ifglsgroupskip
13290 \renewcommand*{\glsgroupskip}{}%
13291 \else
13292 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13293 \fi
13294 }
13295 }
13296 {}

```

Four column style:

```

13297 \ifcsdef{@glsstyle@altlongragged4col}
13298 {%
13299 \renewglossarystyle{altlongragged4col}{%
13300 \renewenvironment{theglossary}%
13301 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13302 >{\raggedright}p{\glspagelistwidth}}}%
13303 {\end{longtable}}}%
13304 \renewcommand*{\glossaryheader}{}%
13305 \renewcommand*{\glsgroupheading}[1]{}%
13306 \renewcommand{\glossentry}[2]{%
13307 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13308 \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13309 ##2\tabularnewline

```

```

13310 }%
13311 \renewcommand{\subglossentry}[3]{%
13312     &
13313     \glssubentryitem{##2}%
13314     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13315     \glossentrysymbol{##2} & ##3\tabularnewline
13316 }%

13317 \ifglsnogroupskip
13318     \renewcommand*{\glsgroupskip}{}%
13319 \else
13320     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13321 \fi
13322 }
13323 }
13324 {}

```

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`.

```

13325 \ifcsdef{@glsstyle@super}
13326 {%
13327     \renewglossarystyle{super}{%
13328         \renewenvironment{theglossary}%
13329             {\tablehead{}}\tabletail{}}%
13330         \begin{supertabular}{lp{\glsdescwidth}}}%
13331         {\end{supertabular}}%
13332     \renewcommand*{\glossaryheader}{}%
13333     \renewcommand*{\glsgroupheading}[1]{}%
13334     \renewcommand{\glossentry}[2]{%
13335         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13336         \glossentrydesc{##1}\glspostdescription
13337         \glxtrprelocation ##2\tabularnewline
13338     }%
13339     \renewcommand{\subglossentry}[3]{%
13340         &
13341         \glssubentryitem{##2}%
13342         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13343         \glxtrprelocation ##3\tabularnewline
13344     }%
13345     \ifglsnogroupskip
13346         \renewcommand*{\glsgroupskip}{}%
13347     \else
13348         \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13349     \fi
13350 }
13351 }
13352 {}

```

Three column style:

```

13353 \ifcsdef{@glsstyle@super3col}
13354 {%
13355   \renewglossarystyle{super3col}{%
13356     \renewenvironment{theglossary}%
13357       {\tablehead{}\tabletail{}}%
13358       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13359     {\end{supertabular}}}%
13360   \renewcommand*{\glossaryheader}{}%
13361   \renewcommand*{\glsgroupheading}[1]{}%
13362   \renewcommand{\glossentry}[2]{%
13363     \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13364     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13365   }%
13366   \renewcommand{\subglossentry}[3]{%
13367     &
13368     \glssubentryitem{##2}%
13369     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13370     ##3\tabularnewline
13371   }%

13372   \ifglsgroupskip
13373     \renewcommand*{\glsgroupskip}{}%
13374   \else
13375     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13376   \fi
13377 }
13378 }
13379 {}

```

Four column styles:

```

13380 \ifcsdef{@glsstyle@super4col}
13381 {%
13382   \renewglossarystyle{super4col}{%
13383     \renewenvironment{theglossary}%
13384       {\tablehead{}\tabletail{}}%
13385       \begin{supertabular}{llll}}}%
13386     {\end{supertabular}}}%
13387   \renewcommand*{\glossaryheader}{}%
13388   \renewcommand*{\glsgroupheading}[1]{}%
13389   \renewcommand{\glossentry}[2]{%
13390     \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13391     \glossentrydesc{##1}\glspostdescription &
13392     \glossentrysymbol{##1} & ##2\tabularnewline
13393   }%
13394   \renewcommand{\subglossentry}[3]{%
13395     &
13396     \glssubentryitem{##2}%
13397     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13398     \glossentrysymbol{##2} & ##3\tabularnewline

```

```

13399 }%
13400 \ifglsgroupskip
13401 \renewcommand*{\glsgroupskip}{}%
13402 \else
13403 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13404 \fi
13405 }
13406 }
13407 {}

```

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`.

```

13408 \ifcsdef{@glstyle@superragged}
13409 {%
13410 \renewglossarystyle{superragged}{%
13411 \renewenvironment{theglossary}%
13412 {\tablehead{}\tabletail}%
13413 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
13414 {\end{supertabular}}%
13415 \renewcommand*{\glossaryheader}{}%
13416 \renewcommand*{\glsgroupheading}[1]{}%
13417 \renewcommand{\glossentry}[2]{%
13418 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13419 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
13420 \tabularnewline
13421 }%
13422 \renewcommand{\subglossentry}[3]{%
13423 &
13424 \glssubentryitem{##2}%
13425 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13426 \glstrprelocation ##3%
13427 \tabularnewline
13428 }%
13429 \ifglsgroupskip
13430 \renewcommand*{\glsgroupskip}{}%
13431 \else
13432 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13433 \fi
13434 }
13435 }
13436 {}

```

Three column style:

```

13437 \ifcsdef{@glstyle@superragged3col}
13438 {%

```

```

13439 \renewglossarystyle{superragged3col}{%
13440   \renewenvironment{theglossary}%
13441     {\tablehead{}\tabletail{}}%
13442     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
13443       >{\raggedright}p{\glspagelistwidth}}}%
13444     {\end{supertabular}}%
13445   \renewcommand*{\glossaryheader}{}%
13446   \renewcommand*{\glsgroupheading}[1]{}%
13447   \renewcommand{\glossentry}[2]{%
13448     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13449     \glossentrydesc{##1}\glspostdescription &
13450     ##2\tabularnewline
13451   }%
13452   \renewcommand{\subglossentry}[3]{%
13453     &
13454     \glssubentryitem{##2}%
13455     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13456     ##3\tabularnewline
13457   }%

13458   \ifglsnogroupskip
13459     \renewcommand*{\glsgroupskip}{}%
13460   \else
13461     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13462   \fi
13463 }
13464 }
13465 {}

```

Four columns:

```

13466 \ifcsdef{@glsstyle@altsuperragged4col}
13467 {%
13468   \renewglossarystyle{altsuperragged4col}{%
13469     \renewenvironment{theglossary}%
13470       {\tablehead{}\tabletail{}}%
13471       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
13472         >{\raggedright}p{\glspagelistwidth}}}%
13473       {\end{supertabular}}%
13474     \renewcommand*{\glossaryheader}{}%
13475     \renewcommand{\glossentry}[2]{%
13476       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13477       \glossentrydesc{##1}\glspostdescription &
13478       \glossentrysymbol{##1} & ##2\tabularnewline
13479     }%
13480     \renewcommand{\subglossentry}[3]{%
13481       &
13482       \glssubentryitem{##2}%
13483       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13484       \glossentrysymbol{##2} & ##3\tabularnewline
13485     }%

```



```

13486 \ifglsnogroupskip
13487 \renewcommand*{\glsgroupskip}{}%
13488 \else
13489 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13490 \fi
13491 }
13492 }
13493 {}

```

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`.

```

13494 \ifdef{\@glsstyle@inline}
13495 {%
13496 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13497 \renewcommand*{\glsinlinedescformat}[3]{%
13498 \space#1\glxtrpostdescription}
13499 \renewcommand*{\glsinlinesubdescformat}[3]{%
13500 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13501 }
13502 {}

```

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.

```

13503 \ifdef\glstreenamefmt
13504 {

```

```

\glstreedefaultnamefmt

```

```

13505 \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}

```

```

\glstreenamefmt

```

```

13506 \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13507 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13508 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13509 }
13510 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13511 \ifdef{\@glsstyle@index}
13512 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13513 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13514 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13515 \renewglossarystyle{index}{%
13516   \renewenvironment{theglossary}%
13517     {\setlength{\parindent}{0pt}%
13518      \setlength{\parskip}{0pt plus 0.3pt}%
13519      \let\item\glstreeitem
13520      \let\subitem\glstreesubitem
13521      \let\subsubitem\glstreesubsubitem
13522     }%
13523   {\par}%
13524   \renewcommand*{\glossaryheader}{}%
13525   \renewcommand*{\glsgroupheading}[1]{}%
13526   \renewcommand*{\glossentry}[2]{%
13527     \item\glssentryitem{##1}%
13528     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13529     \glstreesymbol{##1}%
13530     \glstreedesc{##1}%
13531     \glstreeprelocation ##2%
13532   }%
13533   \renewcommand{\subglossentry}[3]{%
13534     \ifcase##1\relax
13535       \item
13536     \or
13537       \subitem
13538       \glssubentryitem{##2}%
13539     \else
13540       \subsubitem
13541     \fi
13542     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13543     \glstreechildsymbol{##2}%
13544     \glstreechilddesc{##2}%
13545     \glstreechildprelocation ##3%
13546   }%
```

```

13547 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
13548 }
13549 }
13550 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13551 \ifdef{\@glsstyle@indexgroup}
13552 {%
13553 \renewglossarystyle{indexgroup}{%
13554 \setglossarystyle{index}%
13555 \renewcommand*{\glsgroupheading}[1]{%
13556 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13557 \nopagebreak\indexspace
13558 \nobreak\@afterheading
13559 }%
13560 }
13561 }
13562 {}

```

Similarly for indexhypergroup.

```

13563 \ifdef{\@glsstyle@indexhypergroup}
13564 {%
13565 \renewglossarystyle{indexhypergroup}{%
13566 \setglossarystyle{index}%
13567 \renewcommand*{\glossaryheader}{%
13568 \item\glstreenavigationfmt{\glsnavigation}%
13569 \nobreak\@afterheading\indexspace}%
13570 \renewcommand*{\glsgroupheading}[1]{%
13571 \item\glstreegroupheaderfmt
13572 {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13573 \nopagebreak\indexspace
13574 \nobreak\@afterheading}%
13575 }%
13576 }
13577 {}

```

Adjust tree style to remove hard coded space before number list.

```

13578 \ifdef{\@glsstyle@tree}
13579 {%
13580 %Provide a command for use with the \glostyle{tree} styles that displays
13581 %the pre-description separator, the
13582 %description and post-description hook.
13583 %\begin{macro}{\glstreedesc}
13584 %\changes{1.31}{2018-05-09}{new}
13585 % \begin{macrocode}
13586 \newcommand{\glstreedesc}[1]{%
13587 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13588 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13589 \newcommand{\glstreesymbol}[1]{%
13590   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13591 }%
```

And for the child entries:

`lstreechilddesc`

```
13592 \newcommand{\glstreechilddesc}[1]{%
13593   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13594 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13595 \newcommand{\glstreechildsymbol}[1]{%
13596   \glstreesymbol{#1}%
13597 }%

13598 \renewglossarystyle{tree}{%
13599   \renewenvironment{theglossary}%
13600     {\setlength{\parindent}{0pt}%
13601     \setlength{\parskip}{0pt plus 0.3pt}}%
13602   {%
13603     \renewcommand*{\glossaryheader}{}%
13604     \renewcommand*{\glsgroupheading}[1]{}%
13605     \renewcommand{\glossentry}[2]{%
13606       \hangindent0pt\relax
13607       \parindent0pt\relax
13608       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13609       \glstreesymbol{##1}%
13610       \glstreedesc{##1}%
13611       \glstreeprelocation##2\par
13612     }%
13613     \renewcommand{\subglossentry}[3]{%
13614       \hangindent##1\glstreeindent\relax
13615       \parindent##1\glstreeindent\relax
13616       \ifnum##1=1\relax
13617         \glssubentryitem{##2}%
13618       \fi
13619       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13620       \glstreechildsymbol{##2}%
13621       \glstreechilddesc{##2}%
13622       \glstreechildprelocation ##3\par
13623     }%
13624     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13625   }%
13626 }
13627 }
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13628 \ifdef{\@glstyle@treegroup}
```

```

13629 {%
13630   \renewglossarystyle{treegroup}{%
13631     \setglossarystyle{tree}%
13632     \renewcommand{\glsgroupheading}[1]{\par
13633       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13634       \nopagebreak\indexspace\nobreak\@afterheading}%
13635   }
13636 }
13637 {}

```

Similarly for treehypergroup

```

13638 \ifdef{\@glsstyle@treehypergroup}
13639 {%
13640   \renewglossarystyle{treehypergroup}{%
13641     \setglossarystyle{tree}%
13642     \renewcommand*\{\glossaryheader}{%
13643       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13644       \nobreak\@afterheading\indexspace}%
13645     \renewcommand*\{\glsgroupheading}[1]{%
13646       \par\noindent
13647       \glstreegroupheaderfmt
13648       {\glstravigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
13649       \nopagebreak\indexspace\nobreak\@afterheading}%
13650   }
13651 }
13652 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13653 \ifdef{\@glsstyle@treenoname}
13654 {%
13655 %Provide a command for use with the \glsstyle{treenoname} styles that displays
13656 %the pre-description separator, the
13657 %description and post-description hook.
13658 %\begin{macro}{\glstreenonamedesc}
13659 %\changes{1.31}{2018-05-09}{new}
13660 %   \begin{macrocode}
13661   \newcommand{\glstreenonamedesc}[1]{%
13662     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13663   }%

```

Similarly for the symbol.

treenamesymbol

```

13664   \newcommand{\glstreenamesymbol}[1]{%
13665     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13666   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13667   \newcommand{\glstreenonamechilddesc}[1]{%
13668     \glossentrydesc{#1}\glspostdescription
13669   }%

```

```

13670 \renewglossarystyle{treenoname}{%
13671   \renewenvironment{theglossary}%
13672     {\setlength{\parindent}{0pt}%
13673      \setlength{\parskip}{0pt plus 0.3pt}}%
13674     {}%
13675   \renewcommand*{\glossaryheader}{}%
13676   \renewcommand*{\glsgroupheading}[1]{}%
13677   \renewcommand{\glossentry}[2]{%
13678     \hangindent0pt\relax
13679     \parindent0pt\relax
13680     \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13681     \glstreenonamesymbol{##1}%
13682     \glstreenonamedesc{##1}%
13683     \glstreeprelocation##2\par
13684   }%
13685   \renewcommand{\subglossentry}[3]{%
13686     \hangindent##1\glstreeindent\relax
13687     \parindent##1\glstreeindent\relax
13688     \ifnum##1=1\relax
13689       \glssubentryitem{##2}%
13690     \fi
13691     \glstarget{##2}{\strut}%
13692     \glstreenonamechilddesc{##2}%
13693     \glstreechildprelocation##3\par
13694   }%
13695   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13696 }
13697 }
13698 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13699 \ifdef{\@glstyle@treenonamegroup}
13700 {%
13701   \renewglossarystyle{treenonamegroup}{%
13702     \setglossarystyle{treenoname}%
13703     \renewcommand{\glsgroupheading}[1]{\par
13704       \noindent\glstreegroupheaderfmt
13705       {\glsggetgrouptitle{##1}}}%
13706     \nopagebreak\indexspace\nobreak\@afterheading
13707   }%
13708 }
13709 }
13710 {}

```

Similarly for `treenonamehypergroup`

```

13711 \ifdef{\@glstyle@treenonamehypergroup}
13712 {%
13713   \renewglossarystyle{treenonamehypergroup}{%
13714     \setglossarystyle{treenoname}%
13715     \renewcommand*{\glossaryheader}{%

```

```

13716      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13717      \nobreak\@afterheading\indexspace}%
13718      \renewcommand*{\glsgroupheading}[1]{%
13719      \par\noindent
13720      \glstreegroupheaderfmt
13721      {\glsnavhypertarget{##1}{\glsgroupgetgrouptitle{##1}}}%
13722      \nopagebreak\indexspace\nobreak\@afterheading}%
13723  }
13724 }
13725 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13726 \ifdef{\glstyle@almtree}
13727 {%

```

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.

```

13728 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13729   {%
13730     \let\par\glxtrAltTreePar
13731     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13732     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13733   }%
13734 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13735 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13736 \newcommand{\glxtrAltTreePar}{%
13737   \@@par
13738   \glxtrAltTreeSetHangIndent
13739   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13740 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13741 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13742   \glxtralttreeSymbolDescLocation{#2}{#3}%
13743 }

```

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.

```
13744 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13745 \newcommand*\glxstralttreeInit{%
13746   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
13747   \glxstrAltTreeIndent=\parindent
13748 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13749 \newcommand*\gglsetwidest}[2][0]{%
13750   \csgdef{@glswidestname\romannumeral#1}{#2}%
13751 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13752 \newcommand*\eglssetwidest}[2][0]{%
13753   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13754 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13755 \newcommand*\xglsetwidest}[2][0]{%
13756   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13757 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13758 \newcommand*\glupdatewidest}[2][0]{%
13759   \ifcsundef{@glswidestname\romannumeral#1}%
13760   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13761   {%
13762     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13763     \settowidth{\dimen@ii}{#2}%
13764     \ifdim\dimen@ii>\dimen@
13765       \csdef{@glswidestname\romannumeral#1}{#2}%
13766     \fi
13767   }%
13768 }
```

glupdatewidest As above but global definition.

```
13769 \newcommand*\gglupdatewidest}[2][0]{%
13770   \ifcsundef{@glswidestname\romannumeral#1}%
13771   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13772   {%
13773     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13774     \settowidth{\dimen@ii}{#2}%
13775     \ifdim\dimen@ii>\dimen@
13776       \csgdef{@glswidestname\romannumeral#1}{#2}%
13777     \fi
```



```

13778 }%
13779 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13780 \newcommand*{\eglsupdatewidest}[2][0]{%
13781   \ifcsundef{@glswidestname\romannumeral#1}%
13782   {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13783   {%
13784     \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13785     \settowidth{\dimen@ii}{#2}%
13786     \ifdim\dimen@ii>\dimen0
13787       \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13788     \fi
13789   }%
13790 }

```

`glsupdatewidest` As above but global.

```

13791 \newcommand*{\xglsupdatewidest}[2][0]{%
13792   \ifcsundef{@glswidestname\romannumeral#1}%
13793   {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13794   {%
13795     \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13796     \settowidth{\dimen@ii}{#2}%
13797     \ifdim\dimen@ii>\dimen0
13798       \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13799     \fi
13800   }%
13801 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13802 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13803 \newcommand*{\glsgetwidestsubname}[1]{%
13804   \ifcsundef{@glswidestname\romannumeral#1}%
13805   {\@glswidestname}%
13806   {\csuse{@glswidestname\romannumeral#1}}%
13807 }

```

`glswidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsgfindwidesttoplevelname`

```

13808 \let\glsgFindWidestTopLevelName\glsgfindwidesttoplevelname

```

`glswidestUsedTopLevelName` Like `\glsgfindwidesttoplevelname` 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.

```

13809 \newrobustcmd*{\glsgFindWidestUsedTopLevelName}[1][\@glo@types]{%
13810   \dimen@=0pt\relax

```

```

13811 \gls@tmplen=Opt\relax
13812 \foralllglossaries[#1]{\@gls@type}%
13813 {%
13814 \forglsentries[\@gls@type]{\@glo@label}%
13815 {%
13816 \ifglsused{\@glo@label}%
13817 {%
13818 \ifglsashasparent{\@glo@label}%
13819 {}%
13820 {%
13821 \settowidth{\dimen@}%
13822 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13823 \ifdim\dimen@>\gls@tmplen
13824 \gls@tmplen=\dimen@
13825 \eglssetwidest{\glsentryname{\@glo@label}}%
13826 \fi
13827 }%
13828 }%
13829 {}%
13830 }%
13831 }%
13832 }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13833 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13834 \dimen@=Opt\relax
13835 \gls@tmplen=Opt\relax
13836 \foralllglossaries[#1]{\@gls@type}%
13837 {%
13838 \forglsentries[\@gls@type]{\@glo@label}%
13839 {%
13840 \ifglsused{\@glo@label}%
13841 {%
13842 \settowidth{\dimen@}%
13843 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13844 \ifdim\dimen@>\gls@tmplen
13845 \gls@tmplen=\dimen@
13846 \eglssetwidest{\glsentryname{\@glo@label}}%
13847 \fi
13848 }%
13849 {}%
13850 }%
13851 }%
13852 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

13853 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13854 \dimen@=Opt\relax

```

```

13855 \gls@tmplen=0pt\relax
13856 \foralllglossaries[#1]{\@gls@type}%
13857 {%
13858   \forglsentries[\@gls@type]{\@glo@label}%
13859   {%
13860     \settowidth{\dimen@}%
13861     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13862     \ifdim\dimen@>\gls@tmplen
13863       \gls@tmplen=\dimen@
13864       \eglssetwidest{\glsentryname{\@glo@label}}%
13865     \fi
13866   }%
13867 }%
13868 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.
Any entry that has a great-grandparent is ignored.

```

13869 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13870   \dimen@=0pt\relax
13871   \dimen@i=0pt\relax
13872   \dimen@ii=0pt\relax
13873   \foralllglossaries[#1]{\@gls@type}%
13874   {%
13875     \forglsentries[\@gls@type]{\@glo@label}%
13876     {%
13877       \ifglsused{\@glo@label}%
13878       {%
13879         \ifglshasparent{\@glo@label}%
13880         {%
13881           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
13882           \ifglshasparent{\@glo@parent}%
13883           {%
13884             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
13885             \ifglshasparent{\@glo@parent}%
13886             {}%
13887             {%
13888               \settowidth{\gls@tmplen}%
13889               {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13890               \ifdim\gls@tmplen>\dimen@ii
13891                 \dimen@ii=\gls@tmplen
13892                 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13893               \fi
13894             }%
13895           }%
13896         }%
13897         \settowidth{\gls@tmplen}%
13898         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13899         \ifdim\gls@tmplen>\dimen@i
13900           \dimen@i=\gls@tmplen

```

```

13901         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13902     \fi
13903 }%
13904 }%
13905 {%
13906     \settowidth{\gls@tmplen}%
13907         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13908     \ifdim\gls@tmplen>\dimen@
13909         \dimen@=\gls@tmplen
13910         \eglssetwidest{\glsentryname{\@glo@label}}%
13911     \fi
13912 }%
13913 }%
13914 {}%
13915 }%
13916 }%
13917 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13918 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13919     \dimen@=0pt\relax
13920     \dimen@i=0pt\relax
13921     \dimen@ii=0pt\relax
13922     \foralllglossaries[#1]{\@gls@type}%
13923     {%
13924         \forglsentries[\@gls@type]{\@glo@label}%
13925         {%
13926             \ifglshasparent{\@glo@label}%
13927             {%
13928                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13929                 \ifglshasparent{\@glo@parent}%
13930                 {%
13931                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13932                     \ifglshasparent{\@glo@parent}%
13933                     {}%
13934                 }%
13935                 \settowidth{\gls@tmplen}%
13936                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13937                 \ifdim\gls@tmplen>\dimen@ii
13938                     \dimen@ii=\gls@tmplen
13939                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13940                 \fi
13941             }%
13942         }%
13943     }%
13944     \settowidth{\gls@tmplen}%
13945         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13946     \ifdim\gls@tmplen>\dimen@i
13947         \dimen@i=\gls@tmplen

```

```

13948         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13949     \fi
13950 }%
13951 }%
13952 {%
13953     \settowidth{\gls@tmplen}%
13954     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13955     \ifdim\gls@tmplen>\dimen@
13956         \dimen@=\gls@tmplen
13957         \eglssetwidest{\glsentryname{\@glo@label}}%
13958     \fi
13959 }%
13960 }%
13961 }%
13962 }

```

`\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.

```

13963 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13964     \dimen@=0pt\relax
13965     \gls@tmplen=0pt\relax
13966     #2=0pt\relax
13967     \foralllglossaries[#1]{\@gls@type}%
13968     {%
13969         \forglsentries[\@gls@type]{\@glo@label}%
13970         {%
13971             \ifglsused{\@glo@label}%
13972             {%
13973                 \settowidth{\dimen@}%
13974                 {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13975                 \ifdim\dimen@>\gls@tmplen
13976                     \gls@tmplen=\dimen@
13977                     \eglssetwidest{\glsentryname{\@glo@label}}%
13978                 \fi
13979                 \settowidth{\dimen@}%
13980                 {\glsentrysymbol{\@glo@label}}%
13981                 \ifdim\dimen@>#2\relax
13982                     #2=\dimen@
13983                 \fi
13984             }%
13985         }%
13986     }%
13987 }%
13988 }

```

`\stAnyNameSymbol` Like the above but doesn't check if the entry has been used.

```

13989 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
13990     \dimen@=0pt\relax
13991     \gls@tmplen=0pt\relax

```

```

13992      #2=Opt\relax
13993      \forallglossaries[#1]{\@gls@type}%
13994      {%
13995        \forglsentries[\@gls@type]{\@glo@label}%
13996        {%
13997          \settowidth{\dimen@}%
13998            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13999          \ifdim\dimen@>\gls@tmplen
14000            \gls@tmplen=\dimen@
14001            \eglssetwidest{\glsentryname{\@glo@label}}%
14002          \fi
14003          \settowidth{\dimen@}%
14004            {\glsentrysymbol{\@glo@label}}%
14005          \ifdim\dimen@>#2\relax
14006            #2=\dimen@
14007          \fi
14008        }%
14009      }%
14010    }

```

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.

```

14011    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14012      \dimen@=Opt\relax
14013      \gls@tmplen=Opt\relax
14014      #2=Opt\relax
14015      #3=Opt\relax
14016      \forallglossaries[#1]{\@gls@type}%
14017      {%
14018        \forglsentries[\@gls@type]{\@glo@label}%
14019        {%
14020          \ifglsused{\@glo@label}%
14021          {%
14022            \settowidth{\dimen@}%
14023              {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14024            \ifdim\dimen@>\gls@tmplen
14025              \gls@tmplen=\dimen@
14026              \eglssetwidest{\glsentryname{\@glo@label}}%
14027            \fi
14028            \settowidth{\dimen@}%
14029              {\glsentrysymbol{\@glo@label}}%
14030            \ifdim\dimen@>#2\relax
14031              #2=\dimen@
14032            \fi
14033            \settowidth{\dimen@}%
14034              {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14035            \ifdim\dimen@>#3\relax

```

```

14036         #3=\dimen@
14037     \fi
14038 }%
14039 {}%
14040 }%
14041 }%
14042 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14043 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14044     \dimen@=0pt\relax
14045     \gls@tmplen=0pt\relax
14046     #2=0pt\relax
14047     #3=0pt\relax
14048     \forallglossaries[#1]{\@gls@type}%
14049     {%
14050         \forglsentries[\@gls@type]{\@glo@label}%
14051         {%
14052             \settowidth{\dimen@}%
14053             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14054             \ifdim\dimen@>\gls@tmplen
14055                 \gls@tmplen=\dimen@
14056                 \eglssetwidest{\glsentryname{\@glo@label}}%
14057             \fi
14058             \settowidth{\dimen@}%
14059             {\glsentrysymbol{\@glo@label}}%
14060             \ifdim\dimen@>#2\relax
14061                 #2=\dimen@
14062             \fi
14063             \settowidth{\dimen@}%
14064             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14065             \ifdim\dimen@>#3\relax
14066                 #3=\dimen@
14067             \fi
14068         }%
14069     }%
14070 }

```

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.

```

14071 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14072     \dimen@=0pt\relax
14073     \gls@tmplen=0pt\relax
14074     #2=0pt\relax
14075     \forallglossaries[#1]{\@gls@type}%
14076     {%
14077         \forglsentries[\@gls@type]{\@glo@label}%
14078         {%

```

```

14079      \ifglsused{\@glo@label}%
14080      {%
14081        \settowidth{\dimen@}%
14082        {\glstreenamefmt{\glentryname{\@glo@label}}}%
14083        \ifdim\dimen@>\gls@tmplen
14084          \gls@tmplen=\dimen@
14085          \eglssetwidest{\glentryname{\@glo@label}}%
14086        \fi
14087        \settowidth{\dimen@}%
14088        {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14089        \ifdim\dimen@>#2\relax
14090          #2=\dimen@
14091        \fi
14092      }%
14093    {}%
14094  }%
14095 }%
14096 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14097 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14098   \dimen@=0pt\relax
14099   \gls@tmplen=0pt\relax
14100   #2=0pt\relax
14101   \forallglossaries[#1]{\@gls@type}%
14102   {%
14103     \forglsentries[\@gls@type]{\@glo@label}%
14104     {%
14105       \settowidth{\dimen@}%
14106       {\glstreenamefmt{\glentryname{\@glo@label}}}%
14107       \ifdim\dimen@>\gls@tmplen
14108         \gls@tmplen=\dimen@
14109         \eglssetwidest{\glentryname{\@glo@label}}%
14110       \fi
14111       \settowidth{\dimen@}%
14112       {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14113       \ifdim\dimen@>#2\relax
14114         #2=\dimen@
14115       \fi
14116     }%
14117   }%
14118 }

```

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`.

```

14119 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
14120   \glstreeindent=\glsxtrtreetopindent\relax
14121 }

```


uteTreeSubIndent

```
\glxstrComputeTreeSubIndent{<level>}{<label>}{<register>}
```

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
14122 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14123 \ifcsundef{@glswidestname\romannumeral#1}%
14124 {%
14125 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14126 }%
14127 {%
14128 \settowidth{#3}{\glstreenamefmt{
14129 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14130 }%
14131 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14132 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14133 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14134 \renewglossarystyle{alttree}{%
14135 \renewenvironment{theglossary}%
14136 {%
14137 \glxstralttreeInit
14138 \def{@glsprevlevel}{-1}%
14139 \mbox{}\par}%
14140 {\par}%
14141 \renewcommand*{\glossaryheader}{}%
14142 \renewcommand*{\glsgroupheading}[1]{}%
14143 \renewcommand{\glossentry}[2]{%
14144 \ifnum{@glsprevlevel}=0\relax
14145 \else
14146 \glxstrComputeTreeIndent{##1}%
14147 \fi
14148 \parindent\glstreeindent
14149 \glxstrAltTreeSetHangIndent
14150 \makebox[Opt][r]%
14151 {%
14152 \glstreenamebox{\glstreeindent}%
14153 {%
14154 \glsenryitem{##1}%
14155 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14156 }%
14157 }%
```

```

14158     \glxtralttreeSymbolDescLocation{##1}{##2}%
14159     \def\@gls@prevlevel{0}%
14160 }
14161 \renewcommand{\subglossentry}[3]{%
14162     \ifnum##1=1\relax
14163         \glssubentryitem{##2}%
14164     \fi
14165     \ifnum\@gls@prevlevel=##1\relax
14166     \else
14167         \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14168         \ifnum\@gls@prevlevel<##1\relax
14169             \setlength\glstreeindent\gls@tmplen
14170             \addtolength\glstreeindent\parindent
14171             \parindent\glstreeindent
14172         \else
14173             \ifnum\@gls@prevlevel=0\relax
14174                 \glxtrComputeTreeIndent{##2}%
14175             \else
14176                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14177             \fi
14178             \addtolength\parindent{-\glstreeindent}%
14179             \setlength\glstreeindent\parindent
14180         \fi
14181     \fi
14182     \glxtrAltTreeSetSubHangIndent{##1}%
14183     \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14184         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14185     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14186     \def\@gls@prevlevel{##1}%
14187 }%
14188 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
14189 }
14190 }%
14191 {%
14192 }

```

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.

