

# **glossaries-extra.sty v1.32: documented code**

Nicola L.C. Talbot

Dickimaw Books

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

2018-05-24

## 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

<b>1</b>	<b>Main Package Code (glossaries-extra.sty)</b>	<b>5</b>
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	27
1.3	Modifications to Commands Provided by glossaries	38
1.3.1	Existence Checks	43
1.3.2	Document Definitions	50
1.3.3	Existing Glossary Style Modifications	56
1.3.4	Entry Formatting, Hyperlinks and Indexing	60
1.3.5	Entry Counting	96
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	152
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
<b>2</b>	<b>Style Adjustments (glossaries-extra-stylemods.sty)</b>	<b>365</b>
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

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

# 1 Main Package Code (glossaries-extra.sty)

## 1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/05/24 v1.32 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

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

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

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

Not already loaded, so pass options to glossaries.

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

Set the defaults.

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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

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

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

`glstr@recordsee` Does nothing by default.

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

`ultnumberformat`

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

`ultNumberFormat`

```

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

```

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

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

```

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

```

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

```

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

```

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

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



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

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

Save the label in case it's needed.

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

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

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

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

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

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings.

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

Increment associated counter.

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

Check if noindex option has been used.

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

Check if thevalue has been set.

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

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

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

Save the entry counter.

```
147         \glsxtr@saveentrycounter
```

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

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

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

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

No need to escape special characters.

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

glslink@prekeys

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

lslink@postkeys

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

lossadd@prekeys

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

ossadd@postkeys

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

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

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

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

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

```

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

```

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

```

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

```

`r@recordcounter`

```

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

```

`p@recordcounter`

```

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

```

`op@recordcounter`

```

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

```

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

```

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

```

unsrtglossaryunit

```

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

```

tr@setup@record Initialise.

```

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

```

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

```

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

```

addloclistfield

```

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

```

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

```

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

```

Add a key to store the group heading.

```

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

```

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

```

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

```

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

Need to provide conditional commands that are backward compatible:

if@glxtrdocdef

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

lsxtrdocdeftrue

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

sxtrdocdeffalse

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

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

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

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

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

```
324 {%
```

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

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

```
327   \fi
```

```
328 }
```

ocdefrestricted

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

oifexistsorwarn

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

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

indexcrossrefs

Automatically index cross references at the end of the document

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

```
332   \if@glxtrindexcrossrefs
```

```
333   \else
```

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

```
335   \fi
```

```
336 }
```

Switch off since this can increase the build time.

```
337 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

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

```

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

```

**iesExtraWarning** Allow users to suppress warnings.

```

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

```

**raWarningNoLine** Allow users to suppress warnings.

```

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

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

```

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

```

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

```

**postdot** Shortcut for `nopostdot=false`

```

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

```

**nopostdot** Needs to redefine `\@glxtr@defpostpunc`

```

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

```

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

```

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

```



```

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

```

`\glsxtrabbrvtype` Glossary type for abbreviations.

```

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

```

`\bbreviationsdef` Set by abbreviations option.

```

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

```

`\bbreviationsdef`

```

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

```

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

```

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

```

```

411 \fi
412 }%

```

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

```

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

```

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

```

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

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

```

Disable this command after it's been used.

```

442 \let\GlsXtrDefineAbbreviationShortcuts\relax
443 }

```

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

```

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

```

```

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

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

```

Disable this command after it's been used.

```

470 \let\GlsXtrDefineAcShortcuts\relax
471 }

```

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

```

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

```

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

**@setupshortcuts** Command used to set the shortcuts option.

```

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

```

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

```
485 \newcommand*{\@glxtr@shortcutsval}{\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   \glsdexists{#1}%
673   {%
674     \@glstrwrglossmark
675     \@glstr@org@doseeglossary{#1}{#2}%
676   }%
677 }
```

oindex@glossary

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

Define abbreviations glossaries if required.

```

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

```

Setup shortcuts if required.

```

703 \@glxtr@setupshortcuts

```

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

```

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

```

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

```

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

```

Now define the user command:

```

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

```

`@@do@wrglossary` Save original definition of `\@@do@wrglossary`.  
717 `\let\glxstr@org@@do@wrglossary\@@do@wrglossary`

`@@do@wrglossary` The new version adds code that can show a marker for debugging and increments the associated counter if enabled.  
718 `\newcommand*{\glxstr@@do@wrglossary}[1]{%`  
719 `\@@glxstrwrglossmark`  
720 `\glxstr@inc@wrglossaryctr{#1}%`  
721 `\glxstr@org@@do@wrglossary{#1}%`  
722 `}`

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

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

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

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

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

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

## 1.2 Extra Utilities

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

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

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

```

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

```

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

```

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

```

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

```

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

```

**vide@storagekey** Unstarred version.