```

14193 \ifdef{\@glsstyle@alttreegroup}
14194 {%
14195     \renewglossarystyle{alttreegroup}{%
14196         \setglossarystyle{alttree}%
14197         \renewcommand{\glsgroupheading}[1]{\par
14198             \def\@gls@prevlevel{-1}%
14199             \hangindent0pt\relax
14200             \parindent0pt\relax
14201             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14202         \nopagebreak\indexspace\nopagebreak
14203     }%
14204 }%

```

```

14205 }%
14206 {%
14207 }

```

Similarly for `alttreehypergroup`.

```

14208 \ifdef{\@glsstyle@alttreehypergroup}
14209 {%
14210   \renewglossarystyle{alttreehypergroup}{%
14211     \setglossarystyle{alttree}%
14212     \renewcommand*{\glossaryheader}{%
14213       \par
14214       \def\@gls@prevlevel{-1}%
14215       \hangindent0pt\relax
14216       \parindent0pt\relax
14217       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14218     }%
14219     \renewcommand*{\glsgroupheading}[1]{%
14220       \par
14221       \def\@gls@prevlevel{-1}%
14222       \hangindent0pt\relax
14223       \parindent0pt\relax
14224       \glstreegroupheaderfmt
14225         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14226       \nopagebreak\indexspace\nopagebreak
14227     }%
14228   }
14229 }%
14230 {%
14231 }

```

2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14232 \ifdef{\@glsstyle@mcolindexgroup}
14233 {%
14234   \renewglossarystyle{mcolindexgroup}{%
14235     \setglossarystyle{mcolindex}%
14236     \renewcommand*{\glsgroupheading}[1]{%
14237       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14238       \nopagebreak\indexspace\nobreak\@afterheading
14239     }%
14240   }
14241 }%
14242 {%
14243 }

```

Similarly for `mcolindexhypergroup`.

```

14244 \ifdef{\@glsstyle@mcolindexhypergroup}
14245 {%

```

```

14246 \renewglossarystyle{mcolindexhypergroup}{%
14247   \setglossarystyle{mcolindex}%
14248   \renewcommand*{\glossaryheader}{%
14249     \item\glstreenavigationfmt{\glsnavigation}%
14250     \indexspace
14251   }%
14252   \renewcommand*{\glsgroupheading}[1]{%
14253     \item\glstreegroupheaderfmt
14254       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14255     \nopagebreak\indexspace\nobreak\@afterheading
14256   }%
14257 }
14258 }%
14259 {%
14260 }

```

Similarly for mcolindexspannav.

```

14261 \ifdef{\@glsstyle@mcolindexspannav}
14262 {%
14263   \renewglossarystyle{mcolindexspannav}{%
14264     \setglossarystyle{index}%
14265     \renewenvironment{theglossary}%
14266     {%
14267       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14268       \setlength{\parindent}{0pt}%
14269       \setlength{\parskip}{0pt plus 0.3pt}%
14270       \let\item\glstreeitem}%
14271     {\end{multicols}}}%
14272   \renewcommand*{\glsgroupheading}[1]{%
14273     \item\glstreegroupheaderfmt
14274       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14275     \nopagebreak\indexspace\nobreak\@afterheading
14276   }%
14277 }
14278 }%
14279 {%
14280 }

```

Similarly for mcoltreegroup.

```

14281 \ifdef{\@glsstyle@mcoltreegroup}
14282 {%
14283   \renewglossarystyle{mcoltreegroup}{%
14284     \setglossarystyle{mcoltree}%
14285     \renewcommand{\glsgroupheading}[1]{\par
14286       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14287     \nopagebreak\indexspace\nobreak\@afterheading
14288   }%
14289 }
14290 }%
14291 {%

```

14292 }

Similarly for mcoltreehypergroup.

```
14293 \ifdef{\@glsstyle@mcoltreehypergroup}
14294 {%
14295   \renewglossarystyle{mcoltreehypergroup}{%
14296     \setglossarystyle{mcoltree}%
14297     \renewcommand*{\glossaryheader}{%
14298       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14299     }%
14300     \renewcommand*{\glsgroupheading}[1]{%
14301       \par\noindent
14302       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14303       \nopagebreak\indexspace\nobreak\@afterheading
14304     }%
14305   }
14306 }%
14307 {%
14308 }
```

Similarly for mcoltreespannav.

```
14309 \ifdef{\@glsstyle@mcoltreespannav}
14310 {%
14311   \renewglossarystyle{mcoltreespannav}{%
14312     \setglossarystyle{tree}%
14313     \renewenvironment{theglossary}%
14314     {%
14315       \begin{multicols}{\glsmcols}%
14316       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14317       \setlength{\parindent}{0pt}%
14318       \setlength{\parskip}{0pt plus 0.3pt}%
14319     }%
14320     {\end{multicols}}%
14321     \renewcommand*{\glsgroupheading}[1]{%
14322       \par\noindent
14323       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14324       \nopagebreak\indexspace\nobreak\@afterheading
14325     }%
14326   }
14327 }%
14328 {%
14329 }
```

Similarly for mcoltreenonamegroup.

```
14330 \ifdef{\@glsstyle@mcoltreenonamegroup}
14331 {%
14332   \renewglossarystyle{mcoltreenonamegroup}{%
14333     \setglossarystyle{mcoltreenoname}%
14334     \renewcommand{\glsgroupheading}[1]{\par
14335       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14336     \nopagebreak\indexspace\nobreak\@afterheading
14337   }
```

```

14337 }%
14338 }
14339 }%
14340 {%
14341 }

```

Similarly for mcoltreenonamehypergroup.

```

14342 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14343 {%
14344   \renewglossarystyle{mcoltreenonamehypergroup}{%
14345     \setglossarystyle{mcoltreenoname}%
14346     \renewcommand*\glossaryheader{%
14347       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14348     \renewcommand*\glsgroupheading}[1]{%
14349       \par\noindent
14350       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14351       \nopagebreak\indexspace\nobreak\@afterheading}%
14352   }
14353 }%
14354 {%
14355 }

```

Similarly for mcoltreenonamespannav.

```

14356 \ifdef{\@glsstyle@mcoltreenonamespannav}
14357 {%
14358   \renewglossarystyle{mcoltreenonamespannav}{%
14359     \setglossarystyle{treenoname}%
14360     \renewenvironment{theglossary}%
14361     {%
14362       \begin{multicols}{\glsmcols}%
14363       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14364       \setlength{\parindent}{0pt}%
14365       \setlength{\parskip}{0pt plus 0.3pt}%
14366     }%
14367     {\end{multicols}}}%
14368   \renewcommand*\glsgroupheading}[1]{%
14369     \par\noindent
14370     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14371     \nopagebreak\indexspace\nobreak\@afterheading}%
14372   }
14373 }%
14374 {%
14375 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{}\par which would unbalance the top of the columns.

```

14376 \ifdef{\@glsstyle@mcolalttree}
14377 {%
14378   \renewglossarystyle{mcolalttree}{%
14379     \setglossarystyle{alttree}%
14380     \renewenvironment{theglossary}%

```

```

14381   {%
14382       \glstralttreeInit
14383       \def\@gls@prevlevel{-1}%
14384       \begin{multicols}{\glsmcols}%
14385   }%
14386   {\par\end{multicols}}%
14387 }
14388 }%
14389 {%
14390 }

```

Redefine mcolalttreegroup to discourage page breaks after the group headings.

```

14391 \ifdef{\@glsstyle@mcolalttreegroup}
14392 {%
14393   \renewglossarystyle{mcolalttreegroup}{%
14394       \setglossarystyle{mcolalttree}%
14395       \renewcommand{\glsgroupheading}[1]{\par
14396           \def\@gls@prevlevel{-1}%
14397           \hangindent0pt\relax
14398           \parindent0pt\relax
14399           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14400       \nopagebreak\indexspace\nopagebreak
14401   }%
14402 }
14403 }%
14404 {%
14405 }

```

Similarly for mcolalttreehypergroup.

```

14406 \ifdef{\@glsstyle@mcolalttreehypergroup}
14407 {%
14408   \renewglossarystyle{mcolalttreehypergroup}{%
14409       \setglossarystyle{mcolalttree}%
14410       \renewcommand*\glossaryheader{%
14411           \par
14412           \def\@gls@prevlevel{-1}%
14413           \hangindent0pt\relax
14414           \parindent0pt\relax
14415           \glstreenavigationfmt{\glsnavigation}%
14416           \par\indexspace
14417       }%
14418       \renewcommand*\glsgroupheading[1]{%
14419           \par
14420           \def\@gls@prevlevel{-1}%
14421           \hangindent0pt\relax
14422           \parindent0pt\relax
14423           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14424           \nopagebreak\indexspace\nopagebreak
14425       }%
14426   }

```

```

14427 }%
14428 {%
14429 }

```

Similarly for mcolalttreespannav.

```

14430 \ifdef{\@glsstyle@mcolalttreespannav}
14431 {%
14432   \renewglossarystyle{mcolalttreespannav}{%
14433     \setglossarystyle{alttree}%
14434     \renewenvironment{theglossary}%
14435     {%
14436       \glsextralttreeInit
14437       \def\@gls@prevlevel{-1}%
14438       \begin{multicols}{\glsmcols}%
14439         [\noindent\glstreenavigationfmt{\glsnavigation}]%
14440     }%
14441     {\par\end{multicols}}%
14442     \renewcommand*{\glsgroupheading}[1]{%
14443       \par
14444       \def\@gls@prevlevel{-1}%
14445       \hangindent0pt\relax
14446       \parindent0pt\relax
14447       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14448       \nopagebreak\indexspace\nopagebreak
14449     }%
14450   }
14451 }%
14452 {%
14453 }

```

Reset the default style

```

14454 \ifx\@glossary@default@style\relax
14455 \else
14456   \setglossarystyle{\@glsextr@current@style}
14457 \fi

```


3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
14458 \NeedsTeXFormat{LaTeX2e}
14459 \ProvidesPackage{glossary-bookindex}[2018/07/26 v1.33 (NLCT)]
```

Load required packages.

```
14460 \RequirePackage{multicol}
14461 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14462 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14463 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14464 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14465 \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.

```
14466 \newcommand*{\glstrbookindexprelocation}[1]{%
14467   \glstrifhasfield{location}{#1}%
14468   {,\glstrprelocation}%
14469   {\glstrprelocation}%
14470 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14471 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14472   \glstrbookindexprelocation{#1}%
14473 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14474 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14475 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
14476 \newcommand{\glxstrbookindexbetween}[2]{}

indexsubbetween Between two level 1 entries identified by the labels in the arguments.
14477 \newcommand{\glxstrbookindexsubbetween}[2]{}

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.
14478 \newcommand{\glxstrbookindexsubsubbetween}[2]{}

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.
14479 \newcommand{\glxstrbookindexatendgroup}[1]{}

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.
14480 \newcommand{\glxstrbookindexsubatendgroup}[1]{}

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.
14481 \newcommand{\glxstrbookindexsubsubatendgroup}[1]{}

kindexgroupskip Group separator.
14482 \newcommand{\glxstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

Format group title.

dexformatheader Group separator.
14483 \newcommand*{\glxstrbookindexformatheader}[1]{%
14484 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14485 }

bookindexbookmark Book mark group heading if supported.
14486 \ifdef\pdfbookmark
14487 {%
14488 \newcommand*{\glxstrbookindexbookmark}[2]{%
14489 \ifdefstring{\@@glossarysec}{chapter}%
14490 {\pdfbookmark[1]{#1}{#2}}%
14491 {\pdfbookmark[2]{#1}{#2}}%
14492 }
14493 }
14494 {%
14495 \newcommand*{\glxstrbookindexbookmark}[2]{}
14496 }

kindexcolspread
14497 \newcommand*{\glxstrbookindexcolspread}{}

dexmulticolseenv
14498 \newcommand*{\glxstrbookindexmulticolseenv}{multicols}
```

Define the style.

```

14499 \newglossarystyle{bookindex}{%
14500   \setglossarystyle{index}%
14501   \renewenvironment{theglossary}%
14502   {%
14503     \ifdefempty{glstrbookindexcolspread}
14504     {%
14505       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14506       {\glstrbookindexcols}%
14507     }%
14508     {%
14509       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14510       {\glstrbookindexcols}[\glstrbookindexcolspread]%
14511     }%
14512     \setlength{\parindent}{0pt}%
14513     \setlength{\parskip}{0pt plus 0.3pt}%
14514     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14515     \let\@glstrbookindex@subsep\glstrbookindexparentsubchildsep
14516     \let\@glstrbookindex@between\@gobble
14517     \let\@glstrbookindex@subbetween\@gobble
14518     \let\@glstrbookindex@subsubbetween\@gobble
14519     \let\@glstrbookindex@atendgroup\relax
14520     \let\@glstrbookindex@subatendgroup\relax
14521     \let\@glstrbookindex@subsubatendgroup\relax
14522     \let\@glstrbookindexgroupskip\relax
14523   }%
14524   {%

```

Do end group hooks.

```

14525     \@glstrbookindex@subsubatendgroup
14526     \@glstrbookindex@subatendgroup
14527     \@glstrbookindex@atendgroup

```

End multicol environment.

```

14528     \expandafter\end\expandafter{\glstrbookindexmulticolseenv}%
14529   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

14530   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

14531   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

14532     \@glstrbookindex@between{##1}%

```

Update separators.

```

14533     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14534     \let\@glstrbookindex@subsep\glstrbookindexparentsubchildsep
14535     \let\@glstrbookindex@subbetween\@gobble
14536     \let\@glstrbookindex@subsubbetween\@gobble
14537     \edef\@glstrbookindex@between{%

```

```

14538     \noexpand\glxstrbookindexbetween{##1}%
14539 }%
14540 \edef\@glxstr@bookindex@atendgroup{%
14541     \noexpand\glxstrbookindexatendgroup{##1}%
14542 }%
14543 \let\@glxstr@bookindex@subatendgroup\relax
14544 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14545 \glstreeitem
14546 \glstryitem{##1}%
14547 \glstarget{##1}{\glxstrbookindexname{##1}}%
14548 \glxstrbookindexprelocation{##1}##2%
14549 }%
14550 \renewcommand{\subglossentry}[3]{%
14551 \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14552 \glstreeitem
14553 \or

```

Level 1.

```

14554 \@glxstr@bookindex@sep
14555 \@glxstr@bookindex@subbetween{##2}%
14556 \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14557 \let\@glxstr@bookindex@subsubbetween\@gobble
14558 \let\@glxstr@bookindex@subsep\glxstrbookindexparentsubchildsep
14559 \edef\@glxstr@bookindex@subbetween{%
14560     \noexpand\glxstrbookindexsubbetween{##2}%
14561 }%
14562 \edef\@glxstr@bookindex@atsubendgroup{%
14563     \noexpand\glxstrbookindexatsubendgroup{##1}%
14564 }%

```

Start sub-item.

```

14565 \glstreesubitem
14566 \glssubentryitem{##2}%
14567 \else

```

All other levels.

```

14568 \@glxstr@bookindex@subsep
14569 \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14570 \let\@glxstr@bookindex@subsep\relax
14571 \edef\@glxstr@bookindex@subsubbetween{%
14572     \noexpand\glxstrbookindexsubsubbetween{##2}%
14573 }%
14574 \edef\@glxstr@bookindex@atsubsubendgroup{%
14575     \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14576 }%

```

Start sub-sub-item.

```
14577 \glstreesubsubitem
14578 \fi
```

Format entry.

```
14579 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14580 \glxtrbookindexsubprelocation{##2}##3%
14581 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14582 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14583 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14584 \@glxtr@bookindex@subsubatendgroup
14585 \@glxtr@bookindex@subatendgroup
14586 \@glxtr@bookindex@atendgroup
14587 \@glxtr@bookindex@groupskip
```

Update separators.

```
14588 \let\@glxtr@bookindex@groupskip\glxtrbookindexgroupskip
14589 \let\@glxtr@bookindex@between\@gobble
14590 \let\@glxtr@bookindex@atendgroup\relax
14591 \let\@glxtr@bookindex@subatendgroup\relax
14592 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14593 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14594 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14595 \glxtrbookindexformatheader{\thisgrptitle}%
14596 \nopagebreak\indexspace\nopagebreak\@afterheading
14597 }%
14598 }
```

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.

```
14599 \newcommand{\glxtrbookindexthepage}{%
14600 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14601 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14602 \newcommand*{\glxtrbookindexmarkentry}[1]{%
14603   \protected@write\@auxout
14604   {\let\glxtrbookindexthepage\relax}%
14605   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14606 }
```

etbookindexmark

```
14607 \newcommand*{\glxtr@setbookindexmark}[2]{%
14608   \ifcsundef{glxtr@idxfirstmark@#1}%
14609   {\csgdef{glxtr@idxfirstmark@#1}{#2}}%
14610   {}%
14611   \csgdef{glxtr@idxlastmark@#1}{#2}%
14612 }
```

indexfirstmarkfmt

```
14613 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
14614   \glstryname{#1}%
14615 }
```

indexfirstmark

```
14616 \newcommand*{\glxtrbookindexfirstmark}{%
14617   \letcs{glxtr@label}{glxtr@idxfirstmark@glxtrbookindexthepage}%
14618   \ifdefglxtr@label
14619   {\glxtrbookindexfirstmarkfmt{glxtr@label}}%
14620   {}%
14621 }
```

indexlastmarkfmt

```
14622 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
14623   \glstryname{#1}%
14624 }
```

okindexlastmark

```
14625 \newcommand*{\glxtrbookindexlastmark}{%
14626   \letcs{glxtr@label}{glxtr@idxlastmark@glxtrbookindexthepage}%
14627   \ifdefglxtr@label
14628   {\glxtrbookindexlastmarkfmt{glxtr@label}}%
14629   {}%
14630 }
```

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	73
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	73
\Glsfmtshort: new	323	\@Glstext@: added redefinition	68
\glsfmtshort: new	322	\@Glsuseri@: added redefinition	73
\Glsfmtshortpl: new	323	\@Glsuserii@: added redefinition	74
\glsfmtshortpl: new	322	\@Glsuseriii@: added redefinition	74
short: switched inline full form to short		\@Glsuseriv@: added redefinition	74
(long)	225	\@Glsuserv@: added redefinition	75
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	75
\@ACRlong: added redefinition	78	\@acrlong: added redefinition	77
\@ACRlongpl: added redefinition	79	\@acrlongpl: added redefinition	78
\@ACRshort: added redefinition	76	\@acrshort: added redefinition	75
\@ACRshortpl: added redefinition	77	\@acrshortpl: added redefinition	76
\@Acrlong: added redefinition	78	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	79	argument	61
\@Acrshort: added redefinition	76	\@glsdescplural@: added redefinition	72
\@Acrshortpl: added redefinition	77	\@glsfirst@: added redefinition	69
\@GLSdesc: added redefinition	72	\@glsfirstplural@: added redefinition	70
\@GLSdescplural@: added redefinition	72	\@glsplural@: added redefinition	70
\@GLSfirst@: added redefinition	69	\@Glsymbolplural@: added	
\@GLSfirstplural@: added redefinition	71	redefinition	73
\@GLSname@: added redefinition	71	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	70	new	131
\@GLSsymbol@: added redefinition	73	\@glsxtr@field@linkdefs: new	68
\@GLSsymbolplural@: added		\@glsxtr@insertdots: new	193
redefinition	73	\@print@glossary: added redefinition	128
\@GLStext@: added redefinition	68	\@glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	74	\abbrvdefaultfont	199
\@GLSuserii@: added redefinition	74	\@glsaccessdesc: new	157
\@GLSuseriii@: added redefinition	74	\@glsaccessdescplural: new	157
\@GLSuseriv@: added redefinition	75	\@glsaccessfirst: new	154
\@GLSuserv@: added redefinition	75	\@glsaccessfirstplural: new	155
\@GLSuservi@: added redefinition	75	\@glsaccesslong: new	159
\@Glsdesc@: added redefinition	72	\@glsaccesslong: new	159
\@Glsdescplural@: added redefinition	72	\@glsaccessname: new	153
\@Glsfirst@: added redefinition	69	\@glsaccessplural: new	154
\@Glsfirstplural@: added redefinition	70	\@glsaccessshort: new	158
\@Glsname@: added redefinition	71	\@glsaccessshort: new	158
\@Glsplural@: added redefinition	70	\@glsaccessshortpl: new	159

\glsaccesssshortpl: new	159	\cGLSpl: new	105
\glsaccessssymbol: new	156	\cGLSpl@: new	105
\glsaccessssymbolplural: new	156	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	153	new	100
\glstentryfmt: added check for short ..	60	\cGLS: new	104
\glslongpltok: new	193	\cGLSformat: new	105
\glsshortpltok: new	193	\cGLSpl: new	105
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	105
name in \setkeys	195	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	16
for plural	190	\glsenableentrycount: new	100
\GLSxtrlongpl: new	209	\glsfirstabbrvdefaultfont: new ..	199
\Glsxtrlongpl: new	209	\glsfirstlongdefaultfont: new ...	199
\glsxtrlongpl: new	208	\Glsfmtfirst: new	325
\glsxtrNoGlossaryWarning: new	21	\glsfmtfirst: new	325
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	326
new	189	\glsfmtfirstpl: new	325
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	325
new	189	\glsfmtplural: new	324
\glsxtrpostlinkendsentence: new ..	189	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	208	\Glsxtrtitleshort	323
\Glsxtrshortpl: new	207	renamed from \glstentryfmtshort ..	323
\glsxtrshortpl: new	206	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	322
\glslabeltok	220	renamed from \glstentryfmtshort ..	322
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	218	\Glsxtrtitleshortpl	323
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	323
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	322
\Glsxtrshort	323	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	322
\glsxtrshort	322	\Glsfmttext: new	324
\Glsfmtshortpl: changed to use		\glsfmttext: new	324
\glsxtrshortpl	323	\glshasattribute: new	168
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	167
\glsxtrshortpl	322	\glsxtremsuffix: new	261
\glsxtrifemptyglossary: new	28	\GlsXtrEnableEntryCounting: new ..	99
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	102
argument	171	\glsxtrscfont: new	233
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	233
argument	171	\glsxtrsmfont: new	247
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	248
default type to \acronymtype	113	short-em: new	269
\newterm: fixed name argument	170	short-em-desc: new	270
0.5 (2015-12-07)		short-em-footnote: new	279
\cGLS: new	104	short-em-long: new	265
\cGLS@: new	105	short-em-long-desc: new	266

short-em-postfootnote: new	281	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	244	attribute	313
short-sc-postfootnote: new	245	\Glsxtrheadtext: now uses headuc	
short-sm: new	251	attribute	316
short-sm-desc: new	252	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	258	attribute	315
short-sm-long: new	249	short-em-footnote: switch off regular	
short-sm-long-desc: new	250	attribute if set	280
short-sm-postfootnote: new	259	short-long: switch off regular attribute	
long-noshort-em: new	272	if set	219
long-noshort-em-desc: new	276	short-long-desc: switch off regular	
long-noshort-sm: new	254	attribute if set	220
long-noshort-sm-desc: new	256	short-sc-footnote: switch off regular	
long-short-em: new	262	attribute if set	244
long-short-em-desc: new	263	short-sm-footnote: switch off regular	
long-short-sm: new	248	attribute if set	258
long-short-sm-desc: new	249	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	217
\Glsaccessstext: new	154	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	218
\@gls@setup@default@short@access:		long-short-sc-desc: switch off regular	
removed \ifglxtruseuchead ...	313	attribute if set	235
\@glxtr@doaccsupp: new	21	footnote: switch off regular attribute if	
\Glsaccessdesc: new	157	set	221
\Glsaccessdescplural: new	158	postfootnote: switch off regular	
\Glsaccessfirst: new	155	attribute if set	223
\Glsaccessfirstplural: new	155	0.5.2 (2015-12-08)	
\Glsaccessname: new	153	\@GLSdesc@: added accessibility support	72
\Glsaccessplural: new	154	\@GLSdescplural@: added accessibility	
\Glsaccesssymbol: new	156	support	72
\Glsaccesssymbolplural: new	156	\@GLSfirst@: added accessibility	
\Glsxtrheadfirst: now uses headuc		support	69
attribute	317	\@GLSfirstplural@: added accessibility	
\glxtrheadfirst: now uses headuc		support	71
attribute	317	\@GLSname@: added accessibility support	71
\Glsxtrheadfirstplural: now uses		\@GLSplural@: added accessibility	
headuc attribute	318	support	70
\glxtrheadfirstplural: now uses		\@GLSsymbol@: added accessibility	
headuc attribute	318	support	73
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute	317	accessibility support	73
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	68
attribute	316	\@GLSdesc@: added accessibility support	72
\Glsxtrheadshort: now uses headuc		\@GLSdescplural@: added accessibility	
attribute	314	support	72
\glxtrheadshort: now uses headuc		\@GLSfirst@: added accessibility	
attribute	313	support	69
\Glsxtrheadshortpl: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	314	support	70

\@Glsname@: add accessibility support ..	71	\glsdohypertarget: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	153
\@Glsplural@: added accessibility support	70	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here	60
\@Glsymbol@: added accessibility support	73	\GlsXtrEnableInitialTagging: new	185
\@Glsymbolplural@: added accessibility support	73	\glsxtrfieldtitlecase: new	172
\@Glstext@: added accessibility support	68	\GlsXtrFormatLocationList: new ...	58
\@gls@setup@default@short@access: removed \glsxtrabbrvfmt	210	\glsxtrnewabbrevpresetkeyhook: new	197
\@glsdesc@: added accessibility support	71	\glsxtrtagfont: new	187
\@glsdescplural@: added accessibility support	72	\KV@printgloss@nonumberlist: added	60
\@glsfirst@: added accessibility support	69	\mfu@checkword@do: added	186
\@glsfirstplural@: added accessibility support	70	\setabbreviationstyle: added check for post-definition style switch	213
\@Glsname@: added accessibility support	71	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	70	\@glsxtr@autoindex@at: new	183
\@Glsymbol@: added accessibility support	72	\@glsxtr@autoindex@encap: new ...	183
\@Glsymbolplural@: added accessibility support	73	\@glsxtr@autoindex@esc: new	183
\@Glstext@: added accessibility support	68	\@glsxtr@autoindex@level: new ...	183
\@glsxtr@activate@initialtagging: new	187	\@glsxtr@autoindex@setname: new .	181
\@glsxtr@do@titlecaps@warn: new .	187	\@glsxtr@doabbreviationsdef: new .	17
\@glsxtr@tag: new	187	\glsdescwidth: added	57
\glossaryentrynumbers: added	57	\glsdohypertarget: removed \GlsXtrNoGlsWarningNoAutoMakeMain	130
\Glossentrydesc: added	185	\glspagelistwidth: added	57
\Glossentryname: added	176	\glsxtrdoautoindexname: new	181
\Glossentrysymbol: added	185	\glsxtrpostnamehook: new	178
\glossentrysymbol: added	185	\if@glsxtr@format@override: new .	180
\GLSaccessdesc: new	157, 165	\ProvidesGlossariesExtraLang: new	329
\GLSaccessdescplural: new ...	158, 165	\RequireGlossariesExtraLang: new	328
\GLSaccessfirst: new	155, 164	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	155, 164	\@newglossaryentry@defunitcounters: new	106
\GLSaccesslong: new	159, 166	\@GLSxtr@p@acrlong@: new	91
\GLSaccesslongpl: new	160, 166	\@GLSxtr@p@acrlongpl@: new	91
\Glsaccesslongpl: new	160	\@GLSxtr@p@acrshort@: new	91
\glsaccesslongpl: new	160	\@GLSxtr@p@acrshortpl@: new	91
\GLSaccessname: new	153, 163	\@GLSxtr@p@long@: new	90
\GLSaccessplural: new	154, 164	\@GLSxtr@p@longpl@: new	91
\GLSaccessshort: new	158, 165	\@GLSxtr@p@plural@: new	89
\GLSaccessshortpl: new	159, 166	\@GLSxtr@p@short@: new	90
\GLSaccesssymbol: new	156, 164	\@GLSxtr@p@shortpl@: new	90
\GLSaccesssymbolplural: new .	157, 165	\@GLSxtr@p@text@: new	89
\GLSaccessstext: new	154, 163	\@GlsXtrEnableOnTheFly: new	53
		\@Glsxtr: new	54
		\@Glsxtr@p@acrlong@: new	91

\@Glsxtr@p@acrlongpl@: new	91	\glstdonohyperlink: added	88
\@Glsxtr@p@acrshort@: new	91	\glsenableentryunitcount: new	108
\@Glsxtr@p@acrshortpl@: new	91	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	90	entry's existence	168
\@Glsxtr@p@longpl@: new	90	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	89	entry's existence	168
\@Glsxtr@p@short@: new	89	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	90	check	188
\@Glsxtr@p@text@: new	89	\Glsxtr: new	54
\@Glsxtrpl: new	55	\glxtr: new	54
\@alt@glshyp@opt: new	85	\glxtrcat: new	53
\@gl@alt@hyp@opt: new	85	\glxtrdowrglossaryhook: new	85
\@gl@alt@hyp@opt@char: new	85	\GlsXtrEnableEntryUnitCounting:	
\@gl@alt@hyp@opt@keys: new	85	new	111
\@gl@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	53
new	107	\Glsxtrpl: new	55
\@gl@local@increment@currunitcount:		\glxtrpl: new	54
new	107	\glxtrpostlocalreset: new	99
\@gl@setdefault@glslink@opts:		\glxtrpostlocalunset: new	98
new	82	\glxtrpostreset: new	99
\@glxtr: new	54	\glxtrpostunset: new	97
\@glxtr@addunitcounter: new	106	\glxtrprotectlinks: new	88
\@glxtr@currunitcount: new	108	\GlsXtrSetAltModifier: new	85
\@glxtr@ifunitcounter: new	106	\GlsXtrSetDefaultGlsOpts: new	84
\@glxtr@p@acrlong@: new	91	\glxtrstarflywarn: new	53
\@glxtr@p@acrlongpl@: new	91	\GlsXtrWarning: new	55
\@glxtr@p@acrshort@: new	91	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new	91	disables \setacronymstyle	113
\@glxtr@p@long@: new	90	1.0 (2016-01-24)	
\@glxtr@p@longpl@: new	90	\@glxtr@autoindexcrossrefs: new	15
\@glxtr@p@plural@: new	89	\@glxtr@idx@displaynumberlist:	
\@glxtr@p@short@: new	89	new	122
\@glxtr@p@shortpl@: new	90	\@glxtr@idx@entrynumberlist: new	123
\@glxtr@p@text@: new	89	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new	108	new	122
\@glxtr@setentryunitcountunsetattr:		\@glxtr@noidx@entrynumberlist:	
new	111	new	123
\@glxtr@unitcountlist: new	106	\@glxtr@noidx@numberlistloop:	
\@glxtrpl: new	54	new	122
\@newglossaryentryposthook: added		\@glxtr@reg@glosslist: new	114
empty see value if not set and added		\makeglossaries: new	115
'see' to field key map	45	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	53	\glxtrdiscardperiod: added check	
\cGlsformat: added	105	for first use	190
\cGlsformat: added	105	short-desc: fixed typo in	
\cGlsplformat: added	106	\glxtrinlinefullformat and	
\cGlsplformat: added	105	added missing second argument	227
\glstdisablehyper: added	87	1.02 (2016-04-25)	
\glstdohyperlink: added	86	\@glxtr@current@style: new	56

\@glsfirst@: set abbreviation and regular format	69	\glsxtrregularfont: new	60
\@glsfirstplural@: set abbreviation and regular format	70	\glsxtruserfield: new	283
\@glsname@: set abbreviation and regular format	71	\glsxtruserparen: new	283
\@glsplural@: set abbreviation and regular format	70	\glsxtrusersuffix: new	284
\@glsymbol@: set regular format	72	\GlsXtrWarnDeprecatedAbbrStyle:	
\@glsymbolplural@: set regular format	73	new	215
\@glstext@: set abbreviation and regular format	68	short-em-long-em: new	267
\@glsxtr@deprecated@abbrstyle:		short-em-long-em-desc: new	268
new	215	short-em-nolong: new	270
\@glsxtr@do@style: new	22	short-em-nolong-desc: new	271
\@glsxtr@doloctag: new	60	short-em-postfootnote: renamed from “postfootnote-em”	281
\@glsxtr@idx@entrynumberlist:		short-footnote: new	223
switched from \let to \newcommand	123	short-long-user: new	290
\@glsxtr@pagetag: new	59	short-long-user-desc: new	292
\@glsxtr@pagetag: new	59	short-nolong: new	226
\@glsxtr@preloctag: new	59	short-nolong-desc: new	228
\@glsxtr@postloctag: new	59	short-postfootnote: new	225
\@glsxtr@preloctag: new	59	short-sc-footnote: renamed from “footnote-sc”	244
\glossentrydesc: added glossdescfont attribute check	172	short-sc-nolong: new	238
\Glossentryname: added glossnamefont attribute check	176	short-sc-nolong-desc: new	240
\glossentryname: added glossnamefont attribute check	174	short-sc-postfootnote: renamed from “postfootnote-sc”	245
moved post name hook inside condition	176	short-sm-footnote: renamed from “footnote-sm”	258
\glsabbrvmfont: new	261	short-sm-nolong: new	252
\glsabbrvuserfont: new	283	short-sm-nolong-desc: new	254
\glsfirstabbrvmfont: new	261	short-sm-postfootnote: renamed from “postfootnote-sm”	259
\glsfirstabbrvuserfont: new	283	\letabbreviationstyle: new	215
\glsfirstlongemfont: new	262	\newabbreviationstyle: bug fix: corrected test for existence	214
\glsfirstlonguserfont: new	284	long-em-noshort-em: new	274
\glsifnotregularcategory: new ...	169	long-em-noshort-em-desc: new ...	277
\glslongdefaultfont: new	199	long-em-short-em: new	263
\glslongemfont: new	262	long-em-short-em-desc: new	264
\glslongfont: new	199	long-noshort: new	232
\glslonguserfont: new	284	long-noshort-desc: new	231
\glsxtrassignfieldfont: new	68	long-noshort-em: renamed from “long-em”	272
\GlsXtrEnablePreLocationTag: new .	58	long-noshort-em-desc: renamed from “long-desc-em”	276
\glsxtrfirstscfont: new	233	long-noshort-sc: renamed from “long-sc”	240
\glsxtrfirstsmfont: new	247	long-noshort-sc-desc: renamed from “long-desc-sc”	242
\glsxtrlongshortdescsort: new ...	218	long-noshort-sm: renamed from “long-sm”	254
\glsxtrpostnamehook: added category check	178		

long-noshort-sm-desc: renamed from	General: disabled docdef key at the start
\long-desc-sm 256	of the document 27
long-short-user: new 284	docdef option changed to choice 14
long-short-user-desc: new 290	\glstr@usesee: new 45
\renewabbreviationstyle: new 215	\glstrusesee: new 45
style: new 22	\glstruseseeformat: new 45
1.05 (2016-06-10)	@if@glstrdocdefrestricted: new .. 15
\eglssetwidest: new 384	1.07 (2016-08-15)
\glFindWidestAnyName: new 386	@@glstrp: new 92
\glFindWidestAnyNameLocation:	\GLSfirst@: added check for
new 392	nohyperfirst attribute 69
\glFindWidestAnyNameSymbol: new 389	\GLSfirstplural@: added check for
\glFindWidestAnyNameSymbolLocation:	nohyperfirst attribute 71
new 391	\GLSxtrp: new 93
\glFindWidestLevelTwo: new 388	\@Glsfirst@: added check for
\glFindWidestUsedAnyName: new .. 386	nohyperfirst attribute 69
\glFindWidestUsedAnyNameLocation:	\@Glsfirstplural@: added check for
new 391	nohyperfirst attribute 70
\glFindWidestUsedAnyNameSymbol:	\@Glsxtrp: new 92
new 389	\@glspreglossaryhook: added
\glFindWidestUsedAnyNameSymbolLocation:	\glossxtrsetpopts 188
new 390	\@glfirst@: added check for
\glFindWidestUsedLevelTwo: new . 387	nohyperfirst attribute 69
\glFindWidestUsedTopLevelName:	\@glfirstplural@: added check for
new 385	nohyperfirst attribute 70
\glfirstlongfootnotefont: new .. 221	\@glxtrinmark: new 311
\glsgetwidestname: new 385	\@glxtrnotinmark: new 311
\glsgetwidestsubname: new 385	\@glxtrp: new 92
\glslongfootnotefont: new 221	\@glxtrp@opt: new 91
\glxtrAltTreeIndent: new 383	\glossxtrsetpopts: new 92
\glxtralttreeInit: new 384	\glsp: new 94
\glxtrAltTreePar: new 383	\glsp: new 94
\glxtrAltTreeSetHangIndent: new 393	\glxtr@entry@p: new 93
\glxtrAltTreeSetSubHangIndent:	\glxtrabbrvfootnote: new 221
new 393	\glxtrchecknohyperfirst: new 69
\glxtralttreeSubSymbolDescLocation:	\glxtrfieldtitlecasecs: new 172
new 383	\glxtrifinmark: new 310
\glxtralttreeSymbolDescLocation:	\GLSxtrp: new 95
new 383	\Glsxtrp: new 94
\glxtrComputeTreeIndent: new ... 392	\glxtrp: new 93
\glxtrComputeTreeSubIndent: new 393	\glxtrsetpopts: new 92
\glxtrtreetopindent: new 384	short-long-desc: added text key 220
short-em-long: fixed incorrect font used	fixed misspelling of \glabbrvfont in
by long form 266	plural key 220
\xglsetwidest: new 384	long-short-desc: added missing text
1.06 (2016-06-18)	key 218
\@glsoifexistsorwarn: new 15	fixed misspelling of \glabbrvfont . 218
\@glstr@docdefval: new 15	footnote: changed first forms to use
\@glstr@usesee: new 45	\glfirstlongfootnotefont ... 221

postfootnote: removed \footnote		\@printglossary: redefined to save	
from first keys	223	options	120
switched from \glsfirstlongfont to		\glsxtr@makeglossaries: new	121
\glsfirstlongfootnotefont ...	224	1.10 (2016-12-17)	
\RestoreAcronyms: modified		\@GLSp1@: fixed bug caused by typo in	
\@gls@link@checkfirsthyper to		command name	62
set \glsxtrifwasfirstuse	114	1.11 (2017-01-19)	
1.08 (2016-12-13)		\@glsxtr@do@redef@forglsentries:	
\@glsxtr@record: new	8	new	6
\@GLS@: added \@glsxtr@record	62	\@glsxtr@noidx@do: new	142
\@GLSp1@: added \@glsxtr@record ...	62	\@glsxtr@redef@forglsentries: new .	6
\@Gls@: added \@glsxtr@record	61	\@glsxtr@shortcutsval: new	20
\@GLSp1@: added \@glsxtr@record ...	62	\@glsxtr@unsrt@getgrouptitle: new	141
\@Gls@: added \@glsxtr@record	61	\@print@noidx@glossary: added	
\@gls@link@: added		redefinition	125
\@glsxtr@record	62	\glsxtr@addloclistfield: added	
\@gls@field@link: added		group key	13
\@glsxtr@record	61	added location key	12
\@gls@saveentrycounter: new	27	\glsxtr@fields: new	134
\@glsdisp: added \@glsxtr@record ..	62	\glsxtr@linkprefix: new	134
\@GLSp1@: added \@glsxtr@record ...	61	\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	134
\@glsxtr@warn@onexistsordo: new ...	6	\glsxtr@texencoding: new	134
\@glsxtr@warn@undefaction: new ...	6	\glsxtr@writefields: new	134
\@print@unsrt@glossary: new	139	\GlsXtrLoadResources: new	133
General: added record package option ...	13	\glsxtrpageref: new	37
\glsadd: added \@glsxtr@record	67	\glsxtrresourcefile: changed	
\glsdoifexists: now defines		extension to .gls tex	133
\glslabel	43	\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	136	definition	187
\glsxtr@resource: new	134	\@glsxtr@counterrecordhook: new .	136
\glsxtr@saveentrycounter: new	27	\@glsxtr@display@loc: new	126
\glsxtr@setup@record: new	12	\@glsxtr@docounterrecord: new ...	136
\glsxtrassignfieldfont: added check		\@glsxtr@longnewglossaryentry:	
for existence	68	new	39
\glsxtrresourcefile: new	133	\@glsxtr@noop@recordcounter: new .	11
\printunsrtglossaries: new	138	\@glsxtr@op@recordcounter: new ...	11
\printunsrtglossary: new	138	\@glsxtr@provide@storagekey: new .	28
1.09 (2016-12-16)		\@glsxtr@s@longnewglossaryentry:	
\@glsxtr@gettype: new	121	new	39
\@glsxtr@mixed@assign@sortkey:		\@glsxtrentryfmt: new	30
new	121	\@glsxtrindexaliased: new	83
		\@glsxtrsetaliasnoindex: new	83

\@newglossaryentryposthook: added check for alias key	49	\glxtrindexaliased: new	83
\@no@glxtrindexaliased: new	83	\GlsXtrLetField: new	34
\@printunsrtglossary: new	138	\GlsXtrLetFieldToField: new	34
General: added target key to printgloss family	120	\GlsXtrLoadResources: removed restriction on only one per document	133
\apptoglossarypreamble: new	38	\glxtrlocrangefmt: new	127
\csGlsXtrLetField: new	34	\glxtrpostlongdescription: new ..	40
\eGlsXtrSetField: new	34	\glxtrprovidestoragekey: new	28
\gGlsXtrSetField: new	34	\GlsXtrRecordCounter: new	136
\glsdohyperlink: added check for alias field	86	\glxtrresourcecount: new	133
\glsnoidxdisplayloc: added redefinition	126	\glxtrresourcefile: added catcode change for @	133
\glissettoctitle: added patch	41	\glxtrsetaliasnoindex: new	83
\glxtr@counterrecord: new	136	\GlsXtrSetField: new	34
\glxtr@langtag: new	134	\glxtrsetfieldifexists: new	34
\glxtr@newabbreviation: new	194	\glxtrunsrtdo: new	141
\glxtr@org@newignoredglossary: Added check for existence	40	\glxtrusefield: new	33
\glxtr@pluralsuffixes: new	134	\glxtrusefield: new	33
\glxtr@provideignoredglossary: new	42	short-postlong-user: new	287
\glxtr@s@newignoredglossary: Added check for existence	41	short-postlong-user-desc: new ...	289
\glxtr@s@provideignoredglossary: new	42	\longnewglossaryentry: added starred version	39
\glxtrabbrvpluralsuffix: new ...	199	long-postshort-user: new	285
\glxtralias: new	49	long-postshort-user-desc: new ...	287
\glxtrcopytoglossary: new	43	postdot: new	16
\glxtrdeffield: new	33	\pretoglossarypreamble: new	38
\glxtrdisplayendloc: new	126	\print@noop@unsrtglossaryunit: new	141
\glxtrdisplayendlochook: new ...	127	\print@op@unsrtglossaryunit: new	141
\glxtrdisplaysingleloc: new	126	\printunsrtglossary: added starred form	138
\glxtrdisplaystartloc: new	126	\printunsrtglossaryhandler: new .	140
\glxtrredefield: new	34	\printunsrtglossaryunit: new	12
\glxtrentryfmt: new	30	\printunsrtglossaryunitsetup: new	141