```

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

```

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

```

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

```

Provide the no-link command if not already defined.

```

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

```

vide@storagekey Starred version.

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

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

`\GlsXtrFmtField`

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

`\GlsXtrFmtDefaultOptions`

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

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

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

`\@glxtrfmt` Unstarred form.

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

`\s@glxtrfmt` Starred form.

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

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

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

`\@glxtrfmt` Actual inner working.

```
791 \newcommand*{\@glxtrfmt}[4]{%
```

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

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

```

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

```

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

```

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

```

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

```

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

```

**glxtrentryfmt** No link or indexing.

```

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

```

**@glxtrentryfmt**

```

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

```

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

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

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

`glxtrfieldlistgadd` Similarly but uses `\listcsgadd`.

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

`glxtrfieldlistseadd` Similarly but uses `\listcseadd`.

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

`glxtrfieldlistxadd` Similarly but uses `\listcsxadd`.

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

Now provide commands to iterate over these lists.

`glxtrfielddolistloop`

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

`glxtrfieldforlistloop`

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

List element tests:

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

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

`glxtrfieldxifinlist` Expands item.

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

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

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

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

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

`lsxtrifhasfield` Unstarred version adds grouping.

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

`lsxtrifhasfield` Starred version omits grouping.

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

`rIfFieldNonZero` Designed for numeric fields.

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

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

Designed for numeric fields.

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



XtrIfFieldCmpNum

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

Designed for numeric fields.

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

sXtrIfFieldUndef

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

Just uses \ifcsundef.

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

\glxtrusefield

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

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

\Glsxtrusefield

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

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

\glxtrdeffield

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

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

glxtredeffield

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

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

etfieldifexists

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

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

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

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

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

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

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

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

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

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

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

`xGlsXtrSetField`

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

`eGlsXtrSetField`

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

`XtrIfFieldEqStr`

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

```

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

```

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

```

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

```

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

```

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

```

`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`.

```

965 \ifdef\foreignlanguage
966 {
967   \ifdef\GetTrackedDialectFromLanguageTag
968   {
969     \newcommand{\GlsXtrForeignText}[2]{%

```

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

```

970     \let\@glstr@org@currentfieldvalue\glscurrentfieldvalue
971     \glstrifhasfield{\GlsXtrForeignTextField}{#1}%
972     {%
973       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
974       {\glscurrentfieldvalue}{\@glstr@dialect}%

```

```

975      \let\@glxtr@locale\glscurrentfieldvalue
976      \let\glscurrentfieldvalue\@glxtr@org@currentfieldvalue
977      \ifdefempty\@glxtr@dialect
978      {%

```

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

```

979      \ifundef\TrackedDialectClosestSubMatch
980      {%
981      \GlossariesExtraWarning{Can't obtain dialect label
982      (tracklang v1.3.6+ required)}%
983      }%
984      {\let\@glxtr@dialect\TrackedDialectClosestSubMatch}%
985      }%
986      }%
987      \ifdefempty\@glxtr@dialect
988      {%

```

No tracked dialect found for the root language.

```

989      }%
990      {%

```

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

```

991      \ifcsundef{captions\@glxtr@dialect}{}%
992      {%

```

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

```

993      \IfTrackedDialectHasMapping{\@glxtr@dialect}%
994      {%
995      \edef\@glxtr@dialect{%
996      \GetTrackedDialectToMapping{\@glxtr@dialect}}%

```

Does a caption hook exist for this?

```

997      \ifcsundef{captions\@glxtr@dialect}{}%
998      {%

```

No mapping. Try root language label instead.

```

999      \ifcsundef{captions\@tracklang@lang}{}%
1000      {%
1001      \let\@glxtr@dialect\@tracklang@lang
1002      }%
1003      }%
1004      }%
1005      {%

```

No mapping. Try root language label instead.

```

1006      \ifcsundef{captions\@tracklang@lang}{}%
1007      {%
1008      \let\@glxtr@dialect\@tracklang@lang
1009      }%
1010      }%
1011      }%

```

```

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

```

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

```

1034 \newcommand*{\GlsXtrForeignTextField}{userii}

```

**nDialectWarning**

```

1035 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
1036     \GlossariesExtraWarning{Can't determine valid dialect label
1037         for locale '#1' (root language: #2)}%
1038 }

```

**\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.