\glxtrfieldddolistloop: new	31	\provideignoredglossary: new	41
\glxtrfielddforlistloop: new	31	\s@glxtr@provide@storagekey: new	29
\glxtrfielddifylist: new	31	\s@printunsrtglossary: new	138
\glxtrfielddlistadd: new	31	\xGlsXtrSetField: new	34
\glxtrfielddlistadd: new	31	1.13 (2017-02-07)	
\glxtrfielddlistgadd: new	31	\@glsdisp: removed	
\glxtrfielddlistxadd: new	31	\@glxtr@org@glsdisp	62
\glxtrfielddxifylist: new	32	\glxtrsetaliasnoindex: switched to \providecommand	83
\glxtrfmt: new	29	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new	29	\@gls@link: added redefinition	64
\GlsXtrFmtField: new	29	\@gls@noidx@getgrouptitle: new ..	123
\glxtrifkeydefined: new	28	\@gls@removespaces: new	127
		\@glxtr@do@automake@err: new ...	135
		\@glxtr@org@gloautosee: new	25

\@glxtr@record: added third arg	7	postfootnote: fixed spelling of	
\@glxtr@recordsee: new	11	\glsabbrvfont	223
General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	67	\@glo@autosee: added redefinition	26
added \glsadd option thevalue	66	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	87	bug	123
\glsenableentrycount: fixed		\@glxtr@addunusedxrefs: added	
assignment of \@cGls@	101	check for seealso field	50
\glsenableentryunitcount: fixed		\@glxtr@checkgroup: use \csuse	
assignment of \@cGls@	109	instead of \csname	142
\glsnavigation: new	125	\@glxtr@dorecordnodefer: new	11
\glxtr@org@getgrouptitle: new ..	124	\@print@unsrt@glossary: corrected	
\glxtr@recordsee: new	7	misspelt command	139
\glxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	135	new	140
\glxtrdisplayendloc: added check		General: added check for	
for empty format	126	\@gls@setupsort@none	14
\glxtrgetgrouptitle: new	124	\gls@checkseeallowed: added	
\glxtrinitwrgloss: new	63	redefinition	26
\glxtrlocationhyperlink: new ...	127	\glxtr@writefields: added	
\glxtrsetgrouptitle: new	124	\providecommand lines	134
\glxtrsupphypernumber: new	128	\glxtrautoindex: new	181
\ifglxtrwrglossbefore: new	63	\glxtrautoindexassignsort: new .	181
1.15 (2017-05-10)		\glxtrautoindexentry: new	181
\@glxtr@dorecord: corrected		\glxtrindexseealso: new	46
premature expansion of \@glslocref	10	\glxtrseealsolabels: new	49
short-em-long-em: fixed spelling of		\glxtrseelist: new	46
\glsabbrvfont	267	\glxtruseseealso: new	46
short-long: fixed spelling of		\glxtruseseealsoformat: new	46
\glsabbrvfont	219	\seealso: new	46
short-long-user: fixed spelling of		autoseeindex: new	16
\glsabbrvfont	291	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@gls@setup@default@short@access:	
\glsabbrvfont	288	removed some inconsistencies in the	
short-postlong-user-desc: fixed		abbreviation styles	216
spelling of \glsabbrvfont	289	\@glxtr@mark@wordseps: new	194
long-em-short-em: fixed spelling of		\@glxtr@markwordseps: new	194
\glsabbrvfont	264	\@glxtr@noidx@displaynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	285	\glxtrundeftag	122
long-postshort-user-desc: fixed		\@glxtr@noidx@entrynumberlist:	
spelling of \glsabbrvfont	287	replace hard-coded ?? with	
long-short: fixed spelling of		\glxtrundeftag	123
\glsabbrvfont	217	\@glxtr@noidx@numberlistloop:	
long-short-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	284	\glxtrundeftag	123
footnote: fixed spelling of		\@glxtrtrifhyphenstart: new	293
\glsabbrvfont	221	\glsabbrvhyphenfont: new	293
		\glsabbrvonlyfont: new	306

<code>\glsabbrvscfont:new</code>	233	<code>short-hyphen-postlong-hyphen-desc:</code>	
<code>\glsabbrvsmfont:new</code>	247	<code>new</code>	306
<code>\glsabbrvuserfont:initialised to</code>		<code>short-long-user-desc:corrected first</code>	
<code>default font</code>	283	<code>forms</code>	292
<code>\glsfirstabbrvhyphenfont:new</code> ...	293	<code>short-nolong-desc-noreg:new</code> ...	228
<code>\glsfirstabbrvonlyfont:new</code>	306	<code>short-nolong-noreg:new</code>	226
<code>\glsfirstabbrvscfont:new</code>	233	<code>long-em-noshort-em-desc-noreg:</code>	
<code>\glsfirstabbrvsmfont:new</code>	247	<code>new</code>	279
<code>\glsfirstlonghyphenfont:new</code> ...	293	<code>long-em-noshort-em-noreg:new</code> ...	275
<code>\glsfirstlongonlyfont:new</code>	307	<code>long-hyphen-noshort-desc-noreg:</code>	
<code>\glslonghyphenfont:new</code>	293	<code>new</code>	295
<code>\glslongonlyfont:new</code>	306	<code>long-hyphen-postshort-hyphen:new</code>	299
<code>\glslonguserfont:initialised to default</code>		<code>long-hyphen-postshort-hyphen-desc:</code>	
<code>font</code>	284	<code>new</code>	300
<code>\glxtr@newabbreviation:added</code>		<code>long-hyphen-short-hyphen:new</code> ...	294
<code>\glxtrorgshort and</code>		<code>long-hyphen-short-hyphen-desc:</code>	
<code>\glxtrorglong</code>	195	<code>new</code>	294
<code>\GlsXtrDefineAcShortcuts:new</code> ...	18	<code>long-noshort-desc-noreg:new</code> ...	231
<code>\glxtrgenabbrvfmt:added check for</code>		<code>long-noshort-noreg:new</code>	232
<code>\ifglxtrininsertinside</code>	210	<code>long-only-short-only:new</code>	307
<code>\glxtrhyphensuffix:new</code>	293	<code>long-only-short-only-desc:new</code> ..	308
<code>\glxtrifhyphenstart:new</code>	292	<code>long-short-user-desc:corrected first</code>	
<code>\glxtrlonghyphen:new</code>	298	<code>forms</code>	290
<code>\glxtrlonghyphennoshort:new</code> ...	295	1.18 (2017-08-10)	
<code>\glxtrlonghyphenshort:new</code>	293	<code>stylemods:changed default value to</code>	
<code>\glxtrlongshortdescname:new</code> ...	218	<code>"default"</code>	21
<code>\glxtronlydescname:new</code>	308	1.19 (2017-09-09)	
<code>\glxtronlydescsort:new</code>	308	<code>\@glxtr@defaultnumberformat:new</code> .	7
<code>\glxtronlysuffix:new</code>	307	<code>\@glxtr@dorecord:Use</code>	
<code>\glxtrparen:new</code>	197	<code>\@glxtr@recordlocf instead of</code>	
<code>\glxtrposthyphenlong:new</code>	303	<code>\@glxtr@recordlocf</code>	10
<code>\glxtrposthyphenshort:new</code>	298	<code>\@glxtr@dorecordnodefer:Use</code>	
<code>\glxtrposthyphensubsequent:new</code>	298	<code>\theglentrycounter for the</code>	
<code>\glxtrshortdescname:new</code>	227	<code>location rather than \@glxtr@recordlocf</code> ..	11
<code>\glxtrshorthyphen:new</code>	303	<code>\@glxtr@record@setting:new</code>	13
<code>\glxtrshorthyphenlong:new</code>	301	<code>\@glxtr@record@setting@alsoindex:</code>	
<code>\glxtrshortlongdescname:new</code> ...	220	<code>new</code>	13
<code>\glxtrshortlongdescsort:new</code> ...	220	<code>\@glxtrifhasfield:new</code>	32
<code>\GlsXtrsubsequentfmt:new</code>	213	General:added <code>\glslink</code> option	
<code>\glxtrsubsequentfmt:new</code>	212	<code>theHvalue</code>	63
<code>\GlsXtrsubsequentplfmt:new</code>	213	added <code>\glslink</code> option <code>thevalue</code> ...	63
<code>\glxtrsubsequentplfmt:new</code>	212	<code>\glxtr@writefields:removed</code>	
<code>\glxtrword:new</code>	194	<code>double-quotes around \jobname</code> ..	135
<code>\glxtrwordsep:new</code>	194	<code>\glxtrdoautoindexname:changed</code>	
<code>short-hyphen-long-hyphen:new</code> ...	302	<code>format test</code>	181
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink:new</code>	87
<code>new</code>	302	<code>\glxtrifhasfield:new</code>	32
<code>short-hyphen-postlong-hyphen:new</code>	304	<code>\GlsXtrSetDefaultNumberFormat:</code>	
		<code>new</code>	7

\s@glxstrifhasfield: new	32	\@glxtrsetaliasnoindex: changed to use \glxstrifhasfield instead of \ifglshasfield	83
1.20 (2017-09-11)		\@glxtrwrglossmark: new	24
\@glxtrhypernameprefix: new	121	\@rGLS: new	149
\glsdohypertarget: added redefinition	121	\@rGLS@: new	149
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glxtrhypernameprefix	141	\@rGLSpl: new	150
1.21 (2017-11-03)		\@rGLSpl@: new	150
\@@glxtr@record: added check for default options	9	\@rGls: new	149
\@@glxtrwrglossmark: new	24	\@rGls@: new	149
\@gls@setup@default@short@access: modified index to remove hard coded \space	378	\@rGlspl: new	149
modified list to remove hard coded \space	367	\@rGlspl@: new	149
moved conditional outside of \glsgroupskip	370–377	\@rgls: new	148
redefined altlistgroup to discourage breaks after group headings	368	\@rgls@: new	148
redefined altlisthypergroup to discourage breaks after group headings	369	\@rglspl: new	148
redefined indexgroup to discourage breaks after group headings	379	\@rglspl@: new	148
redefined indexhypergroup to discourage breaks after group headings	379	General: adjusted mcolalttree	398
redefined listgroup to discourage breaks after group headings	368	ac	21
redefined listhypergroup to discourage breaks after group headings	368	new	401
\@glslink: changed \let to \def	88	redefined alttreegroup to discourage breaks after group headings	394
\@glxtr@checkgroup: new	142	redefined alttreehypergroup to discourage breaks after group headings	395
\@glxtr@defpostpunc: new	16	redefined mcolalttreegroup to discourage breaks after group headings	399
\@glxtr@do@record@wrglossary: new	8	redefined mcolalttreehypergroup to discourage breaks after group headings	399
\@glxtr@dosee@alsoindex@glossary: new	25	redefined mcolalttreespannav to discourage breaks after group headings	400
\@glxtr@doseeglossary: new	25	redefined mcolindexgroup to discourage breaks after group headings	395
\@glxtr@noidx@do: removed code dealing with the group	143	redefined mcolindexhypergroup to discourage breaks after group headings	395
\@glxtr@record@setting@off: new .	13	redefined mcolindexspannav to discourage breaks after group headings	396
\@glxtr@record@setting@only: new	13	redefined mcoltreegroup to discourage breaks after group headings	396
\@glxtr@rglstrigger@record: new	147	redefined mcoltreehypergroup to discourage breaks after group headings	397
\@glxtrglossentry: new	136		
\@glxtrnewgls: new	144		

redefined mcoltreenonamegroup to discourage breaks after group headings	397	\glxstr@org@dohyperlink: new	85
redefined mcoltreenonamehypergroup to discourage breaks after group headings	398	\glxstr@setbookindexmark: new ...	406
redefined mcoltreenonamespannav to discourage breaks after group headings	398	\glxstrbookindexatendgroup: new ..	402
redefined mcoltreespannav to discourage breaks after group headings	397	\glxstrbookindexbetween: new	402
redefined treenonamegroup to discourage breaks after group headings	382	\glxstrbookindexbookmark: new ...	402
redefined treenonamehypergroup to discourage breaks after group headings	382	\glxstrbookindexcols: new	401
debug: new	24	\glxstrbookindexcolspread: new ..	402
\gglsetwidest: new	384	\glxstrbookindexfirstmark: new ..	406
\gl:disablehyper: added check for existence	87	\glxstrbookindexfirstmarkfmt: new	406
changed to use \def rather than \let ..	87	\glxstrbookindexformatheader: new	402
\gl:dohypertarget: redefined treegroup to discourage breaks after group headings	380	\glxstrbookindexgroupskip: new ..	402
redefined treehypergroup to discourage breaks after group headings	381	\glxstrbookindexlastmark: new ...	406
\gl:enablehyper: changed to use \def rather than \let	87	\glxstrbookindexlastmarkfmt: new	406
\Glsfmtname: new	323	\glxstrbookindexmarkentry: new ..	406
\gl:fmtname: new	323	\glxstrbookindexname: new	401
\gl:shex: new	330	\glxstrbookindexparentchildsep: new	401
\gl:slstchildpostlocation: new ..	367	\glxstrbookindexparentschildsep: new	401
\gl:slstchildprelocation: new ...	367	\glxstrbookindexprelocation: new	401
\gl:slstprelocation: new	367	\glxstrbookindexsubatendgroup: new	402
\gl:snahyperlink: patched	85	\glxstrbookindexsubbetween: new ..	402
\gl:seeitemformat: new	45	\glxstrbookindexsubname: new	401
\gl:sshowtarget: new	25	\glxstrbookindexsubprelocation: new	401
\gl:streechildprelocation: new ...	378	\glxstrbookindexsubsubatendgroup: new	402
\gl:streeprelocation: new	378	\glxstrbookindexsubsubbetween: new	402
\gl:triggerrecordformat: new	148	\glxstrbookindexthepage: new	405
\gl:useabbrvfont: new	210	\glxstrdetoklocation: new	146
\gl:uselongfont: new	210	\glxstrehablerecordcount: new ...	146
\glxstr@do@alsoindex@wrglossary: new	8	\glxstrglossentry: new	136
\glxstr@org@do@wrglossary: new ..	27	\glxstrgroupfield: new	142
		\Glsxrheadname: new	315
		\glxtrheadname: new	314
		\GlsXtrIfFieldEqStr: new	35
		\glxtriflabelinlist: new	140
		\glxtrifrecordtrigger: new	147
		\glxtrindexseealso: added check that the entry exists	47
		\glxtrinithyperoutside: new	64
		\GlsXtrLocationRecordCount: new ..	146
		\glxtrnewgls: new	143, 144
		\glxtrnewGLSlike: new	145
		\glxtrnewglslike: new	145
		\glxtrnewrgls: new	145
		\glxtrnewrGLSlike: new	145

\glstrnewrglslike: new	145	\@glstr@orgprintglossary: changed	
\glstrprelocation: new	366, 401	explicit \let for \nopostdesc to	
\GlsXtrRecordCount: new	145	\glstractivatenopost	119
\glstrrecordtriggervalue: new ..	146	\@glstrglossentryother: new	138
\glstrresourcefile: now disables		\glossentrynameother: new	179
record key	133	\glseeitemformat: switched check	
\glstrresourceinit: new	133	from regular to short	45
\GlsXtrSetRecordCountAttribute:		\glstr@setaccessdisplay: new ...	178
new	146	\glstr@writefields: provide	
\glstrtitlename: new	315	\glstr@record in aux file	134
\glstrtitleorpdforheading: new .	311	\glstractivatenopost: new	119
\GlsXtrTotalRecordCount: new	145	\glstrbookindexprelocation:	
\glstrwrglossmark: new	24	removed check for no post dot	401
short-em: new	269	\glstrglossentryother: new	137
short-sc: corrected first letter		\glstrnopostpunc: new	120
uppercasing	238	1.23 (2017-11-12)	
short-sm: corrected first letter		\@glstrfmt: added check for indexing	30
uppercasing	252	added grouping	30
\ifglstr@hyperoutside: new	63	new	29
all: new	365	\@glstr@nopostpunc@postdesc: new	120
nolong-short: new	229	\@glstr@store@postpunc: new ..	120
nolong-short-em: new	271	\@glstrentryfmt: fixed missing label	
nolong-short-noreg: new	229	argument	30
nolong-short-sc: new	240	\@glstrfmt: new	29
nolong-short-sm: new	254	\eglupdatewidest: new	385
nopostdot: new	16	\gglupdatewidest: new	384
postpunc: new	16	\glupdatewidest: new	384
\printunrtglossaryentryprocesshook:		\GlsXtrDefineAbbreviationShortcuts:	
new	140	changed \newabbr definition to use	
\printunrtglossarypredoglossary:		\providecommand	18
new	140	\GlsXtrDefineAcShortcuts: changed	
\printunrtglossaryskipentry: new	140	\newabbr definition to use	
\rGLS: new	149	\providecommand	19
\rGls: new	148	\glstrfmtdisplay: new	30
\rgls: new	148	\glstrifcustomdiscardperiod: new	189
\rGLSformat: new	151	\GlsXtrIfFieldUndef: new	33
\rGlsformat: new	150	\glstrrestorepostpunc: new	120
\rglsformat: new	150	\s@glstrfmt: new	29
\rGLSpl: new	150	\s@glstrfmt: new	29
\rGlspl: new	149	\xglupdatewidest: new	385
\rglspl: new	148	1.24 (2017-11-14)	
\rGLSplformat: new	151	\gladd: added \@gl@setsort	67
\rGlsplformat: new	150	\glstrforcsvfield: new	32
\rglsplformat: new	150	\glstrlocalsetgrouptitle: new ..	124
\s@glstrifhasfield: switched from		1.25 (2017-11-14)	
\ifdef to \ifundef	32	\glstrbookindexmulticolenv: new	402
1.22 (2017-11-08)		1.25 (2017-11-24)	
\@glstr@nopostpunc: new	120	\glsextrapostnamehook: new	178
		\glstrfootnotename: new	221

\glxtrlongnoshortdescname: new .	230	\glxtrGeneralLatinIIrules: new .	340
\glxtrlongnoshortname: new	232	\glxtrGeneralLatinIrules: new ..	340
\glxtrlongshortname: new	216	\glxtrGeneralLatinIVrules: new .	342
\glxtrlongshortuserdescname: new	287	\glxtrGeneralLatinVIIIrules: new	345
\glxtronlyname: new	307	\glxtrGeneralLatinVIIrules: new	344
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVrules: new .	343
changed to use \glxtrparen	189	\glxtrGeneralLatinVrules: new ..	342
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules: new ..	339
changed to use \glxtrparen	189	\glxtrgeneralpuncIrules: new ...	338
\glxtrshortlongname: new	219	\glxtrgeneralpuncrules: new	338
\glxtrshortlonguserdescname: new	289	\glxtrhyphenrules: new	338
\glxtrshortnolongname: new	225	\glxtrLatinA: new	345
1.26 (2018-01-05)		\glxtrLatinAA: new	347
\@glxtr@do@inc@linkcount: new ..	151	\glxtrLatinAELigature: new	347
\glslinkpresetkeys: new	64	\glxtrLatinE: new	345
\glxtr@inc@linkcount: new	64	\glxtrLatinEszettSs: new	347
\GlsXtrEnableLinkCounting: new ..	152	\glxtrLatinEszettSz: new	347
\GlsXtrIfLinkCounterDef: new	152	\glxtrLatinEth: new	347
\glxtrinclinkcounter: new	152	\glxtrLatinH: new	346
\GlsXtrLinkCounterName: new	152	\glxtrLatinI: new	346
\GlsXtrLinkCounterValue: new	152	\glxtrLatinInsularG: new	348
\GlsXtrTheLinkCounter: new	152	\glxtrLatinK: new	346
1.27 (2018-02-26)		\glxtrLatinL: new	346
\@glset@setup@default@short@access:		\glxtrLatinLslash: new	348
added glossaries-extra-bib2gls.sty .	329	\glxtrLatinM: new	346
\@glxtrdialecthook: new	27	\glxtrLatinN: new	346
\Alpha: new	332	\glxtrLatinO: new	346
\Beta: new	332	\glxtrLatinOELigature: new	347
\Chi: new	333	\glxtrLatinOslash: new	348
\Digamma: new	333	\glxtrLatinP: new	346
\Epsilon: new	332	\glxtrLatinS: new	346
\Eta: new	332	\glxtrLatinSchwa: new	347
\glxtr@loaddialect: new	329	\glxtrLatinT: new	346
\glxtrBasicDigitrules: new	362	\glxtrLatinThorn: new	347
\glxtrcombiningdiacriticIIrules:		\glxtrLatinWynn: new	348
new	336	\glxtrLatinX: new	347
\glxtrcombiningdiacriticIIrules:		\glxtrMathGreekIIrules: new	354
new	336	\glxtrMathGreekIrules: new	353
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new	358
new	336	\glxtrMathItalicBeta: new	358
\glxtrcombiningdiacriticIVrules:		\glxtrMathItalicChi: new	361
new	337	\glxtrMathItalicDelta: new	359
\glxtrcombiningdiacriticrules:		\glxtrMathItalicEpsilon: new ...	359
new	335	\glxtrMathItalicEta: new	359
\glxtrcontrolrules: new	334	\glxtrMathItalicGamma: new	359
\glxtrcurrencyrules: new	339	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new	362	new	350
\glxtrfractionrules: new	363	\glxtrMathItalicGreekIrules: new	349
\glxtrGeneralLatinIIIrules: new	341	\glxtrMathItalicIota: new	359

<code>\glxtrMathItalicKappa:new</code>	360	<code>\glxtrUpPi:new</code>	357
<code>\glxtrMathItalicLambda:new</code>	360	<code>\glxtrUpPsi:new</code>	358
<code>\glxtrMathItalicLowerGreekIIrules:</code>		<code>\glxtrUpRho:new</code>	357
<code>new</code>	352	<code>\glxtrUpSigma:new</code>	357
<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau:new</code>	358
<code>new</code>	351	<code>\glxtrUpTheta:new</code>	356
<code>\glxtrMathItalicMu:new</code>	360	<code>\glxtrUpUpsilon:new</code>	358
<code>\glxtrMathItalicNabla:new</code>	362	<code>\glxtrUpXi:new</code>	357
<code>\glxtrMathItalicNu:new</code>	360	<code>\glxtrUpZeta:new</code>	356
<code>\glxtrMathItalicOmega:new</code>	361	<code>\Iota:new</code>	332
<code>\glxtrMathItalicOmicron:new</code>	360	<code>\Kappa:new</code>	332
<code>\glxtrMathItalicPartial:new</code>	362	<code>\Mu:new</code>	332
<code>\glxtrMathItalicPhi:new</code>	361	<code>\Nu:new</code>	332
<code>\glxtrMathItalicPi:new</code>	360	<code>\Omicron:new</code>	332
<code>\glxtrMathItalicPsi:new</code>	361	<code>\omicron:new</code>	333
<code>\glxtrMathItalicRho:new</code>	360	<code>\Rho:new</code>	332
<code>\glxtrMathItalicSigma:new</code>	361	<code>\Tau:new</code>	332
<code>\glxtrMathItalicTau:new</code>	361	<code>\Upalpha:new</code>	333
<code>\glxtrMathItalicTheta:new</code>	359	<code>\Upbeta:new</code>	333
<code>\glxtrMathItalicUpperGreekIIrules:</code>		<code>\Upchi:new</code>	334
<code>new</code>	351	<code>\Upsilon:new</code>	333
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta:new</code>	333
<code>new</code>	350	<code>\Upiota:new</code>	333
<code>\glxtrMathItalicUpsilon:new</code>	361	<code>\Upkappa:new</code>	333
<code>\glxtrMathItalicXi:new</code>	360	<code>\Upmu:new</code>	333
<code>\glxtrMathItalicZeta:new</code>	359	<code>\Upnu:new</code>	333
<code>\glxtrMathUpGreekIIrules:new</code>	349	<code>\Upomicron:new</code>	333
<code>\glxtrMathUpGreekIrules:new</code>	348	<code>\upomicron:new</code>	334
<code>\glxtrnonprintablerules:new</code>	335	<code>\Uprho:new</code>	333
<code>\glxtrprovidecommand:new</code>	330	<code>\Uptau:new</code>	334
<code>\glxtrspacerules:new</code>	335	<code>\Upzeta:new</code>	333
<code>\glxtrSubScriptDigitrules:new</code>	362	<code>\Zeta:new</code>	332
<code>\glxtrSuperScriptDigitrules:new</code>	362		
<code>\glxtrUpAlpha:new</code>	355	1.28 (2018-03-06)	
<code>\glxtrUpBeta:new</code>	355	<code>\@glxtr@docdefval</code> : changed from	
<code>\glxtrUpChi:new</code>	358	count register to macro	15
<code>\glxtrUpDelta:new</code>	355	<code>\@glxtrdialecthook</code> : save and restore	
<code>\glxtrUpDigamma:new</code>	356	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon:new</code>	356	363
<code>\glxtrUpEta:new</code>	356	<code>\glxtrredeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code>	355	<code>\protected@csedef</code>	34
<code>\glxtrUpIota:new</code>	356	<code>\glxtrlocalsetgrouptitle</code> : changed	
<code>\glxtrUpKappa:new</code>	356	<code>\csedef\protected@csedef</code>	124
<code>\glxtrUpLambda:new</code>	357	<code>\glxtrsetgrouptitle</code> : changed	
<code>\glxtrUpMu:new</code>	357	<code>\csxdef\protected@csxdef</code>	124
<code>\glxtrUpNu:new</code>	357	1.29 (2018-04-09)	
<code>\glxtrUpOmega:new</code>	358	<code>\@glxtr@removespaces</code> : added expansion	127
<code>\glxtrUpOmicron:new</code>	357	<code>\@glxtr@dorecord</code> : don't suppress	
<code>\glxtrUpPhi:new</code>	358	expansion of <code>\@glxtrrecordloc</code> if	
		counter isn't page	11

\@glxstr@wrglossary@locationhyperlink: new	23	\Glsxtrlongpl: added \@glxstr@record	209
\glxstr@inc@wrglossaryctr: new ...	22	\glxtrlongpl: added \@glxstr@record	208
\glxstr@wrglossarylocation: new ..	330	\GLSxtrshort: added \@glxstr@record	204
\GlsXtrBibTeXEntryAliases: new ..	331	\Glsxtrshort: added \@glxstr@record	203
\glxstrfieldforlistloop: corrected argument order in \forlistcsloop	31	\glxtrshort: added \@glxstr@record	203
\GlsXtrIndexCounterLink: new	330	\glxtrshort: added \@glxstr@record	203
\GlsXtrInternalLocationHyperlink: new	23	\GLSxtrshortpl: added \@glxstr@record	208
\GlsXtrProvideBibTeXFields: new ..	331	\Glsxtrshortpl: added \@glxstr@record	207
indexcounter: new	23	\glxtrshortpl: added \@glxstr@record	207
\setentrycounter: new	127	\GlsXtrStartUnsetBuffering: new ..	97
1.30 (2018-04-25)		\GlsXtrStopUnsetBuffering: new ...	98
\@@glxstr@record: added check for post-key hook	9	indexcounter: added check for wrglossary counter	23
added check for pre-key hook	9	\s@GlsXtrStopUnsetBuffering: new ..	98
\@GLSxtr@fullpl: added \@glxstr@record	202	1.31 (2018-05-09)	
\@GlsXtrStopUnsetBuffering: new ..	98	\@GlsXtrStartUnsetBuffering: new ..	97
\@Glsxtr@fullpl: added \@glxstr@record	202	\@gl@ifaccessattribute@set: new	161
\@glxstr@dorecord: don't suppress expansion of \@glarecordlocoref ..	11	\@gl@initaccesskeys: new ...	160, 166
\@glxstr@full: added \@glxstr@record	199	\@gl@setup@default@short@access: new	161, 166
\@glxstr@fullpl: added \@glxstr@record	201	\@glxstr@record@noglossarywarning: new	133
\@glxstr@glossadd@postkeys: new ..	10	\@glxstrbuffer@nodup@unset: new ..	98
\@glxstr@glossadd@prekeys: new ...	10	General: added prefix key for glslink	64
\@glxstr@glslink@postkeys: new ...	10	added prefix key for printgloss ..	121
\@glxstr@glslink@prekeys: new	10	changed \let to \def	121
\@glxstr@local@textformat: new ...	64	\glsaddeach: new	67
\@glxstr@unset: new	97	\glsapturedgroup: new	330
\@glxstrbuffer@unset: new	98	\glsdefpostdesc: new	188
\glsadd: added \glsaddpostsetkeys	67	\glsdefpostlink: new	189
added \glsaddpresetkeys	67	\glsdefpostname: new	178
\glsaddpostsetkeys: new	67	\glsdohypertarget: bug fix: ensure that new version is picked up	121
\glsaddpresetkeys: new	67	\glslistdesc: new	367
\glsuserdescription: new	284	\glslocalreseteach: new	99
\glxtrabbreviationfont: new	60	\glslocalunseteach: new	99
\GlsXtrDualBackLink: new	331	\glstreechilddesc: new	380
\GlsXtrDualField: new	330	\glstreechildsymbol: new	380
\GlsXtrExpandedFmt: new	64	\glstreedefaultnamefmt: new	377
\GLSxtrlong: added \@glxstr@record	206	\glstreegroupheaderfmt: added redefinition	377
\Glsxtrlong: added \@glxstr@record	205	\glstreenamefmt: added redefinition	377
\glxtrlong: added \@glxstr@record	205		
\GLSxtrlongpl: added \@glxstr@record	210		

\glstreenavigationfmt:added		\glsxtrpostlinkAddSymbolDescOnFirstUse:	
redefinition	378	new	190
\glstreenonamechilddesc:new	381	\GlsXtrRecordWarning:new	131
\glstreenonamesymbol:new	381	\glsxtrRevertTocMarks:new	310
\glstreesymbol:new	380	\GlsXtrStandaloneGlossaryType:	
\glsxtr@newabbreviation:added		new	137
\ExtraCustomAbbreviationFields		\GlsXtrStandaloneSubEntryItem:	
.....	195	new	137
\GlsXtrForUnsetBufferedList:new .	98	\s@GlsXtrStartUnsetBuffering:new	97
\GlsXtrIfFieldCmpNum:new	33	1.32 (2018-05-24)	
\GlsXtrIfFieldEqNum:new	32	\GlsXtrForeignText:new	35
\GlsXtrIfFieldEqXpStr:new	35	\GlsXtrForeignTextField:new	37
\GlsXtrIfFieldNonZero:new	32	\GlsXtrUnknownDialectWarning:new	37
\GlsXtrIfHasNonZeroChildCount:		1.33 (2018-07-26)	
new	330	\ifglused:added redefinition	27
\GlsXtrIfXpFieldEqXpStr:new	35		

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	
\\$	330
\.	16, 17, 189, 377
\@	51, 133
\@cGLS@	101, 110
\@cGLSpl@	101, 110
\@cGls@	101, 109
\@cGlspl@	101, 110
\@cgls@	101, 109
\@cglspl@	101, 109
\@do@wrglossary	8, 10, 116
\@do@wrglossary	12–14, 27, 67, 83
\@glo@assign@sortkey	122
\@glo@list	6
\@glo@type	139
\@glossarysec	402
\@gls@expand@field	29
\@glslocalreset	99
\@glslocalunset	98
\@glsreset	99
\@glsunset	97
\@glsxtr@autoindex@escspch	183, 184
\@glsxtr@checkspch	182, 184, 185
\@glsxtr@disabledflycommand	56
\@glsxtr@org@postdescription	120
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 136
\@glsxtrfmt	29
\@glsxtrp	92, 93
\@glsxtrp@postloctag	58
\@glsxtrp@preloctag	58, 59
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 47, 51, 116
\@newglossaryentry@defcounters	100
\@newglossaryentry@defunitcounters	108
\@par	383
\@ACRlong	89
\@ACRlongpl	89
\@ACRshort	89
\@ACRshortpl	89
\@Acrlong	89
\@Acrlongpl	89
\@Acrshort	88
\@Acrshortpl	89
\@GLS@	88, 104, 105, 150
\@GLSdesc@	72
\@GLSpl@	88, 104, 105, 150
\@GLSplural@	89
\@GLSSymbol@	73
\@GLSxtr@	89
\@GLSxtr@full	200
\@GLSxtr@fullpl	202
\@GLSxtr@p@acrlong@	89
\@GLSxtr@p@acrlongpl@	89
\@GLSxtr@p@acrshort@	89
\@GLSxtr@p@acrshortpl@	89
\@GLSxtr@p@long@	88
\@GLSxtr@p@longpl@	88
\@GLSxtr@p@plural@	88
\@GLSxtr@p@short@	88
\@GLSxtr@p@shortpl@	88
\@GLSxtr@p@text@	88
\@GLSxtrlong	88, 206
\@GLSxtrlongpl	88, 209, 210
\@GLSxtrp	96
\@GLSxtrshort	88, 204
\@GLSxtrshortpl	88, 208
\@Gls@	88, 103, 104, 149
\@Gls@acentryname	112
\@Gls@entry@field	80, 95, 179
\@Gls@entryname	112
\@GlsXtrEnableOnTheFly	53
\@GlsXtrStartUnsetBuffering	97
\@GlsXtrStopUnsetBuffering	98

\@Glspl@	88, 103, 104, 149	\@end@glxtr@addunused	50
\@Glsplural@	89	\@end@glxtr@gettype	118, 121
\@Glstext@	89	\@end@glxtr@usesee	45
\@Glsxtr	54, 55	\@end@glxtrifhyphenstart	292, 293
\@Glsxtr@full	200	\@endfortrue	32, 178, 213
\@Glsxtr@fullpl	201	\@firstofone	68, 139, 173, 180, 186
\@Glsxtr@p@acrlong@	89	\@firstofthree	62,
\@Glsxtr@p@acrlongpl@	89		68, 76–79, 85, 200, 201, 203, 205, 207, 209
\@Glsxtr@p@acrshort@	88	\@firstoftwo	69–73, 77, 79, 82, 85, 114, 178,
\@Glsxtr@p@acrshortpl@	89		179, 191, 192, 200–202, 207–210, 311, 312
\@Glsxtr@p@long@	88	\@for	6, 22, 32, 50, 67, 99, 100, 112,
\@Glsxtr@p@longpl@	88		115, 118, 125, 139, 146, 152, 171, 178, 186
\@Glsxtr@p@plural@	88	\@glo@alias	47, 48
\@Glsxtr@p@short@	88	\@glo@assign@sortkey	118
\@Glsxtr@p@shortpl@	88	\@glo@autosee	25
\@Glsxtr@p@text@	88	\@glo@autoseehook	48
\@Glsxtrlong	88, 205	\@glo@category	106
\@Glsxtrlongpl	88, 209	\@glo@check@sortallowed	118
\@Glsxtrp	95	\@glo@counterprefix	10, 11, 127
\@Glsxtrpl	55	\@glo@countunit	106
\@Glsxtrshort	88, 203	\@glo@default@sorttype	118
\@Glsxtrshortpl	88, 207	\@glo@desc	39
\@acrlong	89	\@glo@descplural	39, 40
\@acrlongpl	89	\@glo@group	13
\@acrshort	88	\@glo@label	
\@acrshortpl	89		12, 13, 28, 45, 48–50, 80, 87, 386–392
\@addtoreset	151	\@glo@location	12
\@afterheading		\@glo@loclist	12
	368, 369, 379, 381–383, 395–398, 405	\@glo@name	181
\@alt@glshyp@opt	85	\@glo@no@assign@sortkey	121
\@auxout	11, 12, 51, 59,	\@glo@parent	387, 388
	102, 110, 115, 116, 128, 129, 133–136, 406	\@glo@see	45, 46, 48–50
\@bibgl@restoreat	133	\@glo@seealso	47, 48
\@cGLS	104	\@glo@sort	181
\@cGLS@	101, 104, 110	\@glo@sorttype	118, 125
\@cGLSpl	105	\@glo@text	62
\@cGLSpl@	101, 105, 110	\@glo@thislettergrp	142
\@cGls@	101, 109	\@glo@thisvalue	283
\@cGlspl@	101, 110	\@glo@tmp	28, 46, 80
\@cgls@	101, 109	\@glo@type	50, 86, 112, 115, 118,
\@cglspl@	101, 109		119, 121, 125, 126, 128, 129, 131, 132, 139
\@disable@onlypremakeg	116	\@glo@types	169, 170, 385–392
\@do@auxoutstuff	128, 129	\@glossary@default@style	56, 119, 400
\@do@gl@getcounterprefix	10, 11	\@glossarystyle	119
\@do@gl@see	48, 49	\@gls@	88, 103, 104, 148
\@do@newglossaryentry	112, 113, 197	\@gls@@link	62
\@do@seeglossary	13, 14, 25, 51, 116	\@gls@ReturnAfterFi	127
\@do@wrglossary	66, 147	\@gls@actualchar	182
\@empty	68, 76–79, 120, 127, 182, 200–210	\@gls@adjustmode	67

<code>\@gls@alt@hyp@opt</code>	85	<code>\@gls@longpl</code>	193, 195, 196
<code>\@gls@alt@hyp@opt@char</code>	85	<code>\@gls@map</code>	178
<code>\@gls@alt@hyp@opt@keys</code>	85	<code>\@gls@nameaccess</code>	160, 162
<code>\@gls@automake</code>	118	<code>\@gls@nohyperlist</code>	40, 42
<code>\@gls@between</code>	125	<code>\@gls@noidx@do</code>	125
<code>\@gls@checkedmidx</code>	182, 184	<code>\@gls@noidx@getgrouptitle</code>	139
<code>\@gls@checkmidxchars</code>	47, 181	<code>\@gls@noidx@nosanitizesort</code>	117
<code>\@gls@codepage</code>	129	<code>\@gls@noidx@sanitizesort</code>	117
<code>\@gls@counter</code>	9, 11, 23, 64, 67, 83, 147	<code>\@gls@noidx@loclist@finalsep</code>	122
<code>\@gls@currentlettergroup</code> ...	125, 139, 142	<code>\@gls@noidx@loclist@prev</code>	122
<code>\@gls@declareoption</code>	5	<code>\@gls@noidx@loclist@sep</code>	122
<code>\@gls@default@longpl</code>	195, 196	<code>\@gls@noref@warn</code>	116, 126
<code>\@gls@doautomake</code>	118, 135	<code>\@gls@org@glsnoidx@displayloc</code> ..	122, 123
<code>\@gls@doautomake@err</code>	135	<code>\@gls@org@glssseeformat</code>	122, 123
<code>\@gls@enablesavenonumberlist</code>	51	<code>\@gls@preglossaryhook</code>	119, 186
<code>\@gls@encapchar</code>	182	<code>\@gls@prevlevel</code>	393–395, 399, 400
<code>\@gls@entry@count</code>	101, 102	<code>\@gls@quotechar</code>	182
<code>\@gls@entry@field</code>		<code>\@gls@reference</code>	51, 52, 115, 116
...	28, 29, 33, 48, 80, 93–96, 101, 137, 138	<code>\@gls@restoreat</code>	51
<code>\@gls@entry@unitcount</code>	110	<code>\@gls@saveentrycounter</code> ..	13, 14, 27, 65, 67, 147
<code>\@gls@field@font</code>	68–75	<code>\@gls@see@noindex</code>	26, 133
<code>\@gls@field@link</code>	68–75, 80, 81	<code>\@gls@setdefault@glslink@opts</code>	
<code>\@gls@firstaccess</code>	160, 163	9, 30, 65, 84
<code>\@gls@getcounterprefix</code>	10, 11	<code>\@gls@setsort</code>	65, 67
<code>\@gls@getgrouptitle</code>	124, 139	<code>\@gls@setupsort@none</code>	14
<code>\@gls@grptitle</code>	86, 125	<code>\@gls@short</code>	195, 196
<code>\@gls@hyp@opt</code>		<code>\@gls@shortaccess</code>	160–163
...	80, 81, 85, 104, 105, 144, 148–150, 199–209	<code>\@gls@shortaccesspl</code>	160–162
<code>\@gls@hyp@opt@cs</code>	85	<code>\@gls@shortpl</code>	193, 196
<code>\@gls@ifaccessattribute@set</code>	161	<code>\@gls@sort</code>	142
<code>\@gls@ifinlist</code>	141	<code>\@gls@textaccess</code>	160, 162
<code>\@gls@increment@currcount</code>	101	<code>\@gls@thislabel</code>	67, 99
<code>\@gls@increment@currunitcount</code>	109	<code>\@gls@thisval</code>	178
<code>\@gls@initaccesskeys</code>	195	<code>\@gls@tmp</code>	35, 125
<code>\@gls@keymap</code> ..	12, 13, 28, 45, 47, 80, 134, 178	<code>\@gls@tmpb</code>	184
<code>\@gls@label</code> ..	8, 9, 11, 51, 84, 85, 116, 136, 213	<code>\@gls@type</code>	116–118, 213, 386–392
<code>\@gls@levelchar</code>	182	<code>\@gls@write@entrycounts</code>	101
<code>\@gls@link</code>	30, 61–63, 76–79, 200–210	<code>\@gls@write@entryunitcounts</code>	110
<code>\@gls@link@checkfirsthyper</code>	62, 114	<code>\@gls@write@entryunitcounts@do</code>	111
<code>\@gls@link@label</code>	64, 147	<code>\@gls@xref</code>	12, 47
<code>\@gls@link@nocheckfirsthyper</code>		<code>\@gls@abbrv@current@abbreviation</code> ..	195, 210
.....	61, 75–79, 199–210	<code>\@gls@sacronymlists</code>	112
<code>\@gls@link@opts</code>	64	<code>\@gls@doifexistsorwarn</code> ...	15, 174–177, 179
<code>\@gls@list</code>	125	<code>\@gls@entry</code>	102, 110, 111
<code>\@gls@local@increment@currcount</code> ...	101	<code>\@gls@link</code>	66, 86–88
<code>\@gls@local@increment@currunitcount</code> ..	109	<code>\@gls@localreset</code>	99
<code>\@gls@location</code>	142, 143	<code>\@gls@localunset</code>	98, 99
<code>\@gls@loclist</code>	122, 123, 142, 143	<code>\@gls@nextpages</code>	119
<code>\@gls@long</code>	195	<code>\@gls@nonextpages</code>	119

\@glsnumberformat	\@glsxtr@activate@initialtagging ..
..... 9, 11, 64, 67, 83, 147, 178, 181 186, 187
\@glsorder	\@glsxtr@addunitcounter
..... 115 106
\@glspl@	\@glsxtr@addunused
..... 88, 103, 104, 148 50
\@glsplural@	\@glsxtr@addunusedxrefs
..... 89 50
\@glsplunc@token	\@glsxtr@attrval .. 65, 66, 172–177, 179, 181
..... 191	\@glsxtr@autoindex@at
\@glsrecordlocref 181–183
..... 10, 11	\@glsxtr@autoindex@doextra@esc 181
\@glsshowtarget	\@glsxtr@autoindex@encap
..... 87 181–183
\@glsstyle@altlist	\@glsxtr@autoindex@esc 182, 184, 185
..... 367	\@glsxtr@autoindex@escat
\@glsstyle@altlistgroup 182, 183
..... 368	\@glsxtr@autoindex@escencap ... 182, 183
\@glsstyle@altlisthypergroup	\@glsxtr@autoindex@esclevel ... 182, 183
..... 369	\@glsxtr@autoindex@escquote ... 182, 184
\@glsstyle@alttree	\@glsxtr@autoindex@level
..... 383 182, 183
\@glsstyle@alttreegroup	\@glsxtr@autoindex@setname
..... 394 181
\@glsstyle@alttreehypergroup	\@glsxtr@autoindexcrossrefs 14, 15, 45, 48
..... 395	\@glsxtr@autoseeindexfalse
\@glsstyle@index 14
..... 378	\@glsxtr@autoseeindextrue
\@glsstyle@indexgroup 16
..... 379	\@glsxtr@bookindex@atendgroup . 403–405
\@glsstyle@indexhypergroup	\@glsxtr@bookindex@atsubendgroup .. 404
..... 379	\@glsxtr@bookindex@atsubsubendgroup 404
\@glsstyle@inline	\@glsxtr@bookindex@between 403, 405
..... 377	\@glsxtr@bookindex@sep
\@glsstyle@list 403, 404
..... 367	\@glsxtr@bookindex@subatendgroup ..
\@glsstyle@listdotted 403–405
..... 366	\@glsxtr@bookindex@subbetween . 403, 404
\@glsstyle@listgroup	\@glsxtr@bookindex@subsep
..... 368 403, 404
\@glsstyle@listhypergroup	\@glsxtr@bookindex@subsubatendgroup
..... 368 403–405
\@glsstyle@mccolalttree	\@glsxtr@bookindex@subsubbetween ..
..... 398 403, 404
\@glsstyle@mccolalttreegroup	\@glsxtr@bookindex@groupskip ... 403, 405
..... 399	\@glsxtr@cat
\@glsstyle@mccolalttreehypergroup .. 399 100, 112, 146, 186
\@glsstyle@mccolalttreesspannav 400	\@glsxtr@checkgroup
\@glsstyle@mccolindexgroup 139
..... 395	\@glsxtr@counterrecordhook
\@glsstyle@mccolindexhypergroup 395 11
\@glsstyle@mccolindexspannav	\@glsxtr@csname
..... 396 107, 109
\@glsstyle@mccoltreegroup	\@glsxtr@current@style
..... 396 56, 400
\@glsstyle@mccoltreehypergroup	\@glsxtr@currentunitcount
..... 397 107, 109
\@glsstyle@mccoltreenamegroup 397	\@glsxtr@currunitcount
\@glsstyle@mccoltreenamehypergroup 398 108, 110
\@glsstyle@mccoltreenameespannav .. 398	\@glsxtr@debugnr
\@glsstyle@mccoltreesspannav 24
..... 397	\@glsxtr@debugval
\@glsstyle@tree 24
..... 379	\@glsxtr@declareoption ... 5, 16, 18, 21, 23
\@glsstyle@treegroup	\@glsxtr@defaultnoglossarywarning .. 21
..... 380	\@glsxtr@defaultnumberformat
\@glsstyle@treehypergroup 7, 9, 64, 67, 83, 178, 181
..... 381	\@glsxtr@defpostpunc
\@glsstyle@treename 16, 17, 25
..... 381	
\@glsstyle@treenamegroup	
..... 382	
\@glsstyle@treenamehypergroup ... 382	
\@glstarget	
..... 87, 88, 121	
\@glstext@	
..... 89	
\@glsunset	
..... 97, 98	
\@glswidestname	
..... 385, 393	
\@glsxtr	
..... 54, 55	
\@glsxtr@@do@@wrglossary	
..... 116	
\@glsxtr@abbreviationsdef	
..... 18, 26	
\@glsxtr@accessdisplay	
..... 178–180	

<code>\@glsxtr@deprecated@abbrstyle</code>	<code>\@glsxtr@glossnamefont</code> .
242, 244, 245,	174–177, 179, 180
247, 256, 258, 259, 261, 274, 277, 281, 283	<code>\@glsxtr@gobbleto@endescpsh</code>
<code>\@glsxtr@dialect</code>	184
36, 37	<code>\@glsxtr@groupheading</code>
<code>\@glsxtr@disabledflycommand</code>	139, 142
55	<code>\@glsxtr@idx@displaynumberlist</code>
<code>\@glsxtr@display@loc</code>	117
126	<code>\@glsxtr@idx@entrynumberlist</code>
<code>\@glsxtr@do@wrindex</code>	117
84	<code>\@glsxtr@ifcsstart</code>
<code>\@glsxtr@do@glsgldisablehyperinlist</code> ..	53
82	<code>\@glsxtr@ifpunctoken</code>
<code>\@glsxtr@do@inc@linkcount</code>	192
152	<code>\@glsxtr@ifunitcounter</code>
<code>\@glsxtr@do@record@wrglossary</code>	106
8, 14	<code>\@glsxtr@insert@dots</code>
<code>\@glsxtr@do@redef@forglsentries</code>	193
7	<code>\@glsxtr@insert@dots@next</code>
<code>\@glsxtr@do@style</code>	193, 194
22, 329	<code>\@glsxtr@insertdots</code>
<code>\@glsxtr@do@titlecaps@warn</code>	161, 195
173–176, 179, 186	<code>\@glsxtr@label</code>
<code>\@glsxtr@doabbreviationsdef</code>	32, 50, 152, 171, 172
18	<code>\@glsxtr@loadstyles</code>
<code>\@glsxtr@doaccsupp</code>	365, 366
21, 24	<code>\@glsxtr@local@textformat</code>
<code>\@glsxtr@docdefsetting</code>	65, 66
15	<code>\@glsxtr@locale</code>
<code>\@glsxtr@docdefval</code>	36, 37
15, 51, 52	<code>\@glsxtr@longnewglossaryentry</code>
<code>\@glsxtr@docounterrecord</code>	39
11	<code>\@glsxtr@mark@wordseps</code>
<code>\@glsxtr@doglossary</code>	194
139, 140	<code>\@glsxtr@mark@wordseps@next</code>
<code>\@glsxtr@doiflabelinlist</code>	194
141	<code>\@glsxtr@markwordseps</code>
<code>\@glsxtr@dolocktag</code>	195, 196
58, 59	<code>\@glsxtr@mixed@assign@sortkey</code>
<code>\@glsxtr@dorecord</code>	118
8, 10	<code>\@glsxtr@noidx@displaynumberlist</code> ..
<code>\@glsxtr@dorecordnodefer</code>	117
8, 10	<code>\@glsxtr@noidx@do</code>
<code>\@glsxtr@dosee@alsoindex@glossary</code> ..	141
14	<code>\@glsxtr@noidx@entrynumberlist</code>
<code>\@glsxtr@doseeglossary</code>	117
13, 25	<code>\@glsxtr@noidx@numberlistloop</code>
<code>\@glsxtr@dostylewarn</code>	117
213	<code>\@glsxtr@nomissingglstextnr</code>
<code>\@glsxtr@enabletagging</code>	21
185	<code>\@glsxtr@nomissingglstextval</code>
<code>\@glsxtr@end@</code>	21
53	<code>\@glsxtr@noop@recordcounter</code>
<code>\@glsxtr@endescspsh</code>	11, 13
182–185	<code>\@glsxtr@nopostpunc</code>
<code>\@glsxtr@entrycount@org@localreset</code> 101	<code>\@glsxtr@nopostpunc@postdesc</code>
<code>\@glsxtr@entrycount@org@localunset</code> 101	119
<code>\@glsxtr@entrycount@org@reset</code>	120
101	<code>\@glsxtr@notfoundinlist</code>
<code>\@glsxtr@entrycount@org@unset</code>	192
101	<code>\@glsxtr@op@recordcounter</code>
<code>\@glsxtr@entryunitcount@org@localreset</code>	14
109	<code>\@glsxtr@optlist</code>
<code>\@glsxtr@entryunitcount@org@localunset</code>	55
109	<code>\@glsxtr@org@@starttoc</code>
<code>\@glsxtr@entryunitcount@org@reset</code> .	310
109	<code>\@glsxtr@org@GLS@</code>
<code>\@glsxtr@entryunitcount@org@unset</code> .	62
109	<code>\@glsxtr@org@GLSpl@</code>
<code>\@glsxtr@err@undefaction</code>	62
7, 13	<code>\@glsxtr@org@Gls@</code>
<code>\@glsxtr@field@linkdefs</code>	61, 62
61	<code>\@glsxtr@org@Glspl@</code>
<code>\@glsxtr@format@overridefalse</code>	62
180	<code>\@glsxtr@org@Glsxtrtitlefirst</code> .
<code>\@glsxtr@format@overridetrue</code>	311, 312
180	<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>
<code>\@glsxtr@foundinlist</code>	311, 313
192	<code>\@glsxtr@org@Glsxtrtitlefull</code> ..
<code>\@glsxtr@full</code>	311, 313
199	<code>\@glsxtr@org@Glsxtrtitlefullpl</code> 311, 313
<code>\@glsxtr@fullpl</code>	<code>\@glsxtr@org@Glsxtrtitlelong</code> ..
201	311, 313
<code>\@glsxtr@gettype</code>	<code>\@glsxtr@org@Glsxtrtitlelongpl</code> 311, 313
118	<code>\@glsxtr@org@Glsxtrtitlename</code> ..
<code>\@glsxtr@glossdescfont</code>	311, 312
172–174	<code>\@glsxtr@org@Glsxtrtitleplural</code> 311, 312
	<code>\@glsxtr@org@Glsxtrtitleshort</code> .
	311, 312
	<code>\@glsxtr@org@Glsxtrtitleshortpl</code> 311, 312
	<code>\@glsxtr@org@Glsxtrtitletext</code> ..
	311, 312

<code>\@glsxtr@org@MakeUppercase</code>	311, 312	<code>\@glsxtr@pagetag</code>	58, 59
<code>\@glsxtr@org@checkfirsthyper</code> ...	81, 114	<code>\@glsxtr@prevunitcount</code>	108
<code>\@glsxtr@org@currentfieldvalue</code>	36	<code>\@glsxtr@printglossnr</code>	120, 121
<code>\@glsxtr@org@delimN</code>	59	<code>\@glsxtr@printglossopts</code>	55, 118, 120
<code>\@glsxtr@org@delimR</code>	59	<code>\@glsxtr@printglossval</code>	120
<code>\@glsxtr@org@doseeglossary</code>	25, 116	<code>\@glsxtr@printunsrtglossaryskipentry</code>	139, 140
<code>\@glsxtr@org@glautosee</code>	26	<code>\@glsxtr@provide@addstoragekey</code>	29
<code>\@glsxtr@org@glolinkprefix</code>	65, 66	<code>\@glsxtr@provide@storagekey</code>	28
<code>\@glsxtr@org@gls@</code>	61	<code>\@glsxtr@record</code>	13, 14, 61–63, 67, 199, 201–210
<code>\@glsxtr@org@glsdohypertarget</code>	121	<code>\@glsxtr@record@noglossarywarning</code> ..	14
<code>\@glsxtr@org@glsgignore</code>	59	<code>\@glsxtr@record@setting</code>	8, 10, 13, 47, 51, 52, 115
<code>\@glsxtr@org@glspl@</code>	61	<code>\@glsxtr@record@setting@alsoindex</code> .	8, 10, 47, 115
<code>\@glsxtr@org@glstrtrtitlefirst</code> .	311, 312	<code>\@glsxtr@record@setting@off</code>	51
<code>\@glsxtr@org@glstrtrtitlefirstplural</code>	311, 312	<code>\@glsxtr@record@setting@only</code>	115
<code>\@glsxtr@org@glstrtrtitlefull</code> ..	311, 313	<code>\@glsxtr@recordsee</code>	14, 25, 47
<code>\@glsxtr@org@glstrtrtitlefullpl</code> ..	311, 313	<code>\@glsxtr@redef@forglsentries</code>	7, 26
<code>\@glsxtr@org@glstrtrtitlelong</code> ..	311, 313	<code>\@glsxtr@redefstyles</code>	21, 22, 329
<code>\@glsxtr@org@glstrtrtitlelongpl</code> ..	311, 313	<code>\@glsxtr@reg@glosslist</code>	115–118, 121
<code>\@glsxtr@org@glstrtrtitlename</code> ..	311, 312	<code>\@glsxtr@restore@postpunc</code>	120
<code>\@glsxtr@org@glstrtrtitleorpdforheading</code>	311, 312	<code>\@glsxtr@rglstrtrigger@record</code> ...	148–150
<code>\@glsxtr@org@glstrtrtitleplural</code> ..	311, 312	<code>\@glsxtr@s@longnewglossaryentry</code>	39
<code>\@glsxtr@org@glstrtrtitleshort</code> .	311, 312	<code>\@glsxtr@savepreloctag</code>	58, 59
<code>\@glsxtr@org@glstrtrtitleshortpl</code> ..	311, 312	<code>\@glsxtr@setentrycountunsetattr</code> ...	100
<code>\@glsxtr@org@glstrtrtitletext</code> ..	311, 312	<code>\@glsxtr@setentryunitcountunsetattr</code> ..	111
<code>\@glsxtr@org@makeglossaries</code>	115	<code>\@glsxtr@setupshortcuts</code>	20, 21, 26
<code>\@glsxtr@org@markboth</code>	310	<code>\@glsxtr@shortcutsnr</code>	20
<code>\@glsxtr@org@markright</code>	309, 310	<code>\@glsxtr@shortcutsval</code>	20, 135
<code>\@glsxtr@org@newacronymstyle</code> ..	113, 114	<code>\@glsxtr@swaptwo</code>	192
<code>\@glsxtr@org@postdescription</code> ..	120, 187	<code>\@glsxtr@tag</code>	186
<code>\@glsxtr@org@see@noindex</code>	133	<code>\@glsxtr@taggingcs</code>	186
<code>\@glsxtr@org@setacronymstyle</code> ..	113, 114	<code>\@glsxtr@textformat</code>	66
<code>\@glsxtr@org@theHvalue</code>	8, 9	<code>\@glsxtr@theHvalue</code> ...	8–10, 63, 65, 67, 147
<code>\@glsxtr@org@unset@buffer</code>	97, 98	<code>\@glsxtr@thevalue</code>	8–10, 63, 65–67, 147
<code>\@glsxtr@org@prefix</code>	10, 11	<code>\@glsxtr@thisloctag</code>	59
<code>\@glsxtr@org@printglossary</code>	55, 120	<code>\@glsxtr@titlelabel</code>	123, 124, 141
<code>\@glsxtr@org@warndep</code>	193	<code>\@glsxtr@tmp</code>	22, 64, 126, 127
<code>\@glsxtr@p@acrlong@</code>	89	<code>\@glsxtr@type</code>	172
<code>\@glsxtr@p@acrlongpl@</code>	89	<code>\@glsxtr@unitcountlist</code>	106
<code>\@glsxtr@p@acrshort@</code>	88	<code>\@glsxtr@unset</code>	97, 98
<code>\@glsxtr@p@acrshortpl@</code>	89	<code>\@glsxtr@unset@buffer</code>	97, 98
<code>\@glsxtr@p@long@</code>	88	<code>\@glsxtr@unsrt@getgroupptitle</code>	139
<code>\@glsxtr@p@longpl@</code>	88	<code>\@glsxtr@usesee</code>	45
<code>\@glsxtr@p@plural@</code>	88	<code>\@glsxtr@warn@onexistsordo</code>	7, 14
<code>\@glsxtr@p@short@</code>	88	<code>\@glsxtr@warn@undefaction</code>	7, 14
<code>\@glsxtr@p@shortpl@</code>	88		
<code>\@glsxtr@p@text@</code>	88		
<code>\@glsxtr@pagetag</code>	58, 59		

<code>\@glxstr@wrglossary@locationhyperlink</code>	<code>\@mfu@nocaplist</code>	187
..... 24	<code>\@one</code>	102, 111, 144
<code>\@glxstr@wrglossnr</code>	<code>\@newglossaryentry@defcounters</code>	100, 108
<code>\@glxstr@wrglossval</code>	<code>\@newglossaryentryposthook</code>
<code>\@glxstrbuffer@nodup@unset</code> 12, 13, 28, 48, 80	
<code>\@glxstrbuffer@unset</code>	<code>\@newglossaryentryprehook</code>
<code>\@glxstrdialecthook</code> 12, 13, 28, 39, 48, 80	
<code>\@glxstrdocdeffalse</code>	<code>\@nil</code>	127, 142
<code>\@glxstrentryfmt</code>	<code>\@nnil</code>	182, 184, 185, 192–194
<code>\@glxstrfmt</code>	<code>\@no@glxstrindexaliased</code>	83