```

1039 \ifdef\GlsEntryCounterLabelPrefix
1040 {%
1041     \newcommand*{\glstrpageref}[1]{%
1042         \ifglssentrycounter
1043             \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1044         \else
1045             \ifglssubentrycounter
1046                 \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1047             \else
1048                 \gls{#1}%
1049             \fi
1050         \fi

```

```

1051 }
1052 }%
1053 {%
1054   \newcommand*{\glxtrpageref}[1]{%
1055     \ifglentrycounter
1056       \pageref{glentry-\glstoklabel{#1}}%
1057     \else
1058       \ifglssubentrycounter
1059         \pageref{glentry-\glstoklabel{#1}}%
1060       \else
1061         \gls{#1}%
1062       \fi
1063     \fi
1064   }
1065 }%

```

lossary preamble

```

1066 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1067   \ifcsdef{glolist@#1}%
1068   {%
1069     \ifcsundef{@glossarypreamble@#1}%
1070     {\csdef{@glossarypreamble@#1}{}}%
1071   }%
1072   \csappto{@glossarypreamble@#1}{#2}%
1073 }%
1074 {%
1075   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1076 }%
1077 }

```

lossary preamble

```

1078 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
1079   \ifcsdef{glolist@#1}%
1080   {%
1081     \ifcsundef{@glossarypreamble@#1}%
1082     {\csdef{@glossarypreamble@#1}{}}%
1083   }%
1084   \cspreto{@glossarypreamble@#1}{#2}%
1085 }%
1086 {%
1087   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1088 }%
1089 }

```

## 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`

```
1090 \renewcommand*{\longnewglossaryentry}{%
1091   \@ifstar\@glstr@s@longnewglossaryentry\@glstr@longnewglossaryentry
1092 }
```

`ewglossaryentry` Starred version.

```
1093 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1094   \glstoifnoexists{#1}%
1095   {%
1096     \bgroup
1097     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1098     \long\def\@newglossaryentryprehook{%
1099       \long\def\@glo@desc{#3}%
1100       \@org@newglossaryentryprehook
1101     }%
1102     \renewcommand*{\gl@assign@desc}[1]{%
1103       \global\cslet{glo@\glsetoklabel{#1}@desc}{\@glo@desc}%
1104       \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
1105     }
1106     \gl@defglossaryentry{#1}{#2}%
1107   \egroup
1108 }%
1109 }
```

`ewglossaryentry` Unstarred version.

```
1110 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1111   \glstoifnoexists{#1}%
1112   {%
1113     \bgroup
1114     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1115     \long\def\@newglossaryentryprehook{%
1116       \long\def\@glo@desc{#3\glstrpostlongdescription}%
1117       \@org@newglossaryentryprehook
1118     }%
1119     \renewcommand*{\gl@assign@desc}[1]{%
1120       \global\cslet{glo@\glsetoklabel{#1}@desc}{\@glo@desc}%
1121       \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
1122     }
1123     \gl@defglossaryentry{#1}{#2}%
1124   \egroup
1125 }%
1126 }
```

The following is different from the base glossaries.sty:

```
1121   \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
1122   }
1123   \gl@defglossaryentry{#1}{#2}%
1124 \egroup
1125 }%
1126 }
```

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

```
1127 \newcommand*{\glstrpostlongdescription}{\leavevmode\unskip\nopostdesc}
```

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

ignoredglossary Redefine to check for star.

```
1128 \renewcommand{\newignoredglossary}{%
1129 \ifstar\glstr@s@newignoredglossary\glstr@org@newignoredglossary
1130 }
```

ignoredglossary The original definition is patched to check for existence.

```
1131 \newcommand*{\glstr@org@newignoredglossary}[1]{%
1132 \ifcsdef{glolist@#1}
1133 {%
1134 \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1135 }%
1136 {%
1137 \ifdefempty\@ignored@glossaries
1138 {%
1139 \edef\@ignored@glossaries{#1}%
1140 }%
1141 {%
1142 \eappto\@ignored@glossaries{,#1}%
1143 }%
1144 \csgdef{glolist@#1}{,}%
1145 \ifcsundef{gls@#1@entryfmt}%
1146 {%
1147 \defglsentryfmt[#1]{\glsentryfmt}%
1148 }%
1149 {}%
1150 \ifdefempty\@gls@nohyperlist
1151 {%
1152 \renewcommand*{\@gls@nohyperlist}{#1}%
1153 }%
1154 {%
1155 \eappto\@gls@nohyperlist{,#1}%
1156 }%
1157 }%
1158 }
```

ignoredglossary Starred form.