<code>\@glxstrglossentry</code>	<code>\@no@makeglossaries</code>	132
<code>\@glxstrglossentryother</code>	<code>\@nocounterr</code>	152
<code>\@glxstrhypernameprefix</code>	<code>\@nopostdesc</code>	119
<code>\@glxstrifhasfield</code>	<code>\@onelevel@sanitize</code>	12, 47, 55, 124, 141
<code>\@glxstrifhyphenstart</code>	<code>\@onlypreamble</code>
<code>\@glxstrindexaliased</code>	.. 56, 59, 110, 133, 136, 152, 181, 183–185	
<code>\@glxstrindexcrossrefsfalse</code>	<code>\@org@glossaryentrynumbers</code>	119
<code>\@glxstrindexcrossrefstrue</code>	<code>\@org@newglossaryentryprehook</code>	39
<code>\@glxstrinmark</code>	<code>\@print@unsrt@glossary</code>	138
<code>\@glxstrlong</code>	<code>\@printgloss@setsort</code>	118, 119
<code>\@glxstrlongpl</code>	<code>\@printglossary</code>	55, 138
<code>\@glxstrnewgls</code>	<code>\@printunsrt@glossary@handler</code>	140
<code>\@glxstrnewgls@inner</code>	<code>\@printunsrtglossary</code>	138
<code>\@glxstrnewgls@innercsname</code>	<code>\@rGLS</code>	149
<code>\@glxstrnotinmark</code>	<code>\@rGLS@</code>	149
<code>\@glxstrp</code>	<code>\@rGLSpl</code>	150
<code>\@glxstrp@opt</code>	<code>\@rGLSpl@</code>	150
<code>\@glxstrpl</code>	<code>\@rGls</code>	148
<code>\@glxstrpostloctag</code>	<code>\@rGls@</code>	149
<code>\@glxstrpreloctag</code>	<code>\@rGlspl</code>	149
<code>\@glxstrsetaliasnoindex</code>	<code>\@rGlspl@</code>	149
<code>\@glxstrshort</code>	<code>\@rgls</code>	148
<code>\@glxstrshortpl</code>	<code>\@rgls@</code>	148
<code>\@glxstrundeftag</code>	<code>\@rglspl</code>	148
<code>\@glxstrwrglossmark</code>	<code>\@rglspl@</code>	148
<code>\@gobble</code>	<code>\@sGlsXtrEnableOnTheFly</code>	53
.. 7, 13, 16, 68, 140, 141, 193, 403–405	<code>\@secondofthree</code>	69,
<code>\@gobbletwo</code>	70, 76–79, 81, 200, 202, 204, 205, 207, 209	
<code>\@ifnextchar</code>	<code>\@secondoftwo</code>	62, 68,
<code>\@ifpackageloaded</code>	71–79, 82, 87, 114, 121, 178, 192, 199–	
5, 17, 135, 153, 172, 174, 176, 178, 180, 329, 333, 364	201, 203–210, 224, 246, 260, 282, 311, 312	
<code>\@ifstar</code>	<code>\@sglsxtr@provide@storagekey</code>	28
28, 29, 32, 39, 40, 42, 53, 85, 97, 98, 138, 185	<code>\@starttoc</code>	310
<code>\@ifundefined</code>	<code>\@thirdofthree</code>	68–71,
<code>\@ignored@glossaries</code>	76–79, 81, 201, 202, 204, 206, 208, 210, 312	
<code>\@input</code>	<code>\@thirdoftwo</code>	71–75
<code>\@input@</code>	<code>\@this@key</code>	178
<code>\@istfilename</code>	<code>\@tracklang@lang</code>	36, 37
<code>\@makeglossary</code>	<code>\@warn@nomakeglossaries</code>	129
<code>\@mfu@domakefirsttuc</code>		
186, 187		

\@xdy@main@language	128	\acl	19
\@xdycrossrefhook	46	\ACLP	19
\@xdy@language	128	\Aclp	19
\@xdy@locationclassorder	47	\aclp	19
\\	127	\ACP	19
		\Acp	19
		\acp	18
_	52, 130, 131	\ACRfullfmt	113
		\Acrfullfmt	113
		\acrfullfmt	113
		\ACRfullplfmt	113
		\Acrfullplfmt	113
		\acrfullplfmt	113
		\acronymentry	112
		\acronymfont	76–79, 91, 113, 114
		\acronymname	17
		\acronymsort	112
		\acronymtype	17, 112, 113
		\acrpluralsuffix	112, 135
		\ACS	19
		\Acs	19
		\acs	18
		\ACSP	19
		\Acsp	19
		\acsp	19
		\actualchar	184
		\addtolength	394
		\advance	102, 111, 134, 144
		\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	329, 364
		\appto	12, 13, 22, 28, 45–49, 80, 84, 100, 108, 139–141, 180, 191, 194, 365
		\arabic	23, 405
		\AS	18
		\As	18
		\as	18
		\ASP	18
A			
\AA	347		
\aa	347		
\AB	18		
\Ab	18		
\ab	18		
abbreviation styles:			
long-hyphen-postshort-hyphen	298, 300		
long-hyphen-short-hyphen	294, 299		
long-postshort-user	287		
long-short-user	285		
nolong-short	229		
short	227		
short-hyphen-long-hyphen	302, 304		
short-hyphen-postlong-hyphen	303, 306		
short-long-user	287		
short-nolong	226, 229		
short-nolong-desc	228		
short-postlong-user	289		
\abbreviationsname	17		
\abbrvpluralsuffix			
.	135, 161, 162, 196, 217, 219, 222, 224, 225, 227, 230, 234, 236, 237, 239, 241, 242, 244, 246, 248, 250, 251, 253, 255, 256, 258, 260, 262, 264, 266, 267, 269, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 291, 294, 296, 299, 302, 304, 307		
\ABP	18		
\Abp	18		
\abp	18		
\AC	19		
\Ac	19		
\ac	18		
\ACF	19		
\Acf	19		
\acf	19		
\ACFP	19		
\Acfp	19		
\acfp	19		
\ACL	19		
\Acl	19		

<code>\Asp</code>	18	<code>\cGLs</code>	18, 19, 100, 111
<code>\asp</code>	18	<code>\cglS</code>	18, 100, 111
<code>\AtBeginDocument</code>	24, 27, 57, 135	<code>\cGLSformat</code>	104
<code>\AtEndDocument</code>	49, 101, 110, 128, 129	<code>\cGLSformat</code>	103
B			
<code>babel</code> package	181, 183, 191	<code>\cGLSpl</code>	18, 19, 100, 111
<code>\begin</code>	125, 130, 131, 139, 367, 369–376, 379, 381, 396–400, 403	<code>\cGlspl</code>	18, 19, 100, 111
<code>\begingroup</code>	8, 9, 30, 83, 137, 138, 151	<code>\cglSpl</code>	18, 100, 111
<code>\bgroup</code>	39, 119	<code>\cGLSplformat</code>	104
<code>bib2gls</code> ...	23, 30, 142, 147, 148, 329, 330, 332	<code>\cGlsplformat</code>	103
C			
<code>\c@wrglossary</code>	23	<code>\cglSplformat</code>	103, 105
<code>\catcode</code>	51, 133	<code>\cGlspl</code>	18, 19, 100, 111
category attributes:		<code>\cglSpl</code>	18, 100, 111
<code>accessinsertdots</code>	161	<code>\cGLSplformat</code>	104
<code>apospplural</code>	196	<code>\cGlsplformat</code>	103
<code>discardperiod</code>	190	<code>\cglSplformat</code>	103, 105
<code>entrycount</code>	97, 99, 100, 102, 111	<code>\changes</code>	315, 379, 381
<code>firstshortaccess</code>	163	<code>\char</code>	124
<code>firstuc</code>	176	<code>\columnwidth</code>	57
<code>glossdesc</code>	172	<code>\count@</code>	102, 110, 111
<code>glossdescfont</code>	172	<code>\csappto</code>	38
<code>glossname</code>	174	<code>\csdef</code>	28, 33, 34, 38, 80, 81, 101, 106, 107, 109, 167, 178, 188, 189, 214, 215, 223, 246, 260, 281, 285, 287–289, 299, 301, 304, 306, 366, 384
<code>glossnamefont</code>	174, 176	<code>\cseappto</code>	43
<code>headuc</code>	313	<code>\csedef</code>	107
<code>indexname</code>	181	<code>\csgdef</code>	34, 40–42, 52, 58, 101, 107, 109, 110, 384, 406
<code>indexonlyfirst</code>	84	<code>\cslet</code>	34, 39, 40, 119
<code>insertdots</code>	195	<code>\csletcs</code>	34, 215
<code>linkcount</code>	151	<code>\csname</code>	6, 29, 41, 47, 51, 56, 60, 62, 64, 67, 76–81, 83, 92, 107, 116, 125, 128, 131, 132, 139, 144–147, 152, 172, 193, 200–210, 216, 393
<code>linkcountmaster</code>	151	<code>\cspreto</code>	38
<code>markshortwords</code>	195	<code>\csuse</code>	9, 30, 31, 41, 49, 58, 80, 81, 93–95, 106–110, 118, 123, 125, 126, 136, 137, 141, 142, 167, 178, 188, 189, 214, 215, 384, 385, 387, 388
<code>markwords</code>	195, 196, 292, 294, 302	<code>\csxdef</code>	45, 48, 107, 110
<code>nameshortaccess</code>	162	<code>\currentglossary</code>	119, 137, 138, 405
<code>nohyper</code>	82	<code>\CurrentOption</code>	24, 365, 366
<code>nohyperfirst</code>	69–71	<code>\CurrentTrackedLanguage</code>	363
<code>noshortplural</code>	196	<code>\CurrentTrackedLanguageTag</code>	135
<code>regular</code>	60, 105, 216–221, 223, 226, 228, 229, 231– 233, 235, 236, 238, 239, 242–244, 246, 249–253, 256–258, 260, 263–265, 267, 268, 270, 271, 273, 275, 277, 279–281, 284, 290–292, 294–297, 302, 303, 307, 309	<code>\CurrentTrackedScript</code>	363, 364
<code>textformat</code>	65	<code>\CurrentTrackedTag</code>	329, 364
<code>textshortaccess</code>	162	<code>\CustomAbbreviationFields</code>	197, 217–221, 223, 225, 227, 230, 232–238, 240, 244, 245, 248–251, 253, 255, 258, 260, 262, 263, 265–270, 272, 274, 277, 279, 281, 284, 285, 287, 289– 292, 294–297, 299, 300, 302–304, 306–308
<code>\cdot</code>	24		
<code>\centering</code>	402		
<code>\cGLS</code>	18, 19, 100, 111		

D

`\DeclareAcronymList` 112
`\DeclareOption` 5, 365
`\DeclareOptionX` 5, 24
`\def` 9, 10, 12–15, 25, 27,
 29, 30, 33, 39, 41, 45, 47, 48, 50, 53–55,
 57, 60–79, 85, 87–91, 97, 103–105, 112,
 116, 118–122, 124–128, 139, 141, 142,
 144, 147–150, 160–162, 182–187, 191–
 196, 199–210, 213, 293, 295, 298, 301,
 303, 304, 364, 377, 378, 393–395, 399, 400
`\defglentryfmt` 40–42
`\define@boolkey` 15, 16, 63, 82
`\define@choicekey`
 7, 13, 15, 16, 20, 21, 24, 63, 120
`\define@key` 12, 13, 16, 21,
 22, 28, 47, 63, 64, 66, 67, 80, 121, 160, 193
`\DefineAcronymSynonyms` 20
`\delimN` 59
`\delimR` 59
`\detokenize` 53
`\dimen@` 114, 384–392
`\dimen@i` 387, 388
`\dimen@ii` 384, 385, 387, 388
`\dimexpr` 57, 383
`\disable@keys` 17, 27, 52, 133
`\do` 6, 22, 32, 50, 67, 99, 100, 112,
 115, 118, 125, 139, 146, 152, 171, 178, 186
`\do@gl@link@checkfirsthyper`
 30, 61–63, 65, 75–79, 199–210
`\do@gl@disablehyperinlist` 65, 83
`doc package` 184
`\dolistcsloop` 31
`\DTLifinlist` 116, 117, 121
`\DTLifint` 124

E

`\eappto`
 11, 22, 40–42, 140, 142, 161–163, 181, 365
`\edef` 6, 8–11, 36, 40–43, 46,
 48, 49, 51, 64, 65, 67, 82, 83, 86, 87, 106,
 107, 109, 115, 116, 121, 124, 126–129,
 133, 137, 138, 147, 151, 172–175, 177–
 179, 182, 184, 193, 363, 387, 388, 403, 404
`\eglssetwidest` 386–392
`\egroup` 39, 40, 119
`\else` 8–12, 15–17, 20, 21, 25, 30,
 33, 38, 52, 53, 58, 60, 62, 66, 83, 84, 102,
 114, 115, 117, 119–121, 124, 126, 127,

 130, 132, 134, 147, 180–182, 184, 192–
 194, 196, 203–210, 212, 213, 217, 219,
 220, 222–231, 234, 236–250, 252–264,
 266, 268–283, 285, 286, 288, 289, 291–
 293, 295, 298–301, 304, 305, 307, 308,
 367, 369–380, 382, 393, 394, 400, 402, 404
`\emph` 261, 262
`\empty` 126, 127
`\encapchar` 184
`\end` 121, 125,
 130, 131, 140, 367, 369–376, 396–400, 403
`\end@gl@xtr@display@loc` 126
`\endcsname` 6, 29, 41,
 47, 51, 56, 60, 62, 64, 67, 76–81, 83, 92,
 107, 116, 125, 128, 129, 131, 132, 139,
 144–147, 152, 172, 193, 200–210, 216, 393
`\endgroup` 8, 10, 30, 83, 137, 138, 151
`\ensuremath` 24
entry categories:
 abbreviation 210
 general 166, 168
 index 170
`\epreto` 181
`\equal` 132, 189
`etoolbox package` 5
`\expandafter` 24, 29, 30, 32, 36,
 45, 46, 50, 53–55, 64, 80, 81, 85, 92, 98,
 105, 106, 116–118, 121, 125–127, 139,
 142, 144, 151, 161, 162, 172, 175, 176,
 178, 180, 181, 184, 192, 195, 196, 292, 403
`\expandonce` . 112, 142, 161–163, 182, 218, 296
`\ExtraCustomAbbreviationFields`
 161–163, 195, 197

F

`\fi` 7–12, 14–16, 18, 20, 21,
 24–26, 30, 33, 38, 45, 47–49, 51–53, 56–
 58, 60, 62, 63, 66, 83, 84, 102, 110, 111,
 114, 117–121, 124, 126, 127, 129, 130,
 132, 134, 135, 147, 180–182, 185, 192,
 194, 196, 203–210, 212, 213, 217, 219,
 220, 222–231, 234, 236–250, 252–264,
 266, 268–283, 285, 286, 288, 289, 291,
 293, 295, 298–301, 304, 305, 307, 308,
 367, 370–380, 382, 384–394, 400, 402, 405
first use 407
 flag 407
 text 407
`\firstacronymfont` 113, 114
`fontspec package` 135

`\footnote` 221
`\forall glossaries` 50, 139, 170, 172, 386–392
`\forall glossentries` 102, 111
`\forall TrackedDialect` 329, 364
`\foreignlanguage` 35, 37
`\forall glossentries` 6, 50, 170, 172, 386–392
`\forall listcsloop` 31, 111, 125
`\forall listloop` 98, 122, 123, 187
`\futurelet` 191

G

`\gdef` 59, 183, 184
`\Genacrfullformat` 113
`\genacrfullformat` 113
`\GenericAcronymFields` 113
`\Genplacrfullformat` 113
`\genplacrfullformat` 113
`\GetTrackedDialectFromLanguageTag` 35, 36
`\GetTrackedDialectToMapping` 36
`\glo@grabfirst` 142
`\glo@name` 175, 176, 180
`\gloaliaslabel` 87
`\global` 10, 39, 40, 119, 142
`\glo linkprefix` 64–66, 87, 121, 135
glossaries package
.... 14, 25, 26, 37, 44, 46, 47, 49, 118, 366
glossaries-accsupp package 21, 24, 153, 196
glossaries-extra package 2, 364
glossaries-extra-bib2gls package 14, 27, 329, 364
glossaries-extra-stylemods package . 21, 188, 329
glossaries-stylemods package 401
glossaries.sty package 40
`\GlossariesExtraWarning` 6,
16, 36–38, 53, 55, 66, 114, 116, 126,
130, 133, 139, 172–175, 177, 179, 186, 215
`\GlossariesExtraWarningNoLine` 16, 102, 111
`\GlossariesWarning` 58, 117, 119, 122, 123, 213
`\GlossariesWarningNoLine` 116, 129
glossary styles:
altlist 367
altlistgroup 368
altlisthypergroup 369
alttree 383, 384, 393
alttreegroup 394
alttreehypergroup 395
index 378
indexgroup 379
indexhypergroup 379
inline 377

list 367
listdotted 366
listdottedstyle 367
listgroup 368
listhypergroup 368
mcolalttree 398
mcolalttreegroup 399
mcolalttreehypergroup 399
mcolalttreespannav 400
mcolindexgroup 395
mcolindexhypergroup 395
mcolindexspannav 396
mcoltreegroup 396
mcoltreehypergroup 397
mcoltreenonamegroup 397
mcoltreenonamehypergroup 398
mcoltreenonamespannav 398
mcoltreespannav 397
sublistdotted 367
tree 379
treegroup 380
treehypergroup 381
treenoname 381
treenonamegroup 382
treenonamehypergroup 382
glossary-bookindex package 366
glossary-hypernav package 86
glossary-long package 371
glossary-longbooktabs package 371
glossary-tree package 377, 378
`\glossaryentrynumbers` 60, 119, 143
`\glossaryheader` 125,
139, 367–376, 378–382, 393, 395–399, 403
`\glossaryname` 118
`\glossarypostamble` 125, 140, 141
`\glossarypreamble` 125, 139
`\glossarysection` ... 125, 131, 133, 139, 141
`\glossarytitle` 41, 118, 119, 125, 131, 133, 139
`\glossarytoctitle` 41, 118, 125, 131, 133, 139
`\glossentry`
119, 143, 366–376, 378, 380, 382, 393, 403
`\glossentrydesc`
366, 367, 369–376, 379–381, 383
`\glossentryname`
137, 366–376, 378, 380, 382, 393, 394, 401
`\glossentrynameother` 138
`\glossentrysymbol` 370–374, 376, 380, 381, 383
`\glossxtrsetpopts` 188
`\glostyle` 379, 381

<code>\GLS</code>	100, 111, 146	<code>\Glsaccessfirst</code>	69
<code>\Gls</code>	54, 100, 111, 146	<code>\glsaccessfirst</code>	69
<code>\gls</code>	38, 54, 56, 100, 111, 117, 130, 146	<code>\GLSaccessfirstplural</code>	71
<code>\gls@assign@desc</code>	39	<code>\Glsaccessfirstplural</code>	71
<code>\gls@assign@field</code>	12, 13, 28, 80	<code>\glsaccessfirstplural</code>	70
<code>\gls@checkseeallowed</code>	52, 116	<code>\Glsaccesslong</code>	78,
<code>\gls@codepage</code>	129		197, 205, 217, 226, 230, 231, 234, 240,
<code>\gls@defdocnewglossaryentry</code> .	51, 100, 108		241, 243, 248, 254–257, 263, 264, 272–
<code>\gls@defglossaryentry</code>	39, 40, 54, 55		278, 285, 286, 294, 296, 297, 300, 302, 308
<code>\gls@dotocitle</code>	119	<code>\glsaccesslong</code> ...	78, 197, 205, 206, 217,
<code>\gls@glossary</code>	47		219, 220, 222–225, 227–231, 234, 236–
<code>\gls@grplabel</code>	86		245, 247, 248, 250, 252–259, 261, 262,
<code>\gls@ifnotmeasuring</code>	99		264, 266, 268, 269, 271–282, 285, 286,
<code>\gls@level</code>	142, 143		289, 291, 294, 296, 297, 300, 302, 307, 308
<code>\gls@noidxglossary</code>	116	<code>\Glsaccesslongpl</code>	
<code>\gls@org@glossaryentryfield</code>	119		.. 79, 198, 209, 217, 226, 231, 234, 240,
<code>\gls@org@glossarysubentryfield</code> ...	119		241, 243, 249, 254–257, 263, 264, 272–
<code>\gls@orgTrackLangRequireDialectPrefix</code>			278, 285, 286, 294, 296, 297, 300, 302, 308
	363, 364	<code>\glsaccesslongpl</code>	79,
<code>\gls@save@numberlist</code>	57, 58, 60		198, 209, 210, 217, 220, 222–231, 234,
<code>\gls@set@xr@key</code>	47		236–245, 247, 248, 250, 252–259, 261,
<code>\gls@tmplen</code>	386–392, 394		262, 264, 266, 268–283, 285, 286, 289,
<code>\gls@type</code>	116		291, 292, 294, 296, 297, 300, 302, 307, 308
<code>\glsabbrvdefaultfont</code> ...	199, 217, 219,	<code>\GLSaccessname</code>	71
	222, 224, 225, 227, 230, 283, 293, 296, 306	<code>\Glsaccessname</code>	71
<code>\glsabbrvmfont</code>		<code>\glsaccessname</code>	46, 71
	261–270, 272, 274, 276, 278, 280–282	<code>\GLSaccessplural</code>	70
<code>\glsabbrvfont</code>		<code>\Glsaccessplural</code>	70
	89, 90, 113, 199, 203, 204, 207, 208,	<code>\glsaccessplural</code>	70
	210, 212, 213, 216–225, 227, 230, 232,	<code>\Glsaccessshort</code>	76, 204, 213, 220,
	234, 236, 237, 239, 241, 242, 244, 246,		222–224, 228, 229, 236, 238, 239, 245–
	248, 250, 251, 253, 255, 256, 258, 260,		247, 250, 252, 253, 259, 261, 266, 268,
	262, 264, 266, 267, 269, 270, 272, 274,		269, 271, 280–282, 288, 289, 291, 300, 305
	276, 278, 280, 282, 284, 286, 288, 290–	<code>\glsaccessshort</code> ..	76, 197, 198, 203, 204,
	292, 294, 296, 298–300, 302, 304, 305, 307		212, 217, 219, 222, 224–229, 231, 234,
<code>\glsabbrvhyphenfont</code>			236–241, 243–248, 250–264, 266, 268–
	293–295, 299, 301–304, 306		273, 275–278, 280, 282, 285, 286, 288,
<code>\glsabbrvonlyfont</code>	306, 307, 309		289, 291, 294, 296, 297, 299, 302, 305, 308
<code>\glsabbrvscfont</code> .	233–239, 241, 242, 244–246	<code>\Glsaccessshortpl</code>	77, 207, 213, 220, 222–
<code>\glsabbrvsmfont</code>			225, 228, 229, 236, 238, 239, 245, 247,
	247–251, 253, 255, 256, 258, 260		250, 252, 253, 259, 261, 266, 268, 269,
<code>\glsabbrvuserfont</code>	283–289, 291		271, 280–283, 288, 289, 291, 300, 305, 308
<code>\GLSaccessdesc</code>	72	<code>\glsaccessshortpl</code>	
<code>\Glsaccessdesc</code>	72, 173, 185		77, 198, 207, 208, 213, 217–
<code>\glsaccessdesc</code>	71, 173, 189, 190		219, 222, 224–229, 231, 234, 236–250,
<code>\GLSaccessdescplural</code>	72		252–254, 256, 257, 259–264, 266, 268–
<code>\Glsaccessdescplural</code>	72		273, 275, 277, 278, 280–282, 285, 286,
<code>\glsaccessdescplural</code>	72		288, 289, 291, 294, 297, 299, 302, 305, 308
<code>\GLSaccessfirst</code>	69	<code>\GLSaccesssymbol</code>	73

<code>\Glsaccesssymbol</code>	73, 185	<code>\glsdonohyperlink</code>	66, 87, 88
<code>\glsaccesssymbol</code>	72, 185, 190	<code>\glsdosanitizesort</code>	117
<code>\GLSaccesssymbolplural</code>	73	<code>\glsenableentrycount</code>	100, 102, 110
<code>\Glsaccesssymbolplural</code>	73	<code>\glsenableentryunitcount</code>	101, 111
<code>\glsaccesssymbolplural</code>	73	<code>\glsentrycounter</code>	23, 127
<code>\GLSaccesstext</code>	68	<code>\GlsEntryCounterLabelPrefix</code>	37, 38
<code>\Glsaccesstext</code>	69	<code>\glsentrycurrcount</code>	101, 102, 108
<code>\glsaccesstext</code>	46, 68	<code>\Glsentrydesc</code>	157, 165, 173
<code>\glsacrshortcutstrue</code>	20, 21	<code>\glsentrydesc</code>	157, 165, 174
<code>\glsacspacemax</code>	114	<code>\Glsentrydescplural</code>	158, 165
<code>\glsadd</code>	30, 50, 67, 130	<code>\glsentrydescplural</code>	157, 158, 165
<code>\glsadd options</code>		<code>\Glsentryfirst</code>	105, 150, 155, 164
<code>theHvalue</code>	9	<code>\glsentryfirst</code>	105, 150, 155, 164, 325
<code>thevalue</code>	9, 10	<code>\Glsentryfirstplural</code> ...	106, 151, 155, 164
<code>\glsaddpostsetkeys</code>	10, 67	<code>\glsentryfirstplural</code>	105, 150, 155, 156, 164, 326
<code>\glsaddpresetkeys</code>	10, 67	<code>\glsentryfmt</code>	40–42
<code>\glsaddstoragekey</code>	49, 166, 331, 332	<code>\Glsentryfull</code>	113
<code>\glsbackslash</code>	53	<code>\glsentryfull</code>	113
<code>\glscapscase</code>	62, 68–79, 81, 200–212	<code>\Glsentryfullpl</code>	113
<code>\glscategory</code> ..	60, 68, 82, 89, 90, 167–169, 172–179, 185, 188, 189, 199–204, 207, 208	<code>\glsentryfullpl</code>	113
<code>\glscategorylabel</code>	82, 161–163, 193, 195, 196, 223, 246, 260, 281, 285, 287–289, 299, 301, 304, 306	<code>\glsentryitem</code>	137, 138, 366–376, 378, 380, 382, 393, 404
<code>\glsclsebrace</code>	46, 131, 132	<code>\Glsentrylong</code>	90, 91, 105, 150, 159, 166
<code>\glscounter</code>	9	<code>\glsentrylong</code>	90, 91, 105, 150, 159, 166, 223, 246, 260, 281, 288, 289, 304, 326
<code>\glscurrententrylabel</code>	58, 59, 119, 128, 137–140, 187, 188	<code>\Glsentrylongpl</code>	90, 91, 106, 160, 166
<code>\glscurrentfieldvalue</code>	30–33, 35, 36, 283, 330, 331	<code>\glsentrylongpl</code> ...	90, 91, 105, 160, 166, 327
<code>\glscustomtext</code> ..	61, 62, 76–79, 200–210, 212	<code>\Glsentrylongplural</code>	151
<code>\glsdefaulttype</code> ..	6, 17, 38, 118, 130, 138, 141	<code>\glsentrylongplural</code>	150
<code>\glsdescriptionaccessdisplay</code> ..	157, 173	<code>\Glsentryname</code>	153, 163, 174, 176, 177
<code>\glsdescriptionpluralaccessdisplay</code>	157, 158	<code>\glsentryname</code>	136, 137, 153, 163, 181, 323, 324, 386–392, 406
<code>\glsdescwidth</code>	369–376	<code>\glsentrynumberlist</code>	117, 123, 390–392
<code>\glsdetoklabel</code>	8, 9, 27, 31–35, 37–40, 43–46, 50–53, 64, 67, 83, 87, 101, 106–111, 116, 119, 122, 123, 137, 138, 142, 145–147, 172, 175, 176, 180, 387, 388	<code>\Glsentryplural</code>	154, 164
<code>\glsdisplaynumberlist</code>	117, 122	<code>\glsentryplural</code>	154, 163, 164, 324, 325
<code>\glsdohyperlink</code>	85, 86, 88	<code>\glsentryprevcount</code>	101, 102, 108
<code>\glsdohypertarget</code>	88, 121	<code>\glsentryprevmaxcount</code>	109
<code>\glsdoifexists</code>	15, 25, 27, 34, 43, 45–47, 61, 62, 67, 75– 79, 87, 99, 116, 122, 123, 137, 138, 199–210	<code>\glsentryprevtotalcount</code>	108
<code>\glsdoifexistsordo</code>	30, 63	<code>\Glsentryshort</code>	89, 91, 158, 165
<code>\glsdoifexistsorwarn</code>	15, 172, 173, 185	<code>\glsentryshort</code>	89– 91, 114, 158, 165, 285, 287, 298, 322, 323
<code>\glsdoifnoexists</code>	39	<code>\Glsentryshortpl</code>	90, 91, 159, 165
		<code>\glsentryshortpl</code>	90, 91, 159, 165, 166, 322, 323
		<code>\Glsentrysymbol</code>	156, 164
		<code>\glsentrysymbol</code>	156, 164, 389–391
		<code>\Glsentrysymbolplural</code>	157, 164
		<code>\glsentrysymbolplural</code> ..	156, 157, 164, 165

<code>\Glsentrytext</code>	154, 163	<code>\glsfirstlongfootnotefont</code> 221–225, 244–247, 258–261, 279–283
<code>\glsentrytext</code>	87, 153, 154, 163, 324	<code>\glsfirstlonghyphenfont</code>	293–304
<code>\glsentrytype</code>	137	<code>\glsfirstlongonlyfont</code>	307–309
<code>\Glsentryuseri</code>	74	<code>\glsfirstlonguserfont</code>	284–292
<code>\glsentryuseri</code>	74	<code>\GLSfirstplural</code>	318
<code>\Glsentryuserii</code>	74	<code>\Glsfirstplural</code>	318
<code>\glsentryuserii</code>	74	<code>\glsfirstplural</code>	318
<code>\Glsentryuseriii</code>	74	<code>\glsfirstpluralaccessdisplay</code> ..	155, 156
<code>\glsentryuseriii</code>	74	<code>\glsforeachincategory</code>	213
<code>\Glsentryuseriv</code>	74	<code>\glsgenentryfmt</code>	60
<code>\glsentryuseriv</code>	75	<code>\glsgetattribute</code>	65, 86, 102, 106–108, 128, 147, 151, 172–175, 177, 179, 181
<code>\Glsentryuserv</code>	75	<code>\glsgetcategoryattribute</code>	167
<code>\glsentryuserv</code>	75	<code>\glsgetgroupitle</code> 368, 369, 379, 381–383, 394–400
<code>\glsextrapostnamehook</code>	178	<code>\glsgetwidestname</code>	384
<code>\glsfieldfetch</code>	87	<code>\glsgroupheading</code> 142, 367–376, 378–383, 393–400, 405
<code>\glsfieldxdef</code>	171, 172	<code>\glsgroupskip</code>	142, 367, 369–377, 379, 380, 382, 394, 405
<code>\glsfindwidesttoplevelname</code>	385	<code>\glshasattribute</code> 65, 86, 102, 107, 109, 111, 128, 147, 151, 172–177, 179, 181, 217–221, 223, 227–229, 232–235, 237, 244, 246, 248–251, 258, 260, 262–265, 267, 268, 276, 279–281, 284, 285, 287, 288, 290–292, 294–297, 299, 301–304, 306, 307, 309
<code>\GLSfirst</code>	317	<code>\glshascategoryattribute</code>	168
<code>\Glsfirst</code>	318	<code>\glshex</code> ..	334–340, 342–348, 350–353, 355–363
<code>\glsfirst</code>	317	<code>\glshyperlink</code>	87, 331
<code>\glsfirstabbrvdefaultfont</code>	199, 217, 219, 222, 224, 225, 227, 230, 296	<code>\glshypernavsep</code>	125
<code>\glsfirstabbrvmfont</code>	262–283	<code>\glshypernumber</code>	128, 180
<code>\glsfirstabbrvfont</code>	113, 197, 198, 217–231, 234, 236, 237, 239, 241, 242, 244, 246, 248, 250, 251, 253, 255, 256, 258, 260, 262, 264, 266, 267, 269, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 291, 294, 296, 297, 299, 302, 304, 307	<code>\glsifattribute</code> 63, 64, 69, 82, 84, 93, 151, 170, 173–176, 179, 180, 187, 190, 191, 313–322
<code>\glsfirstabbrvhyphenfont</code> 293–295, 298, 299, 301–306	<code>\glsifcategory</code>	170
<code>\glsfirstabbrvonlyfont</code>	307, 308	<code>\glsifcategoryattribute</code> 82, 161–163, 168, 169, 195, 196
<code>\glsfirstabbrvscfont</code>	233–247	<code>\glsifnotregular</code>	68
<code>\glsfirstabbrvsmfont</code>	248–261	<code>\glsifnotregularcategory</code>	169
<code>\glsfirstabbrvuserfont</code>	284–292	<code>\glsifplural</code> 62, 68, 70–73, 75–79, 190, 191, 199–211
<code>\glsfirstaccessdisplay</code>	155	<code>\glsifregular</code>	60, 68, 105, 106, 150, 151
<code>\glsfirstlongdefaultfont</code>	217, 219, 225, 227, 230, 233–243, 248–258, 262, 263, 265, 266, 269–274, 276, 277	<code>\glsifregularcategory</code>	169
<code>\glsfirstlongemfont</code> 264, 265, 267, 268, 274, 275, 277–279	<code>\glsifusetranslator</code>	41
<code>\glsfirstlongfont</code> ...	197, 198, 217–220, 222, 224–232, 234, 236, 237, 239, 241, 242, 244, 246, 248, 250, 251, 253, 255, 256, 258, 260, 262, 264, 266, 267, 269, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 291, 294, 296, 299, 302, 304, 307	<code>\glsignore</code>	59
		<code>\glsinlinedescformat</code>	377
		<code>\glsinlinesubdescformat</code>	377

<code>\glsinsert</code>	62,	<code>\glslongfootnotefont</code>	
	68, 76–79, 200–212, 292, 299, 301, 304, 306		221, 222, 224, 244, 246, 258, 260, 280, 282
<code>\glskeylisttok</code>	112, 113, 194, 197	<code>\glslonghyphenfont</code>	
<code>\glslabel</code>	8, 9, 30, 43,		293, 294, 296, 297, 299, 302, 304
	46, 60, 63–66, 82, 83, 86, 87, 114, 147,	<code>\glslongonlyfont</code>	307
	151, 152, 188–190, 210–212, 223, 246,	<code>\glslongpltok</code>	
	260, 281, 285, 287–289, 299, 301, 304, 306		196, 197, 217–221, 230, 232, 233,
<code>\glslabeltok</code>	112, 194, 197, 217–223,		235, 236, 240, 244, 248–251, 255, 258,
	225, 227–230, 232–238, 241, 244, 246,		262–268, 272, 274, 277, 280, 284, 285,
	248–251, 253, 255, 258, 260, 262–270,		287–292, 294–297, 299, 301–303, 307, 309
	272, 274, 276, 278–282, 284, 285, 287–	<code>\glslongpluralaccessdisplay</code>	160
	292, 294–297, 299, 301–304, 306, 307, 309	<code>\glslongtok</code>	
<code>\glsletentryfield</code>	181		112, 195, 197, 217–221, 223, 225, 227,
<code>\glslink</code>	113		230, 232–238, 240, 241, 244, 245, 248,
<code>\glslink options</code>			249, 251, 253, 255, 258, 260, 262–270,
counter	10		272, 274, 277, 279, 281, 284, 285, 287–
format	180		292, 294–297, 299, 300, 302–304, 307–309
hyper	309	<code>\glslonguserfont</code>	284, 286–289, 291
hyperoutside	63, 64	<code>\glsmcols</code>	396–400
noindex	8, 9, 82, 309	<code>\GLSname</code>	315
textformat	65	<code>\Glsname</code>	315
theHvalue	65	<code>\glsname</code>	315
thevalue	65, 144	<code>\glsnameaccessdisplay</code> ..	153, 174, 175, 177
wrgloss	8, 63	<code>\glsnamefont</code>	174, 176, 177, 179
<code>\glslinkcheckfirsthyperhook</code>	82	<code>\glsnavhyperlink</code>	125
<code>\glslinkpostsetkeys</code>	10, 65, 147	<code>\glsnavhyperlinkname</code>	86
<code>\glslinkpresetkeys</code>	10, 65, 147	<code>\glsnavhypertarget</code>	
<code>\glslinkvar</code>	85		368, 369, 379, 381, 383, 395–400
<code>\glslistchildpostlocation</code>	367	<code>\glsnavigation</code>	
<code>\glslistchildprelocation</code>	367, 368		368, 369, 379, 381, 383, 395–400
<code>\glslistdesc</code>	367, 368	<code>\glsnextpages</code>	119
<code>\glslistdottedwidth</code>	366	<code>\glsnoidxdisplayloc</code>	122, 123
<code>\glslistgroupheaderfmt</code>	368, 369	<code>\glsnoidxdisplaylocclsthandler</code>	122
<code>\glslistnavigationitem</code>	368, 369	<code>\glsnoidxloclist</code>	123, 143
<code>\glslistprelocation</code>	367, 368	<code>\glsnoidxnumberlistloophandler</code>	123
<code>\glslocalunset</code>	62, 147	<code>\glsnonextpages</code>	119
<code>\glslongaccessdisplay</code>	159	<code>\glsnonumberlistfalse</code>	58
<code>\glslongdefaultfont</code>	199, 217, 219,	<code>\glsnonumberlisttrue</code>	58
	221, 225, 227, 230, 234, 236, 237, 239–	<code>\glsnopostdotfalse</code>	120
	243, 248, 250, 251, 253, 255–257, 262,	<code>\glsnopostdottrue</code>	120
	266, 269, 270, 272, 273, 276, 284, 293, 306	<code>\glsnumberlistloop</code>	117
<code>\glslongemfont</code> ..	262, 264, 267, 274, 277, 278	<code>\glsnumlistlastsep</code>	122
<code>\glslongfont</code>	90, 91, 199, 205, 206,	<code>\glsnumlistsep</code>	122
	209, 210, 217–220, 222, 224, 225, 227,	<code>\glsopenbrace</code>	46, 131, 132
	230–232, 234, 236, 237, 239, 241, 242,	<code>\glsorder</code>	115
	244, 246, 248, 250, 251, 253, 255, 256,	<code>\glspagelistwidth</code>	370, 372, 374, 376
	258, 260, 262, 264, 266, 267, 269, 270,	<code>\glspar</code>	141
	272, 274, 276, 278, 280, 282, 284, 286,	<code>\GLSpl</code>	100, 111, 146
	288, 291, 294, 296, 299, 302, 304, 307, 308	<code>\Glspl</code>	55, 100, 111, 146

<code>\glspl</code>	55, 100, 111, 146	<code>\glstextaccessdisplay</code>	153, 154
<code>\GLSplural</code>	316, 317	<code>\glstextformat</code>	63, 66
<code>\Glsplural</code>	317	<code>\glstextup</code>	233
<code>\glsplural</code>	316	<code>\glstreechilddesc</code>	378, 380
<code>\glspluralaccessdisplay</code>	154	<code>\glstreechildpredesc</code>	380
<code>\glspluralsuffix</code>	135, 195, 199	<code>\glstreechildprelocation</code> ...	378, 380, 382
<code>\glspostdescription</code>	16, 17, 120, 187, 366, 367, 369–376, 379–381, 383	<code>\glstreechildsymbol</code>	378, 380
<code>\glspostinline</code>	377	<code>\glstreedefaultnamefmt</code>	377, 378
<code>\glspostlinkhook</code> .	61–63, 76–79, 92, 200–210	<code>\glstreedesc</code>	378–380
<code>\glsprestandardsort</code>	117	<code>\glstreegroupheaderfmt</code>	379, 381–383, 394–400, 402
<code>\glsresetentrylist</code>	125, 139	<code>\glstreeindent</code>	380, 382, 392–394
<code>\glsee</code>	47–49	<code>\glstreeitem</code>	378, 396, 404
<code>\glseeformat</code>	45, 46, 51, 116, 122, 123	<code>\glstreenamebox</code>	393, 394
<code>\glseeelist</code>	46	<code>\glstreenamefmt</code>	377, 378, 380, 382, 384, 386–394
<code>\glsssetabbrvfmt</code>	60, 68, 89, 90, 172–177, 179, 185, 199–204, 207, 208, 210	<code>\glstreenavigationfmt</code>	379, 381, 383, 395–400
<code>\glsssetAttribute</code>	217– 223, 225, 227–230, 232–235, 237, 238, 241, 244, 246, 248–251, 253, 255, 258, 260, 262–265, 267–270, 272, 274, 276, 278–280, 282, 284, 285, 287, 288, 290– 292, 294–297, 299, 301–304, 306, 307, 309	<code>\glstreenonamechilddesc</code>	382
<code>\glsssetcategoryattribute</code>	100, 112, 114, 146, 152, 167, 168, 170, 171, 186	<code>\glstreenonamedesc</code>	381, 382
<code>\glsssetnoexpandfield</code>	12, 13	<code>\glstreenonamesymbol</code>	382
<code>\glsssettoctitle</code>	119	<code>\glstreepredesc</code>	379, 381
<code>\glssshortaccessdisplay</code>	158	<code>\glstreeprelocation</code> ...	378, 380, 382, 383
<code>\glssshortpltok</code>	196, 197, 217–221, 223, 225, 227, 234–238, 244, 245, 248–251, 253, 258, 260, 262–270, 279–281, 284, 285, 287– 292, 294, 295, 299, 301–304, 306, 307, 309	<code>\glstreesubitem</code>	378, 404
<code>\glssshortpluralaccessdisplay</code>	159	<code>\glstreesubsubitem</code>	378, 405
<code>\glssshorttok</code>	112, 195–197, 216–221, 223, 225, 227, 232, 233, 235–238, 240, 244, 245, 248, 249, 251, 253, 255, 258, 260, 262–270, 272, 274, 279, 281, 284, 285, 287, 289–292, 294, 295, 297, 299, 301–304, 306, 307, 309	<code>\glstreesymbol</code>	378, 380
<code>\glsssubentryitem</code>	137, 366–376, 378, 380, 382, 394, 404	<code>\GLstrLetField</code>	34
<code>\glsssymbolaccessdisplay</code>	156	<code>\glstype</code>	62, 64, 76–79, 147, 200–210
<code>\glsssymbolpluralaccessdisplay</code> .	156, 157	<code>\glunset</code>	50, 62, 103, 104, 147
<code>\glstarget</code>	137, 138, 366–376, 378, 380, 382, 393, 394, 404, 405	<code>\gluserdescription</code> ...	284, 285, 288, 291
<code>\GLStext</code>	315, 316	<code>\glswrite</code>	46, 115
<code>\Glstext</code>	316	<code>\glswriteentry</code>	8, 9
<code>\glstext</code>	315, 316	<code>\Glsxtr</code>	55
		<code>\glxtr</code>	55
		<code>\glxtr@@do@wrglossary</code>	8, 10, 12, 13
		<code>\glxtr@addloclistfield</code>	14
		<code>\glxtr@addunused</code>	50
		<code>\glxtr@applyabbrvfmt</code>	210
		<code>\glxtr@applyabbrvstyle</code> ...	193, 195, 214
		<code>\glxtr@counterrecord</code>	136
		<code>\glxtr@do@alsoindex@wrglossary</code> ...	14
		<code>\glxtr@doooption</code>	5, 16, 17, 23, 24, 26
		<code>\glxtr@fields</code>	134
		<code>\glxtr@headentry@p</code>	93, 94
		<code>\glxtr@hyperoutsidefalse</code>	64
		<code>\glxtr@hyperoutsidettrue</code>	63, 64
		<code>\glxtr@ifnextpunc</code>	191
		<code>\glxtr@ifpunctoken</code>	191
		<code>\glxtr@inc@linkcount</code>	65, 152
		<code>\glxtr@inc@wrglossaryctr</code> ...	8, 9, 23, 27

<code>\glxtr@indexonly@saveentrycounter</code>	<code>\glxtrAltTreeSetHangIndent</code> ...	383, 393
..... 13, 14, 27	<code>\glxtrAltTreeSetSubHangIndent</code> ...	394
<code>\glxtr@keylist</code>	<code>\glxtralttreeSubSymbolDescLocation</code>	394
<code>\glxtr@label</code>	<code>\glxtralttreeSymbolDescLocation</code> ..	
<code>\glxtr@langtag</code> 383, 394	
<code>\glxtr@linkprefix</code>	<code>\glxtrassignfieldfont</code>	68–75
<code>\glxtr@loaddialect</code>	<code>\glxtrautoindex</code>	181
<code>\glxtr@makeglossaries</code>	<code>\glxtrautoindexassignsort</code>	181
<code>\glxtr@newabbreviation</code>	<code>\glxtrautoindexentry</code>	181
<code>\glxtr@next</code>	<code>\glxtrbibaddress</code>	331
<code>\glxtr@org@do@wrglossary</code>	<code>\glxtrbibauthor</code>	331
<code>\glxtr@org@dohyperlink</code>	<code>\glxtrbibbooktitle</code>	331
<code>\glxtr@org@getgrouptitle</code>	<code>\glxtrbibchapter</code>	331
<code>\glxtr@org@newignoredglossary</code>	<code>\glxtrbibedition</code>	331
<code>\glxtr@org@makenoidxglossaries</code>	<code>\glxtrbibhowpublished</code>	331
<code>\glxtr@pluralsuffixes</code>	<code>\glxtrbibinstitution</code>	331
<code>\glxtr@process</code>	<code>\glxtrbibjournal</code>	331
<code>\glxtr@provideignoredglossary</code>	<code>\glxtrbibmonth</code>	331
<code>\glxtr@punclist</code>	<code>\glxtrbibnote</code>	331
<code>\glxtr@record</code>	<code>\glxtrbibnumber</code>	331
<code>\glxtr@record@nr</code>	<code>\glxtrbiborganization</code>	331
<code>\glxtr@recordsee</code>	<code>\glxtrbibpages</code>	331
<code>\glxtr@resource</code>	<code>\glxtrbibpublisher</code>	331
<code>\glxtr@s@newignoredglossary</code>	<code>\glxtrbibschooll</code>	331
<code>\glxtr@s@provideignoredglossary</code> ...	<code>\glxtrbibseries</code>	331
<code>\glxtr@saveentrycounter</code> 8, 9, 12, 83	<code>\glxtrbibtitle</code>	331
<code>\glxtr@setaccessdisplay</code>	<code>\glxtrbibtype</code>	332
<code>\glxtr@setbookindexmark</code>	<code>\glxtrbibvolume</code>	332
<code>\glxtr@setup@record</code> 13, 14, 26, 27	<code>\glxtrbookindexatendgroup</code>	404
<code>\glxtr@shortcutsval</code>	<code>\glxtrbookindexatsubendgroup</code>	404
<code>\glxtr@texencoding</code>	<code>\glxtrbookindexatsubsubendgroup</code> ..	404
<code>\glxtr@undefaction@nr</code>	<code>\glxtrbookindexbetween</code>	404
<code>\glxtr@undefaction@val</code>	<code>\glxtrbookindexbookmark</code>	405
<code>\glxtr@usesee</code>	<code>\glxtrbookindexcols</code>	403
<code>\glxtr@warnonexistsordo</code> 7, 13, 14, 44	<code>\glxtrbookindexcolspread</code>	403
<code>\glxtr@writefields</code>	<code>\glxtrbookindexfirstmarkfmt</code>	406
<code>\glxtrabbreviationfont</code>	<code>\glxtrbookindexformatheader</code>	405
<code>\glxtrabbrvfootnote</code>	<code>\glxtrbookindexgroupskip</code>	405
.... 221–223, 244–246, 258–260, 279–281	<code>\glxtrbookindexlastmarkfmt</code>	406
<code>\glxtrabbrvpluralsuffix</code>	<code>\glxtrbookindexmulticolseenv</code>	403
135, 199, 217, 219, 222, 224, 225, 227, 230,	<code>\glxtrbookindexname</code>	401, 404
233, 248, 261, 284, 293, 296, 299, 304, 307	<code>\glxtrbookindexparentchildsep</code>	401, 403
<code>\glxtrabbrvtype</code>	<code>\glxtrbookindexparentschildsep</code> .	
<code>\glxtractivatenopost</code> 403, 404	
<code>\glxtraddallcrossrefs</code>	<code>\glxtrbookindexprelocation</code> ...	401, 404
<code>\glxtralias</code>	<code>\glxtrbookindexsubbetween</code>	404
<code>\glxtrAltTreeIndent</code>	<code>\glxtrbookindexsubname</code>	405
<code>\glxtralttreeInit</code>	<code>\glxtrbookindexsubprelocation</code>	405
<code>\glxtrAltTreePar</code>	<code>\glxtrbookindexsubsubbetween</code>	404

<code>\glxtrbookindexthepage</code>	406	<code>\glxtrfull</code>	18, 19, 320, 321
<code>\glxtrcat</code>	54, 55	<code>\Glsxtrfullformat</code>	
<code>\glxtrchecknohyperfirst</code>	69–71	. 198, 212, 214, 215, 217, 220, 222, 224,	
<code>\glxtrcombingdiacriticIIrules</code> .	335	226, 228, 231, 234, 236, 238, 239, 242,	
<code>\glxtrcombingdiacriticIIrules</code> ..	335	243, 245, 246, 248, 250, 252, 254, 256,	
<code>\glxtrcombingdiacriticIrules</code> ...	335	257, 259, 261, 262, 264, 266, 268, 270,	
<code>\glxtrcombingdiacriticIVrules</code> ..	335	271, 273, 275, 277, 279, 280, 282, 285,	
<code>\glxtrComputeTreeIndent</code>	393, 394	286, 288, 291, 294, 297, 300, 302, 305, 307	
<code>\glxtrComputeTreeSubIndent</code>	394	<code>\glxtrfullformat</code>	
<code>\glxtrcounterprefix</code>	127 198, 212, 214, 215, 217, 219,	
<code>\glxtrcurrencyrules</code>	338	222, 224, 226, 228, 231, 234, 236, 238,	
<code>\Glsxtrdefaultsubsequentfmt</code> ...	213, 214	239, 242–244, 246, 248, 250, 252, 253,	
<code>\glxtrdefaultsubsequentfmt</code> ...	212, 214	256–258, 260, 262, 264, 266, 268, 270,	
<code>\Glsxtrdefaultsubsequentplfmt</code> .	213, 214	271, 273, 275, 277, 279, 280, 282, 285,	
<code>\glxtrdefaultsubsequentplfmt</code> .	213, 214	286, 288, 291, 294, 297, 300, 302, 305, 307	
<code>\GlsXtrDefineAbbreviationShortcuts</code> .	20	<code>\GLSxtrfullpl</code>	18, 19, 321, 322
<code>\GlsXtrDefineAcShortcuts</code>	20, 21	<code>\Glsxtrfullpl</code>	18, 19, 322
<code>\GlsXtrDefineOtherShortcuts</code>	20	<code>\glxtrfullpl</code>	18, 19, 321
<code>\glxtrdetoklocation</code>	146	<code>\Glsxtrfullplformat</code>	
<code>\glxtrdiscardperiod</code>	188	. 198, 212, 214, 215, 217, 220, 222, 224,	
<code>\glxtrdisplayendloc</code>	126	226, 228, 231, 234, 236, 238, 239, 242,	
<code>\glxtrdisplayendloohook</code>	127	243, 245, 246, 249, 250, 252, 254, 256,	
<code>\glxtrdisplaysingleloc</code>	126, 127	258, 259, 261, 263, 264, 266, 268, 270,	
<code>\glxtrdisplaystartloc</code>	126	271, 274, 275, 277, 279, 280, 282, 285,	
<code>\glxtrdoautoindexname</code>	84, 85, 178	286, 288, 291, 294, 297, 300, 302, 305, 308	
<code>\glxtrdopostpunc</code>	223, 246, 260, 281	<code>\glxtrfullplformat</code>	
<code>\glxtrdowrglossaryhook</code>	84 211, 212, 214, 215, 217, 219, 222,	
<code>\GlsXtrDualField</code>	331	224, 226, 228, 231, 234, 236, 238, 239,	
<code>\glxtrremsuffix</code>	262, 264, 266,	242–244, 246, 248, 250, 252, 253, 256,	
267, 269, 270, 272, 274, 276, 278, 280, 282		257, 259, 260, 262, 264, 266, 268, 270,	
<code>\GlsXtrEnableEntryCounting</code>	111	271, 273, 275, 277, 279, 280, 282, 285,	
<code>\GlsXtrEnableEntryUnitCounting</code>	100	286, 288, 291, 294, 297, 300, 302, 305, 307	
<code>\glxtrendfor</code>	32	<code>\glxtrfullsep</code>	197, 198,
<code>\glxtrfieldlistgadd</code>	136	217–220, 222–229, 231, 233–241, 243,	
<code>\glxtrfieldtitlecase</code>	173–176, 179	245, 247–257, 259, 261–269, 271–273,	
<code>\glxtrfieldtitlecasecs</code>	172	275–278, 281–283, 293–298, 301–304, 308	
<code>\glxtrfieldxifinlist</code>	141	<code>\glxtrngenabbrvfmt</code>	60
<code>\glxtrfirstscfont</code>	233	<code>\glxtrgeneralpuncIIrules</code>	338
<code>\glxtrfirstsmfont</code>	247	<code>\glxtrgeneralpuncIrules</code>	338
<code>\GlsXtrFmtDefaultOptions</code>	30	<code>\glxtrgetgroupitle</code>	125, 405
<code>\glxtrfmtdisplay</code>	30	<code>\glxtrgroupfield</code>	142
<code>\GlsXtrFmtField</code>	30	<code>\glxtrheadfirst</code>	312
<code>\glxtrfootnotename</code>		<code>\glxtrheadfirst</code>	312
.... 221, 223, 244, 245, 258, 260, 279, 281		<code>\Glsxtrheadfirstplural</code>	312
<code>\GlsXtrForeignTextField</code>	36	<code>\glxtrheadfirstplural</code>	312
<code>\GlsXtrFormatLocationList</code> 58, 60, 390–392		<code>\Glsxtrheadfull</code>	312
<code>\GLSxtrfull</code>	18, 19, 320, 321	<code>\glxtrheadfull</code>	312
<code>\Glsxtrfull</code>	18, 19, 321	<code>\Glsxtrheadfullpl</code>	312
		<code>\glxtrheadfullpl</code>	312

<code>\Glsxtrheadlong</code>	312	243, 245, 247, 251, 253–255, 257, 259,
<code>\glsxtrheadlong</code>	312	261, 269, 270, 272, 273, 275, 276, 278,
<code>\Glsxtrheadlongpl</code>	312	280, 282, 286, 289, 296, 300, 305, 308, 327
<code>\glsxtrheadlongpl</code>	312	<code>\Glsxtrinilinefullplformat</code> 198, 202, 214,
<code>\Glsxtrheadname</code>	312	215, 223, 224, 226, 228, 229, 231, 238–
<code>\glsxtrheadname</code>	136, 137, 312	241, 243, 245, 247, 252–254, 256, 257,
<code>\Glsxtrheadplural</code>	312	259, 261, 269, 271–273, 275, 277, 278,
<code>\glsxtrheadplural</code>	312	281, 283, 286, 289, 297, 300, 305, 308, 328
<code>\Glsxtrheadshort</code>	312	<code>\glsxtrinilinefullplformat</code>
<code>\glsxtrheadshort</code>	312 198, 201, 202, 214,
<code>\Glsxtrheadshorttpl</code>	312	215, 222, 224, 225, 227, 229, 231, 237,
<code>\glsxtrheadshorttpl</code>	312	239–241, 243, 245, 247, 252–255, 257,
<code>\Glsxtrheadtext</code>	312	259, 261, 269, 271–273, 275, 276, 278,
<code>\glsxtrheadtext</code>	312	281, 282, 286, 289, 296, 300, 305, 308, 328
<code>\glsxtrhyperlink</code>	23, 24, 87	<code>\glsxtrininsertinsidefalse</code>
<code>\glsxtrhyphensuffix</code>	294, 302	216
<code>\glsxtrifcounttrigger</code>	103, 104	<code>\GlsXtrInternalLocationHyperlink</code> 23, 127
<code>\glsxtrifcustomdiscardperiod</code>	188	<code>\glsxtrLatinA</code>
<code>\glsxtrifemptyglossary</code>	125, 132, 139	340–345
<code>\GlsXtrIfFieldCmpNum</code>	32, 33	<code>\glsxtrLatinAELigature</code>
<code>\GlsXtrIfFieldEqNum</code>	137	342, 344, 345
<code>\GlsXtrIfFieldNonZero</code>	330	<code>\glsxtrLatinE</code>
<code>\glsxtrifhasfield</code> ..	35, 36, 83, 330, 331, 401	340–345
<code>\glsxtrifhyphenstart</code>	293, 295, 298, 301, 303, 304	<code>\glsxtrLatinEszettSs</code>
<code>\glsxtrifindexing</code>	84	341–343, 345
<code>\glsxtrifinmark</code>	67, 93–96, 311, 312	<code>\glsxtrLatinEszettSz</code>
<code>\glsxtrifnextpunc</code>	191, 192	342, 344
<code>\glsxtrifperiod</code>	188, 190, 191	<code>\glsxtrLatinEth</code>
<code>\glsxtrifrecordtrigger</code>	148–150	341–344
<code>\glsxtrifwasfirstuse</code>	68–71, 75–79, 82, 114, 189,	<code>\glsxtrLatinH</code>
	190, 200, 203–210, 223, 224, 246, 260,	340–345
	281, 282, 285, 287–289, 299, 301, 304, 306	<code>\glsxtrLatinI</code>
<code>\glsxtrinclinkcounter</code>	152	340–345
<code>\glsxtrindexaliased</code>	83	<code>\glsxtrLatinInsularG</code>
<code>\glsxtrindexseealso</code>	48, 49	340–345
<code>\glsxtrinithyperoutside</code>	65	<code>\glsxtrLatinK</code>
<code>\glsxtrinitwrgloss</code>	65, 147	340–345
<code>\glsxtrinitwrglossbeforefalse</code>	63	<code>\glsxtrLatinL</code>
<code>\glsxtrinitwrglossbeforetrue</code>	63	340–345
<code>\Glsxtrinilinefullformat</code> 198, 200, 214,		<code>\glsxtrLatinM</code>
215, 223, 224, 226, 228, 229, 231, 238–		340–345
241, 243, 245, 247, 252–254, 256, 257,		<code>\glsxtrLatinN</code>
259, 261, 269, 271–273, 275, 277, 278,		340–345
281, 282, 286, 289, 297, 300, 305, 308, 328		<code>\glsxtrLatinO</code>
<code>\glsxtrinilinefullformat</code>		340–345
..... 198, 200, 201, 214, 215, 222,		<code>\glsxtrLatinOELigature</code>
224, 225, 227, 229, 231, 237, 239–241,		342, 344, 345
		<code>\glsxtrLatinP</code>
		340–345
		<code>\glsxtrLatinS</code>
		340–345
		<code>\glsxtrLatinT</code>
		340–345
		<code>\glsxtrLatinThorn</code>
		344
		<code>\glsxtrLatinX</code>
		340–345
		<code>\glsxtrlocationhyperlink</code>
		127
		<code>\glsxtrlocrangefmt</code>
		126, 127
		<code>\GLSxtrlong</code>
		18, 19, 319
		<code>\Glsxtrlong</code>
		18, 19, 320
		<code>\glsxtrlong</code>
		18, 19, 319
		<code>\glsxtrlonghyphen</code>
		300
		<code>\glsxtrlonghyphennoshort</code>
		296, 297
		<code>\glsxtrlonghyphenshort</code>
		294
		<code>\glsxtrlongnoshortdescname</code> .
		230, 277, 296
		<code>\glsxtrlongnoshortname</code>
	 232, 240, 255, 272, 274, 297
		<code>\GLSxtrlongpl</code>
		18, 19, 319, 320
		<code>\Glsxtrlongpl</code>
		18, 19, 320

<code>\glxstrlongpl</code>	18, 19, 319	<code>\glxstronlydescsort</code>	308
<code>\glxstrlongshortdescname</code>		<code>\glxstronlyname</code>	307
.....	218, 235, 249, 263, 265, 295, 300	<code>\glxstronlysuffix</code>	307
<code>\glxstrlongshortdescsort</code>		<code>\glxstrorg@ifKV@glslink@hyper</code>	61
.....	218, 235, 249, 263, 265, 290, 295, 300	<code>\glxstrorglong</code>	195, 218, 296
<code>\glxstrlongshortname</code>		<code>\glxstrorgshort</code>	195, 218
.....	217, 233, 248, 262, 263, 284, 285, 294, 299	<code>\GLSxtrp</code>	93
<code>\glxstrlongshortuserdescname</code> ..	287, 290	<code>\Glsxtrp</code>	93
<code>\glxstrmarkhook</code>	310	<code>\glxstrp</code>	92, 94
<code>\glxstrMathItalicAlpha</code> .	349, 350, 353, 354	<code>\glxstrparen</code>	189,
<code>\glxstrMathItalicBeta</code> ..	349, 350, 353, 354	190, 197, 198, 217–220, 222–229, 231,
<code>\glxstrMathItalicChi</code>	350, 354, 355	233–243, 245, 247–257, 259, 261–273,