```
1159 \newcommand*{\glstr@s@newignoredglossary}[1]{%
1160 \ifcsdef{glolist@#1}
1161 {%
1162 \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1163 }%
1164 {%
1165 \ifdefempty\@ignored@glossaries
```



```

1166   {%
1167     \edef\@ignored@glossaries{#1}%
1168   }%
1169   {%
1170     \eappto\@ignored@glossaries{,#1}%
1171   }%
1172   \csgdef{glolist@#1}{,}%
1173   \ifcsundef{gls@#1@entryfmt}%
1174   {%
1175     \defglsentryfmt[#1]{\glsentryfmt}%
1176   }%
1177   {}%
1178 }%
1179 }

```

`\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.

```

1180 \glsifusetranslator
1181 {%
1182   \renewcommand*{\glsettoctitle}[1]{%
1183     \ifcsdef{gls@tr@set@#1@toctitle}%
1184     {%
1185       \csuse{gls@tr@set@#1@toctitle}%
1186     }%
1187     {%
1188       \ifcsdef{@glotype@#1@title}%
1189       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1190       {\def\glossarytoctitle{\glossarytitle}}%
1191     }%
1192   }%
1193 }
1194 {
1195   \renewcommand*{\glsettoctitle}[1]{%
1196     \ifcsdef{@glotype@#1@title}%
1197     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1198     {\def\glossarytoctitle{\glossarytitle}}%
1199   }
1200 }

```

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

```

1201 \newcommand{\provideignoredglossary}{%
1202   \@ifstar\glstr@s@provideignoredglossary\glstr@provideignoredglossary
1203 }

```

`ignoredglossary` Unstarred version.

```

1204 \newcommand*{\glstr@provideignoredglossary}[1]{%
1205   \ifcsdef{glolist@#1}
1206   {}%
1207   {%

```

```

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

```

`ignoredglossary` Starred form.

```

1230 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1231 \ifcsdef{glolist@#1}
1232 {}%
1233 {%
1234 \ifdefempty\@ignored@glossaries
1235 {%
1236 \edef\@ignored@glossaries{#1}%
1237 }%
1238 {%
1239 \eappto\@ignored@glossaries{,#1}%
1240 }%
1241 \csgdef{glolist@#1}{,}%
1242 \ifcsundef{gls@#1@entryfmt}%
1243 {%
1244 \defglsentryfmt[#1]{\glsentryfmt}%
1245 }%
1246 {}%
1247 }%
1248 }

```

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

```

1249 \newcommand*{\glxtrcopytoglossary}[2]{%
1250 \glsdoifexists{#1}%
1251 {%

```

```

1252 \ifcsdef{glolist@#2}
1253 {%
1254 \cseappto{glolist@#2}{#1,}%
1255 }%
1256 {%
1257 \glxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1258 }%
1259 }%
1260 }

```

### 1.3.1 Existence Checks

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

```

1261 \renewcommand{\glsdoidexists}[2]{%
1262 \ifglstryexists{#1}{#2}%
1263 {%

```

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

```

1264 \edef\glslabel{\glstetoklabel{#1}}%
1265 \glxtrundefaction{Glossary entry ‘\glslabel’
1266 has not been defined}{You need to define a glossary entry before
1267 you can reference it.}%
1268 }%
1269 }

```

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

```

1270 \renewcommand{\glsdoidnoexists}[2]{%
1271 \ifglstryexists{#1}{%
1272 \glxtrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1273 has already been defined}{}}{#2}%
1274 }

```

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

```

1275 \ifdef\glsdoidexistsordo
1276 {%
1277 \renewcommand{\glsdoidexistsordo}[3]{%
1278 \ifglstryexists{#1}{#2}%
1279 {%
1280 \glxtrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1281 has not been defined}{You need to define a glossary entry
1282 before you can use it.}%
1283 #3%
1284 }%
1285 }%
1286 }
1287 {%
1288 \glxtr@warnonexistsordo\glsdoidexistsordo

```

```

1289 \newcommand{\glsdoifexistsordo}[3]{%
1290   \ifglsentryexists{#1}{#2}%
1291   {%
1292     \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1293     has not been defined}{You need to define a glossary entry
1294     before you can use it.}%
1295     #3%
1296   }%
1297 }%
1298 }

```

`arynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```

1299 \ifdef\doifglossarynoexistsordo
1300 {%
1301   \renewcommand{\doifglossarynoexistsordo}[3]{%
1302     \ifglossaryexists{#1}%
1303     {%
1304       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1305       #3%
1306     }%
1307     {#2}%
1308   }%
1309 }
1310 {%
1311   \glstr@warnonexistsordo\doifglossarynoexistsordo
1312   \newcommand{\doifglossarynoexistsordo}[3]{%
1313     \ifglossaryexists{#1}%
1314     {%
1315       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1316       #3%
1317     }%
1318     {#2}%
1319   }%
1320 }
1321