<code>\glxstrMathItalicDelta</code> .	349, 350, 353, 354	275–278, 281–283, 293–298, 301–304, 308
<code>\glxstrMathItalicEpsilon</code> ..	349, 350, 353, 354	<code>\Glsxtrpl</code>	55
<code>\glxstrMathItalicEta</code> ...	349, 350, 353, 354	<code>\glxstrpl</code>	55
<code>\glxstrMathItalicGamma</code> .	349, 350, 353, 354	<code>\glxstrpostdescription</code> .	120, 170, 187, 377
<code>\glxstrMathItalicIota</code> ..	349, 350, 353, 354	<code>\glxstrposthyphenlong</code>	304, 306
<code>\glxstrMathItalicKappa</code> .	349, 350, 353, 354	<code>\glxstrposthyphenshort</code>	299, 301
<code>\glxstrMathItalicLambda</code> ..	349, 350, 353, 354	<code>\glxstrposthyphensubsequent</code>	
<code>\glxstrMathItalicMu</code>	349, 350, 353, 354	299, 301, 304, 306
<code>\glxstrMathItalicNu</code>	349, 350, 353, 355	<code>\glxstrpostlink</code>	188
<code>\glxstrMathItalicOmega</code>	350, 354, 355	<code>\glxstrpostlinkendsentence</code>	188
<code>\glxstrMathItalicOmicron</code> ..	349, 350, 354, 355	<code>\glxstrpostlinkhook</code>	188
<code>\glxstrMathItalicPhi</code>	350, 354, 355	<code>\glxstrpostlocalreset</code>	99, 101, 109
<code>\glxstrMathItalicPi</code>	349, 350, 354, 355	<code>\glxstrpostlocalunset</code>	98, 101, 109
<code>\glxstrMathItalicPsi</code>	350, 354, 355	<code>\glxstrpostlongdescription</code>	39
<code>\glxstrMathItalicRho</code> ...	349, 350, 354, 355	<code>\glxstrpostnamehook</code>	175–177, 180
<code>\glxstrMathItalicSigma</code>	350, 354, 355	<code>\GlsXtrPostNewAbbreviation</code>	197,
<code>\glxstrMathItalicTau</code>	350, 354, 355	214, 215, 217–221, 223, 225–230, 232–
<code>\glxstrMathItalicTheta</code> .	349, 350, 353, 354	235, 237, 238, 241, 244, 246, 248–251,
<code>\glxstrMathItalicUpsilon</code> ...	350, 354, 355	253, 255, 258, 260, 262–265, 267–270,
<code>\glxstrMathItalicXi</code>	349, 350, 354, 355	272, 274, 276, 278–281, 284, 285, 287–
<code>\glxstrMathItalicZeta</code> ..	349, 350, 353, 354	292, 294–297, 299, 301–304, 306, 307, 309
<code>\glxstrnewabbrevpresetkeyhook</code>	196	<code>\glxstrpostreset</code>	99, 101, 109
<code>\glxstrnewnumber</code>	19	<code>\glxstrpostunset</code>	97, 101, 109
<code>\glxstrnewsymbol</code>	19	<code>\glxstrprelocation</code>	
<code>\glxstrNoGlossaryWarning</code>	14, 21, 128	367, 369, 371, 373, 375, 378, 401
<code>\GlsXtrNoGlsWarningAutoMake</code>	132	<code>\glxstrprotectlinks</code>	86–88
<code>\GlsXtrNoGlsWarningBuildInfo</code>	132	<code>\GlsXtrRecordCounter</code>	11
<code>\GlsXtrNoGlsWarningCheckFile</code>	132	<code>\glxstrrecordtriggervalue</code>	147
<code>\GlsXtrNoGlsWarningEmptyMain</code>	132	<code>\GlsXtrRecordWarning</code>	133
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	132	<code>\glxstrregularfont</code>	60, 68
<code>\GlsXtrNoGlsWarningEmptyStart</code>	132	<code>\glxstrresourcecount</code>	134
<code>\GlsXtrNoGlsWarningHead</code>	131	<code>\glxstrresourcefile</code>	134
<code>\GlsXtrNoGlsWarningMisMatch</code>	132	<code>\glxstrresourceinit</code>	133
<code>\GlsXtrNoGlsWarningNoOut</code>	132	<code>\glxstrrestoremarkhook</code>	310
<code>\GlsXtrNoGlsWarningTail</code>	132, 133	<code>\glxstrrestorepostpunc</code>	120
<code>\glxstrnopostpunc</code>	119	<code>\glxstrscfont</code>	233
<code>\glxstronlydescname</code>	308		

<code>\glstrscsuffix</code>	<code>\Glsxtrtitlefirstplural</code>
.... 234, 236, 237, 239, 241, 242, 244, 246	311–313, 326
<code>\GlsXtrSetActualChar</code>	<code>\glstrtitlefirstplural</code>
184	311, 312, 326
<code>\glstrsetaliasnoindex</code>	<code>\Glsxtrtitlefull</code>
13, 14, 83	311–313, 328
<code>\GlsXtrSetEncapChar</code>	<code>\glstrtitlefull</code>
184	311–313, 327
<code>\GlsXtrSetEscChar</code>	<code>\Glsxtrtitlefullpl</code>
184	311–313, 328
<code>\glstrsetfieldifexists</code>	<code>\glstrtitlefullpl</code>
34, 35	311–313, 328
<code>\GlsXtrSetLevelChar</code>	<code>\Glsxtrtitlelong</code>
184	311–313, 326, 327
<code>\glstrsetpopts</code>	<code>\glstrtitlelong</code>
92	311–313, 326
<code>\glstrsetupfulldefs</code>	<code>\Glsxtrtitlelongpl</code>
..... 200–202, 224, 246, 260, 282	311–313, 327
<code>\GLSxtrshort</code>	<code>\glstrtitlelongpl</code>
18, 19, 96, 313, 314	311–313, 327
<code>\Glsxtrshort</code>	<code>\Glsxtrtitlename</code>
18, 19, 314	311, 312, 324
<code>\glstrshort</code>	<code>\glstrtitlename</code>
18, 313	311, 312, 323
<code>\glstrshortdescname</code> ...	<code>\glstrtitleorpdforheading</code>
227, 238, 253, 270 25, 136–138, 311, 312
<code>\glstrshorthyphen</code>	<code>\Glsxtrtitleplural</code>
305	311, 312, 325
<code>\glstrshorthyphenlong</code>	<code>\glstrtitleplural</code>
302	311, 312, 324, 325
<code>\glstrshortlongdescname</code>	<code>\Glsxtrtitleshort</code>
..... 220, 236, 250, 266, 268, 303, 306	311, 312, 323
<code>\glstrshortlongdescsort</code>	<code>\glstrtitleshort</code>
..... 220, 236, 251, 266, 268, 292, 303, 306	311, 312, 322
<code>\glstrshortlongname</code>	<code>\Glsxtrtitleshortpl</code>
219, 235, 249, 265, 267, 287, 291, 302, 304	311, 312, 323
<code>\glstrshortlonguserdescname</code> ..	<code>\glstrtitleshortpl</code>
289, 292	311, 312, 322, 323
<code>\glstrshortnolongname</code> .	<code>\Glsxtrtitletext</code>
225, 237, 251, 269	311, 312, 324
<code>\GLSxtrshortpl</code>	<code>\glstrtitletext</code>
18, 19, 313, 314	311, 312, 324
<code>\Glsxtrshortpl</code>	<code>\GlsXtrTotalRecordCount</code>
18, 19, 314	146
<code>\glstrshortpl</code>	<code>\glstrtreetopindent</code>
18, 19, 313, 314	384, 392
<code>\glstrsmfont</code>	<code>\glstrundefaction</code> 7, 13, 14, 28, 40, 41, 43, 44
247	<code>\glstrundeftag</code>
<code>\glstrsmsuffix</code>	27, 122, 123
..... 248, 250, 251, 253, 255, 256, 258, 260	<code>\GlsXtrUnknownDialectWarning</code>
<code>\GlsXtrStandaloneGlossaryType</code> .	37
137, 138	<code>\glstrunsrtdo</code>
<code>\GlsXtrStandaloneSubEntryItem</code> .	140, 141
137, 138	<code>\glstrUpAlpha</code>
<code>\Glsxtrsubsequentfmt</code>	348, 349, 353, 354
..... 211, 214, 230, 241, 243,	<code>\glstrUpBeta</code>
255, 257, 273, 274, 276, 278, 296, 299, 305	348, 349, 353, 354
<code>\glstrsubsequentfmt</code>	<code>\glstrUpChi</code>
..... 211, 214, 230, 241, 242,	348, 349, 354, 355
255, 257, 273, 274, 276, 278, 296, 299, 305	<code>\glstrUpDelta</code>
<code>\Glsxtrsubsequentplfmt</code>	348, 349, 353, 354
..... 211, 214, 231, 241, 243,	<code>\glstrUpDigamma</code>
255, 257, 273, 274, 276, 278, 296, 300, 305	348, 349, 353
<code>\glstrsubsequentplfmt</code>	<code>\glstrUpEpsilon</code>
..... 211, 214, 230, 241, 242,	348, 349, 353, 354
255, 257, 273, 274, 276, 278, 296, 299, 305	<code>\glstrUpEta</code>
<code>\glstrsupplocationurl</code>	348, 349, 353, 354
127, 128	<code>\glstrUpGamma</code>
<code>\glstrtagfont</code>	348, 349, 353, 354
187	<code>\glstrUpIota</code>
<code>\Glsxtrtitlefirst</code>	348, 349, 353, 354
311, 312, 325	<code>\glstrUpKappa</code>
<code>\glstrtitlefirst</code>	348, 349, 353, 354
311, 312, 325	<code>\glstrUpLambda</code>
	348, 349, 353, 354
	<code>\glstrUpMu</code>
	348, 349, 353, 355
	<code>\glstrUpNu</code>
	348, 349, 353, 355
	<code>\glstrUpOmega</code>
	348, 349, 354, 355
	<code>\glstrUpOmicron</code>
	348, 349, 354, 355
	<code>\glstrUpPhi</code>
	348, 349, 354, 355
	<code>\glstrUpPi</code>
	348, 349, 354, 355
	<code>\glstrUpPsi</code>
	348, 349, 354, 355
	<code>\glstrUpRho</code>
	348, 349, 354, 355
	<code>\glstrUpSigma</code>
	348, 349, 354, 355
	<code>\glstrUpTau</code>
	348, 349, 354, 355

<code>\glxtrUpTheta</code>	348, 349, 353, 354	<code>\ifcsundef</code>	33, 36–38, 40–42, 52, 56, 58, 88, 101, 106–110, 124, 125, 128, 141, 152, 168, 213, 215, 216, 366, 384, 385, 393, 406
<code>\glxtrUpUpsilon</code>	348, 349, 354, 355	<code>\ifcsvoid</code>	49, 167
<code>\glxtrUpXi</code>	348, 349, 354, 355	<code>\ifdef</code>	14, 19, 26, 30, 35, 37, 43, 44, 46, 47, 51, 56, 57, 82, 86, 87, 93–95, 118, 122, 123, 135, 144, 170, 171, 184, 187, 283, 311, 322–328, 330, 363, 364, 366–369, 377–383, 394–400, 402, 405, 406
<code>\glxtrUpZeta</code>	348, 349, 353, 354	<code>\ifdefempty</code>	7–9, 32, 33, 36, 37, 40–42, 45, 46, 65, 67, 100, 112, 115, 118, 126, 139, 142, 146, 147, 161–163, 186, 194, 210, 403
<code>\GlsXtrUseAbbrStyleFmts</code>	218, 221, 227, 229, 230, 232, 233, 235, 237, 240, 249, 251, 254, 263, 265, 267, 269, 271, 276, 279, 287, 290, 292, 295, 296, 298, 301, 303, 306, 309	<code>\ifdefequal</code>	35, 51, 132, 142, 178
<code>\GlsXtrUseAbbrStyleSetup</code>	226, 228, 229, 232, 233, 240, 242, 254, 256, 271, 275, 276, 279	<code>\ifdefstring</code>	6, 23, 35, 181, 186, 402
<code>\glxtruserfield</code>	283	<code>\ifdefvoid</code>	45, 48–50, 87, 106, 123, 124, 127, 142, 143
<code>\glxtruserparen</code>	284–292	<code>\ifdim</code>	57, 114, 384–392
<code>\glxtrusersuffix</code>	284, 286, 288, 291	<code>\IfFileExists</code>	22, 128, 132, 133, 135, 364, 365
<code>\glxtruseseealsoformat</code>	46	<code>\ifglossaryexists</code>	44
<code>\glxtruseseeformat</code>	45	<code>\ifglshasacronym</code>	17, 132
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	193, 215	<code>\ifglshasshortcuts</code>	20
<code>\GlsXtrWarning</code>	54, 55	<code>\ifglshasautomake</code>	118, 132, 135
<code>\glxtrword</code>	194	<code>\ifglshasentrycounter</code>	37, 38
<code>\glxtrwordsep</code>	194, 293, 295, 298, 301, 303, 304	<code>\ifglshasentryexists</code>	9, 43, 44, 54, 55, 58, 68, 142, 168, 187, 188
<code>\glxtrwrglossmark</code>	24	<code>\ifglshasfield</code>	166
H			
<code>\hangindent</code>	380, 382, 383, 393–395, 399, 400	<code>\ifglshasfield</code>	30, 283
<code>\hbox</code>	366	<code>\ifglshaslong</code>	105, 106, 150, 151
<code>\hfill</code>	366	<code>\ifglshasparent</code>	137–139, 142, 386–388
<code>\href</code>	86	<code>\ifglshasshort</code>	46, 60, 68
<code>\hsize</code>	57	<code>\ifglshassymbol</code>	190, 380, 381, 383
<code>\hss</code>	366	<code>\ifglshasindexonlyfirst</code>	84
<code>\hyperlink</code>	87	<code>\ifglshasgroupskip</code>	367, 369–377, 379, 380, 382, 394, 402
<code>\hyperpage</code>	180	<code>\ifglshasnonnumberlist</code>	60
<code>\hyperref</code>	86, 128, 330	<code>\ifglshasnopostdot</code>	16, 120
<code>hyperref package</code>	88, 180, 309, 322	<code>\ifglssanitizesort</code>	117
I			
<code>\if</code>	53	<code>\ifglssubentrycounter</code>	38
<code>\if@glxtr@autoseeindex</code>	25, 26, 45, 48	<code>\ifglssused</code>	50, 82, 84, 102, 111, 114, 210, 386, 387, 389, 390, 392
<code>\if@glxtr@format@override</code>	181	<code>\ifglsxindy</code>	128, 130
<code>\if@glxtr@docdefrestricted</code>	51	<code>\ifglxtr@hyperoutside</code>	66
<code>\if@glxtr@indexcrossrefs</code>	15, 49	<code>\ifglxtr@trinitwrglossbefore</code>	63, 66, 147
<code>\ifblank</code>	28, 54, 55, 115	<code>\ifglxtr@insertinside</code>	203–210, 212, 213, 217, 219, 220, 222–231, 234, 236–250, 252– 264, 266, 268–283, 285, 286, 288, 289, 291, 293, 295, 298–301, 304, 305, 307, 308
<code>\ifbool</code>	27	<code>\ifHy@hyperindex</code>	180
<code>\ifcase</code>	7, 13, 20, 21, 24, 52, 63, 121, 378, 404		
<code>\ifcsdef</code>	28, 38, 40–43, 64, 65, 80, 81, 92–96, 106, 118, 124, 125, 137, 141, 144–146, 151, 172– 175, 177–179, 189, 193, 210, 214, 369–376		
<code>\ifcsstring</code>	28, 168, 213		

N

<code>\NeedsTeXFormat</code>	5, 330, 365, 401	<code>short</code>	158, 165, 193
<code>\new@glossaryentry</code>	52, 118	<code>shortaccess</code>	161–163
<code>\new@ifnextchar</code>	29, 80,	<code>shortaccessplural</code>	161
81, 104, 105, 144, 148–150, 191, 199–209		<code>shortplural</code>	159, 166, 193
<code>\newabbr</code>	18, 19	<code>symbol</code>	156, 164
<code>\newabbreviation</code>	18, 19	<code>symbolplural</code>	156, 157, 164, 165
<code>\newabbreviationhook</code>	196	<code>text</code> 86, 153, 154, 163, 216, 218, 315, 316, 324	
<code>\newabbreviationstyle</code>		<code>textaccess</code>	162
. 216, 218–221, 223, 225–230, 232–238,		<code>user2</code>	37
240, 242, 244, 245, 248–252, 254–256,		<code>\newglossarystyle</code>	403
258, 259, 262, 263, 265–272, 274–277,		<code>\newif</code>	63, 180, 216
279, 281, 284, 285, 287, 289, 290, 292,		<code>\newlength</code>	383, 384
294, 296, 297, 299, 300, 302, 304, 306–308		<code>\newnum</code>	19
<code>\newacronym</code>	112, 113	<code>\newrobustcmd</code>	
<code>\newacronymhook</code>	112 29, 30, 32, 34, 35, 46, 47, 64, 67,	
<code>\newacronymstyle</code>	113, 114	80, 81, 92, 93, 104, 105, 120, 124, 136,	
<code>\newcommand</code>	5–13,	138, 141, 144, 145, 148–150, 179, 186,	
15–35, 37–46, 48–50, 53–56, 58–60, 63,		187, 199–210, 292, 311, 313–322, 385–392	
64, 67–69, 80, 81, 83–85, 87, 88, 91–102,		<code>\newsym</code>	19
104–114, 118–124, 126–131, 133–142,		<code>\newterm</code>	170
144–161, 163–172, 178–194, 197–210,		<code>\newtoks</code>	193
212–216, 218–221, 225, 227, 230, 232,		<code>\newwrite</code>	115
233, 247, 248, 261, 262, 283, 284, 287,		<code>\nobreak</code>	368, 369, 379, 381–383, 395–398
289, 293, 295, 298, 301, 303, 304, 306–		<code>\NoCaseChange</code>	93–96, 137, 313–322
308, 310–331, 334–363, 365, 367, 377–		<code>\noexpand</code> 10, 11, 22, 46, 48, 49, 51, 112, 128,	
381, 383–385, 392, 393, 401, 402, 405, 406		129, 133, 140–142, 151, 182, 197, 365, 404	
<code>\newcount</code>	133, 143	<code>\nofiles</code>	131
<code>\newcounter</code>	23, 151	<code>\noindent</code>	132, 381–383, 396–398, 400
<code>\newentry</code>	19	<code>\nopagebreak</code>	379, 381–383, 394–401, 405
<code>\newglossary</code>	17, 115	<code>\nopostdesc</code>	40, 54, 55, 119, 170
<code>\newglossaryentry</code>		<code>\ns@GLSxtrfull</code>	200
..... 19, 52, 100, 108, 112, 170, 171, 197		<code>\ns@Glsxtrfull</code>	200
<code>\newglossaryentry options</code>		<code>\ns@glxtrfull</code>	199
access	162	<code>\ns@GLSxtrfullpl</code>	202
alias	16, 44, 47–50	<code>\ns@Glsxtrfullpl</code>	201
desc	157, 165	<code>\ns@glxtrfullpl</code>	201
descplural	157, 158, 165	<code>\ns@GLSxtrlong</code>	206
first 86, 154, 155, 164, 216, 317, 318, 325, 407		<code>\ns@Glsxtrlong</code>	205
firstaccess	163	<code>\ns@glxtrlong</code>	204, 205
firstplural	155, 164, 216, 318, 325, 407	<code>\ns@GLSxtrlongpl</code>	209
group	142	<code>\ns@Glsxtrlongpl</code>	209
loclist	31	<code>\ns@glxtrlongpl</code>	208
long	159, 166, 326	<code>\ns@GLSxtrshort</code>	204
longplural	160, 166, 327	<code>\ns@Glsxtrshort</code>	203
name	45, 153, 163, 181, 314, 315, 323	<code>\ns@glxtrshort</code>	203
plural	154, 163, 164, 216, 316, 317, 324	<code>\ns@GLSxtrshortpl</code>	208
see	15, 16, 26, 44, 45, 47, 50, 52, 116	<code>\ns@Glsxtrshortpl</code>	207
seealso	16, 44, 46, 47, 49, 50, 418	<code>\ns@glxtrshortpl</code>	206
		<code>\null</code>	21

<code>\number</code>	51, 107–110, 133, 144	<code>symbols</code>	19, 171
<code>\numexpr</code>	107, 110	<code>undefaction</code>	43
O			
<code>\O</code>	345, 348	<code>error</code>	6
<code>\o</code>	348	<code>warn</code>	6
<code>\or</code>	7, 13, 14, 20, 21, 24, 52, 63, 378, 404	<code>xindy</code>	46, 47
<code>\org@glossaryentrynumbers</code>	57, 58, 119	<code>\PackageError</code>	6, 11, 22, 26, 52, 55, 56, 64, 80, 81, 83, 92, 93, 100, 101, 108, 110, 111, 113, 115, 117, 118, 125, 135, 140, 141, 144, 189, 213–216, 365, 366
<code>\org@glossarytitle</code>	119	<code>\PackageWarning</code>	16
<code>\org@ifKV@glslink@hyper</code>	64, 66	<code>\PackageWarningNoLine</code>	16
P			
<code>\p@gl@hyp@opt</code>	85	<code>\pageref</code>	23, 37, 38
package options:		<code>\par</code>	131, 132, 368, 369, 378, 380–383, 393–400, 402
abbreviations	17, 18	<code>\parindent</code> ..	378, 380, 382–384, 393–400, 403
accsupp	21, 153	<code>\parskip</code>	378, 380, 382, 396–398, 403
acronym	17	<code>\PassOptionsToPackage</code>	5, 22
automake	118, 130, 135	<code>\pdfbookmark</code>	402
true	135	<code>\preglossarypreamble</code>	38
autoseeindex	26	<code>\preto</code>	83
false	25	<code>\print@noop@unsrtglossaryunit</code> ...	12, 13
counter		<code>\print@op@unsrtglossaryunit</code>	14
wrglossary	23	<code>\printabbreviations</code>	17
debug		<code>\printglossaries</code>	116, 130
showtargets	87	<code>\printglossary</code>	17, 116, 130, 131, 133
docdef	14, 15, 51, 52, 100, 108	<code>\printglossary options</code>	
false	51, 52	nonumberlist	60
restricted	15	type	118
true	51, 52	<code>\printnoidxglossaries</code>	130
docdefs		<code>\printnoidxglossary</code>	116, 117, 130
restricted	51	<code>\printnumbers</code>	19, 171
indexcounter	330	<code>\printsymbols</code>	19, 171
nonumberlist	57	<code>\printunsrtglossary</code>	131, 133, 139
nopostdot	16	<code>\printunsrtglossaryentryprocesshook</code>	
false	16	139, 140
numbers	19	<code>\printunsrtglossaryhandler</code>	140, 141
postdot	16	<code>\printunsrtglossarypredoglossary</code> ..	140
record 7, 13, 51, 61, 115, 133, 199, 201–210, 416		<code>\printunsrtglossaryskipentry</code>	139
alsoindex	8, 10	<code>\printunsrtglossaryunit</code>	13, 14, 141
only	8, 131	<code>\printunsrtglossaryunitsetup</code>	141
shortcuts	20	<code>\ProcessOptions</code>	366
ac	20	<code>\ProcessOptionsX</code>	24
all	20	<code>\protect</code>	93–96, 163–166, 194, 197, 198, 216–223, 225–227, 229–246, 248– 260, 262–270, 272–282, 284, 285, 287– 292, 294–297, 299–304, 306–309, 313–322
false	20	<code>\protected@csedef</code>	34, 35, 124, 384, 385
none	20	<code>\protected@csxdef</code>	34, 124, 384, 385
true	20		
sort			
use	67		
style	22		
stylemods	22		

<code>\protected@edef</code>	35,	<code>\rGlsplformat</code>	149
	56, 64, 86, 112, 123, 124, 141, 142, 181, 197	<code>\rglsplformat</code>	148, 151
<code>\protected@write</code>		<code>\romannumeral</code>	384, 385, 393
 11, 12, 51, 59, 115, 116, 133–136, 406		
<code>\providecommand</code>			S
	... 17–19, 29, 46, 59, 81, 83, 102, 110,	<code>\s@glstrfmt</code>	29
	115, 128, 129, 134, 330, 332–334, 366, 401	<code>\s@glshyp@opt</code>	85
<code>\ProvidesFile</code>	329	<code>\s@glstr@enabletagging</code>	185
<code>\ProvidesPackage</code>	5, 330, 365, 401	<code>\s@glstrfmt</code>	29
		<code>\s@glstrifhasfield</code>	32
	Q	<code>\s@GlsXtrStartUnsetBuffering</code>	97
<code>\quotechar</code>	184	<code>\s@GlsXtrStopUnsetBuffering</code>	98
		<code>\s@printunsrtglossary</code>	138, 141
	R	<code>\seealso</code>	46, 47
<code>\raggedright</code>	371, 372, 375, 376, 403	<code>\seename</code>	45
<code>\refstepcounter</code>	23	<code>\setabbreviationstyle</code>	113, 218, 226
<code>\relax</code>	7, 13–15,	<code>\setacronymstyle</code>	113, 114
	18–21, 24, 26, 30, 33, 51, 52, 56, 57, 59,	<code>\setentrycounter</code>	126
	63, 65, 85, 92, 101, 102, 110, 115, 116,	<code>\SetGenericNewAcronym</code>	114
	119, 123, 124, 126, 133–135, 142, 147,	<code>\setglossarystyle</code>	22, 119,
	182, 184, 186, 189, 193–195, 292, 293,		366, 368, 369, 379, 381, 382, 394–400, 403
	378, 380, 382, 385–395, 399, 400, 403–406	<code>\setkeys</code>	9,
<code>relsize package</code>	247		22, 26, 30, 65, 67, 84, 112, 119, 147, 195, 196
<code>\renewcommand</code> 6, 7, 13–17, 20–24, 26, 27, 39–		<code>\setlength</code>	
	45, 51–53, 55, 56, 58–62, 80–82, 84, 86–		57, 378, 380, 382, 383, 394, 396–398, 403
	88, 92, 98–102, 105, 106, 108–121, 123,	<code>\settowidth</code>	114, 384–393
	125–129, 141, 146, 170, 172–177, 185–	<code>\setupglossaries</code>	5, 26
	188, 198, 214, 215, 217–292, 294–297,	<code>\sfcode</code>	16, 17, 189, 377
	299–310, 363, 366–383, 393–400, 403–405	<code>\small</code>	25
<code>\renewenvironment</code>	367, 369–	<code>soul package</code>	98
	376, 378, 380, 382, 393, 396–398, 400, 403	<code>\space</code>	6, 11,
<code>\renewglossarystyle</code>			46, 47, 52, 53, 56, 83, 100–102, 108, 110,
 366–376, 378–382, 393–400		111, 113–117, 119, 129, 132, 133, 136,
<code>\renewrobustcmd</code>	67, 87		140, 141, 144, 189, 190, 194, 198, 218,
<code>\RequireGlossariesExtraLang</code> ...	329, 364		366, 367, 377, 380, 381, 383, 384, 393, 401
<code>\RequirePackage</code> ...	5, 14, 21, 22, 24, 365, 401	<code>\spacefactor</code>	16, 17, 189, 195, 377
<code>\reserved@a</code>	191	<code>\stepcounter</code>	152
<code>\reserved@b</code>	191	<code>\string</code> 6, 11, 12, 46, 47, 51–53, 56, 59, 66, 80,	
<code>\reserved@d</code>	192		81, 83, 92, 93, 100–102, 108, 110, 111,
<code>\RestoreAcronyms</code>	113, 114		113–119, 128–136, 140, 141, 144, 174,
<code>\rGLS</code>	146		176, 177, 179, 181, 189, 330, 334–363, 406
<code>\rGls</code>	146	<code>\strut</code>	366–376, 382
<code>\rgls</code>	146	<code>\subglossentry</code>	
<code>\rGLSformat</code>	149		119, 143, 366–376, 378, 380, 382, 394, 404
<code>\rGlsformat</code>	149	<code>\subitem</code>	378
<code>\rglsformat</code>	148, 151	<code>\subsubitem</code>	378
<code>\rGLSpl</code>	146		
<code>\rGlspl</code>	146		T
<code>\rglspl</code>	146	<code>\tablehead</code>	373–376
<code>\rGLSplformat</code>	150	<code>\tabletail</code>	373–376

<code>\tabularnewline</code>	369–377		
<code>\TeX</code>	130		
<code>\texorpdfstring</code>	30, 93–96, 311, 322–328		
<code>\textbf</code>	377		
<code>textcase</code> package	309		
<code>\textsc</code>	233		
<code>\textsmaller</code>	247		
<code>\texttt</code>	25, 129–132		
<code>\the</code>	112, 113, 127, 134, 184, 197, 216–223, 225, 227–230, 232–238, 240, 241, 244–246, 248–251, 253, 255, 258, 260, 262–270, 272, 274, 276–282, 284, 285, 287–292, 294–297, 299–304, 306–309		
<code>\theHglentrycounter</code> ..	8, 10, 11, 65, 67, 147		
<code>\theHglentrycounter</code>	8–11, 65, 67, 147		
<code>\theindex</code>	180		
<code>\thewrglossary</code>	23		
<code>\this@dialect</code>	329, 364		
<code>\thisgrptitle</code>	405		
<code>\toks@</code>	127, 184		
<code>\TrackedDialectClosestSubMatch</code>	36		
<code>tracklang</code> package	35, 36, 135, 334		
<code>\TrackLangGetDefaultScript</code>	363		
<code>\TrackLangIfHasDefaultScript</code>	363		
<code>\TrackLangRequireDialectPrefix</code> ..	363, 364		
U			
<code>\u</code>	330		
<code>\undef</code>	13, 14, 51, 185		
<code>\underline</code>	187		
<code>\unskip</code>	40, 50, 366		
<code>upgreek</code> package	333		
<code>\usepackage</code>	131, 132		
W			
<code>\warn@nomakeglossaries</code>	116		
<code>\warn@noprintglossary</code>	116, 119		
<code>wrglossary</code> (counter)	23		
<code>\write</code>	46, 102, 110, 115, 128, 129, 135		
X			
<code>\x</code>	127, 151		
<code>\xcapitalisewords</code>	172		
<code>\xdef</code>	119, 140		
<code>\xifinlist</code>	106		
<code>\xifinlistcs</code>	32		
<code>xindy</code>	407		
<code>xindy</code>	14, 114		
<code>xkeyval</code> package	5		
<code>\XKV@checkchoice</code>	60		
<code>\XKV@plfalse</code>	60		
<code>\XKV@resa</code>	60		
<code>\XKV@sttrue</code>	60		