```

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.

```

1322 \appto\newglossaryentryposthook{%
1323   \ifdefvoid\@glo@see
1324   {\csxdef{glo@\@glo@label @see}{}}%
1325   {%
1326     \csxdef{glo@\@glo@label @see}{\@glo@see}%
1327     \if@glstr@autoseeindex
1328       \@glstr@autoindexcrossrefs
1329     \fi

```

```

1330 }%
1331 }
1332 \appto\@gls@keymap{,{see}{see}}

\glxtrusesee Apply \glsseeformat to the see key if not empty.
1333 \newcommand*{\glxtrusesee}[1]{%
1334   \glsdoifexists{#1}%
1335   {%
1336     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1337     \ifdefempty\@glo@see
1338     {}%
1339     {%
1340       \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1341     }%
1342   }%
1343 }

\glxtr@usesee
1344 \newcommand*{\glxtr@usesee}[1][\seename]{%
1345   \@glxtr@usesee[#1]%
1346 }

\@glxtr@usesee
1347 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1348   \glxtruseseeformat{#1}{#2}%
1349 }

xtruseseeformat The format used by \glxtrusesee. The first argument is the tag (such as \seename). The
second argument is the comma-separated list of cross-referenced labels.
1350 \newcommand*{\glxtruseseeformat}[2]{%
1351   \glsseeformat{#1}{#2}{}%
1352 }

lsseeitemformat glossaries originally defined \glsseeitemformat to use \glsentryname but in v3.0 this was
switched to use \glsentrytext due to problems occurring with the name field being sani-
tized. Since this is no longer a problem, glossaries-extra restores the original definition as it
makes more sense to use the name in the cross-reference list. This still uses \glsaccessstext
for abbreviations.
1353 \renewcommand*{\glsseeitemformat}[1]{%
1354   \ifglshashshort{\glslabel}{\glsaccessstext{#1}}{\glsaccessname{#1}}%
1355 }

lsxtruseseealso Apply \glsseeformat to the seealso key if not empty. There's no optional tag to worry about
here.
1356 \newcommand*{\glxtruseseealso}[1]{%
1357   \glsdoifexists{#1}%
1358   {%
1359     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%

```

```

1360 \ifdefempty\@glo@see
1361 {}%
1362 {%
1363 \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
1364 }%
1365 }%
1366 }

```

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

```

1367 \newcommand*{\glxtruseseealsoformat}[1]{%
1368 \glseeformat[\seealso]{#1}{}%
1369 }

```

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

```

1370 \newrobustcmd{\glxtrseelist}[1]{%
1371 \edef\@glo@tmp{\noexpand\glseeelist{#1}}\@glo@tmp
1372 }

```

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

```

1373 \providecommand{\seealso}{see also}

```

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

```

1374 \ifdef\@xdycrossrefhook
1375 {
    Add the cross-reference class definition to the hook.
1376 \appto\@xdycrossrefhook{%
1377 \write\glswrite{(define-crossref-class \string"seealso\string"
1378 :unverified )}%
1379 \write\glswrite{(markup-crossref-list
1380 :class \string"seealso\string"^^J\space\space\space
1381 :open \string"\string\glxtruseseealsoformat\glsoopenbrace\string"
1382 :close \string"\glsclosebrace\string")}%
1383 }

```

Append to class list.

```

1384 \appto\@xdylocationclassorder{\space\string"seealso\string"}

```

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

```

1385 \newrobustcmd*{\glxtrindexseealso}[2]{%
1386 \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1387 \@glxtr@recordsee{#1}{#2}%
1388 \fi
1389 \glsoifexists{#1}%

```

```

1390  {%
1391    \@glsxtrwrglossmark
1392    \def\@gls@xref{#2}%
1393    \@onelevel@sanitize\@gls@xref
1394    \@gls@checkmkidxchars\@gls@xref
1395    \gls@glossary{\csname glo@#1@type\endcsname}{%
1396      (indexentry
1397        :tkey (\csname glo@#1@index\endcsname)
1398        :xref (\string"\@gls@xref\string")
1399        :attr \string"seealso\string"
1400      )
1401    }%
1402  }%
1403 }
1404 }
1405 {
  xindy not in use or glossaries version too old to support this.
1406 \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
1407 }

```

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`.

```

1408 \ifdef\gls@set@xr@key
1409 {

```

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

```

1410 \define@key{glossentry}{alias}{%
1411   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1412 }
1413 \define@key{glossentry}{seealso}{%
1414   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1415 }

```

Add to the key mappings.

```

1416 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1417 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1418 \appto\@newglossaryentryposthook{%
1419   \ifdefvoid\@glo@seealso
1420     {\csxdef{glo@\@glo@label @seealso}{}}%
1421     {%
1422       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%

```

```

1423     \if@glstr@autoseeindex
1424     \@glstr@autoindexcrossrefs
1425     \fi
1426 }%

```

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

```

1427 \ifdefvoid\@glo@alias
1428 {\csxdef{glo@\@glo@label @alias}{}}%
1429 {%
1430 \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1431 }%
1432 }

```

Provide user-level commands to access the values.

`\glstralias`

```

1433 \newcommand*{\glstralias}[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1434 \newcommand*{\glstrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1435 \appto\@glo@autoseehook{%
1436 \ifdefvoid\@glo@alias
1437 {%
1438 \ifdefvoid\@glo@seealso
1439 }%
1440 {%
1441 \edef\@do@glssee{\noexpand\glstrindexseealso
1442 {\@glo@label}{\@glo@seealso}}%
1443 \@do@glssee
1444 }%
1445 }%
1446 {%

```

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

```

1447 \ifdefvoid\@glo@see
1448 {%
1449 \edef\@do@glssee{\noexpand\glssee{\@glo@label}{\@glo@alias}}%
1450 \@do@glssee
1451 }%
1452 }%
1453 }%
1454 }%
1455 }
1456 {

```

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

`\glstralias`

```

1457 \glsaddstoragekey*{alias}{\glstralias}

```



trseealsolabels

```
1458 \glsaddstoragekey*{seealso}{}{\glstrseealsolabels}
```

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.

```
1459 \appto\@newglossaryentryposthook{%
1460   \ifcsvoid{glo@\@glo@label @alias}%
1461   {%
1462     \ifcsvoid{glo@\@glo@label @seealso}%
1463     {}%
1464     {%
1465       \edef\@do@glsssee{\noexpand\glstrindexseealso
1466         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1467       \@do@glsssee
1468     }%
1469   }%
1470   {%
```

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

```
1471   \ifdefvoid\@glo@see
1472   {%
1473     \edef\@do@glsssee{\noexpand\glsssee
1474       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1475     \@do@glsssee
1476   }%
1477   {}%
1478 }%
1479 }
1480 }
```

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

```
1481 \AtEndDocument{\if\glstrindexcrossrefs\glstraddallcrossrefs\fi}
```

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

```
1482 \newcommand*\glstraddallcrossrefs{%
1483   \forallglossaries{\@glo@type}%
1484   {%
1485     \forglsentries[\@glo@type]{\@glo@label}%
1486     {%
1487       \ifglssused{\@glo@label}%
1488       {\expandafter\@glstr@addunusedxrefs\expandafter{\@glo@label}}{}%
1489     }%
1490   }%
1491 }
```

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

```

1492 \newcommand*{\@glsxtr@addunusedxrefs}[1]{%
1493   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@see}%
1494   \ifdefvoid\@glo@see
1495   {}%
1496   {%
1497     \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
1498   }%
1499   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@seealso}%
1500   \ifdefvoid\@glo@see
1501   {}%
1502   {%
1503     \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
1504   }%
1505 }

```

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

```

1506 \newcommand*{\glsxtr@addunused}[1][]{%
1507   \@glsxtr@addunused
1508 }

```

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

```

1509 \def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
1510   \@for\@glsxtr@label:=#1\do
1511   {%
1512     \ifglsused{\@glsxtr@label}{}%
1513     {%
1514       \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1515       \glsunset{\@glsxtr@label}%
1516       \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1517     }%
1518   }%
1519 }

```

`xtrunusedformat`

```

1520 \newcommand*{\glsxtrunusedformat}[1]{\unskip}

```

### 1.3.2 Document Definitions

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

```

1521 \ifdef\gls@begindocdefs
1522 {%
1523   \renewcommand*{\gls@begindocdefs}{%
1524     \ifnum\@glsxtr@docdefval=1\relax
1525     \@gls@enablesavenonumberlist

```

```

1526     \edef\@gls@restoreat{%
1527         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1528     \makeatletter
1529     \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1530     \@gls@restoreat
1531     \undef\@gls@restoreat
1532     \gls@defdocnewglossaryentry
1533 \fi
1534 }
1535 }
1536 {}

```

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

```

1537 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1538 \renewcommand{\makenoidxglossaries}{%
1539     \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1540     {%
1541         \glsxtr@orgmakenoidxglossaries

```

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

```

1542     \renewcommand{\@do@seeglossary}[2]{%
1543         \@glsxtrwrglossmark
1544         \edef\@gls@label{\glsdetoklabel{##1}}%
1545         \protected@write\@auxout{}{%
1546             \string\@gls@reference
1547             {\csname glo@\@gls@label @type\endcsname}%
1548             {\@gls@label}%
1549             {%
1550                 \string\glsseeformat##2}%
1551             }%
1552         }%
1553     }%

```

Check for docdefs=restricted:

```

1554     \if@glsxtrdocdefrestricted

```

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

```

1555     \renewcommand*{\@gls@reference}[3]{%
1556         \ifcsundef{\glsref@##1}{\csgdef{\glsref@##1}{-}}{}%
1557         \ifinlistcs{##2}{\glsref@##1}%
1558         {}%
1559         {\listcsgadd{\glsref@##1}{##2}}%
1560         \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1561         {\csgdef{glo@\glsdetoklabel{##2}@loclist}{-}}{}%
1562         {}%
1563         \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1564     }%

```

```

1565 \else
    Disable document definitions.
1566 \glstrdocdeffalse
1567 \fi
1568 \disable@keys{glossaries-extra}{docdef}%
1569 }%
1570 {%
1571 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1572 not permitted\MessageBreak
1573 with record=\glstr@record@setting\space package option}%
1574 {You may only use \string\makenoidxglossaries\space with the
1575 record=off option}%
1576 }%
1577 }

```

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

```

1578 \renewcommand*{\gls@defdocnewglossaryentry}{%
1579 \ifcase\glstr@docdefval
    docdef=false:
1580 \renewcommand*{\newglossaryentry}[2]{%
1581 \PackageError{glossaries-extra}{Glossary entries must
1582 be \MessageBreak defined in the preamble with \MessageBreak
1583 package option ‘docdef=false’\MessageBreak(consider using
1584 ‘docdef=restricted’)}{Move your glossary definitions to
1585 the preamble. You can also put them in a \MessageBreak separate file
1586 and load them with \string\loadglsentries.}%
1587 }%
1588 \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.

```

1589 \let\gls@checkseeallowed\relax
1590 \let\newglossaryentry\new@term
1591 \or

```

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

```

1592 \let\gls@checkseeallowed\relax
1593 \fi
1594 }%

```

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`

```

1595 \newcommand*{\GlsXtrEnableOnTheFly}{%
1596 \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1597 }

```

`\GlsXtrEnableOnTheFly` 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.)

```

1598 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1599   \renewcommand*{\glsdetoklabel}[1]{%
1600     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1601     {%
1602       \expandafter\detokenize\expandafter{##1}%
1603     }%
1604     {\detokenize{##1}}}%
1605   }%
1606   \@GlsXtrEnableOnTheFly
1607 }
1608 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1609   \expandafter\if\glsbackslash#1%
1610     #3%
1611   \else
1612     #4%
1613   \fi
1614 }

```

`\glsxtrstarflywarn`

```

1615 \newcommand*{\glsxtrstarflywarn}{%
1616   \GlossariesExtraWarning{Experimental starred version of
1617   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1618   read the warnings in the glossaries-extra user manual)}}%
1619 }

```

`\GlsXtrEnableOnTheFly`

```

1620 \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`

```

1621 \newcommand*{\glsxtrcat}{general}

```

`\glsxtr`

```

1622 \newcommand*{\glsxtr}[1] [] {%
1623   \def\glsxtr@keylist{##1}%
1624   \@glsxtr
1625 }

```

`\@glsxtr`

```

1626 \newcommand*{\@glsxtr}[2] [] {%
1627   \ifglsentryexists{##2}%

```

```

1628   {%
1629     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1630   }%
1631   {%
1632     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1633       description={\nopostdesc},##1}%
1634   }%
1635   \expandafter\gls\expandafter[\glstr@keylist]{##2}%
1636 }

```

\Glsxtr

```

1637 \newcommand*{\Glsxtr}[1] [] {%
1638   \def\glstr@keylist{##1}%
1639   \@Glsxtr
1640 }

```

\@Glsxtr

```

1641 \newcommand*{\@Glsxtr}[2] [] {%
1642   \ifglstryexists{##2}%
1643   {%
1644     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1645   }%
1646   {%
1647     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1648       description={\nopostdesc},##1}%
1649   }%
1650   \expandafter\gls\expandafter[\glstr@keylist]{##2}%
1651 }

```

\glstrpl

```

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

```

\@glstrpl

```

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

```

\Glsxtrpl

```

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

```

\@Glsxtrpl

```

1671 \newcommand*{\@Glsxtrpl}[2][]{%
1672   \ifglstryentryexists{##2}
1673   {%
1674     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1675   }%
1676   {%
1677     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
1678       description={\nopostdesc},##1}%
1679   }%
1680   \expandafter\Glspl\expandafter[\glxtr@keylist]{##2}%
1681 }

```

\GlsXtrWarning

```

1682 \newcommand*{\GlsXtrWarning}[2]{%
1683   \def\@glxtr@optlist{##1}%
1684   \@onelevel@sanitize\@glxtr@optlist
1685   \GlossariesExtraWarning{The options '\@glxtr@optlist' have
1686     been ignored for entry '##2' as it has already been defined}%
1687 }

```

Disable commands after the glossary:

```

1688 \renewcommand\@printglossary[2]{%
1689   \def\@glxtr@printglossopts{##1}%
1690   \@glxtr@orgprintglossary{##1}{##2}%
1691   \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
1692   \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1693   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1694   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1695 }

```

abledflycommand

```

1696 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1697   \PackageError{glossaries-extra}%
1698   {\string##1\space can't be used after any of the \MessageBreak
1699     glossaries have been displayed}%
1700   {The on-the-fly commands enabled by
1701     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1702     before the glossaries. If you want to use any entries \MessageBreak
1703     after any of the glossaries, you must use the standard \MessageBreak
1704     method of first defining the entry and then using the \MessageBreak
1705     entry with commands like \string\gls}%
1706   \@@glxtr@disabledflycommand
1707 }%

```

```
1708 \newcommand*{\@glxtr@disabledflycommand}[2][\@##2]
```

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

```
1709 \let\GlsXtrEnableOnTheFly\relax
1710 }
1711 \@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.

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

```
1712 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}
```

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

`\setglossarystyle`

```
1713 \renewcommand*{\setglossarystyle}[1]{%
1714   \ifcsundef{@glsstyle@#1}%
1715   {%
1716     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1717   }%
1718   {%
1719     \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
1720   \protected@edef\@glxtr@current@style{#1}%
1721   }%
1722   \ifx\@glossary@default@style\relax
1723     \protected@edef\@glossary@default@style{#1}%
1724   \fi
1725 }
```

In case we have an old version of glossaries:

```
1726 \ifdef\@glossary@default@style
1727 {}
1728 {%
1729   \let\@glossary@default@style\relax
1730 }
```

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

```
1731 \ifdef\glslistdottedwidth
1732 {%
1733   \ifdim\glslistdottedwidth=.5\hsize
1734     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1735     \AtBeginDocument{%
1736       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
```



```

1737     \setlength{\glslistdottedwidth}{.5\columnwidth}%
1738     \fi
1739   }%
1740 \fi
1741 }
1742 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

1743 \ifdef\glsdescwidth
1744 {%
1745   \ifdim\glsdescwidth=.6\hsize
1746     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1747     \AtBeginDocument{%
1748       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1749         \setlength{\glsdescwidth}{.6\columnwidth}%
1750       \fi
1751     }%
1752   \fi
1753 }
1754 {}%

```

and for \glspagelistwidth:

\glspagelistwidth

```

1755 \ifdef\glspagelistwidth
1756 {%
1757   \ifdim\glspagelistwidth=.1\hsize
1758     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1759     \AtBeginDocument{%
1760       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1761         \setlength{\glspagelistwidth}{.1\columnwidth}%
1762       \fi
1763     }%
1764   \fi
1765 }
1766 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```

1767 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1768 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1769   \glsnonumberlistfalse
1770   \renewcommand*{\glossaryentrynumbers}[1]{%
1771     \ifglstryentryexists{\glscurrententrylabel}%
1772     {%
1773       \@glsxtrpreloctag
1774       \GlsXtrFormatLocationList{#1}%
1775       \@glsxtrpostloctag
1776       \gls@save@numberlist{#1}%

```

```

1777     }{}%
1778 }%
1779 \else
1780   \glsnonnumberlisttrue
1781   \renewcommand*{\glossaryentrynumbers}[1]{%
1782     \ifglsentryexists{\glscurrententrylabel}%
1783     {%
1784       \gls@save@numberlist{#1}%
1785     }{}%
1786 }%
1787 \fi

```

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

```

1788 \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**

```

1789 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1790   \let\@glstrpreloctag\@glstrpreloctag
1791   \let\@glstrpostloctag\@glstrpostloctag
1792   \renewcommand*{\@glstr@pagetag}{#1}%
1793   \renewcommand*{\@glstr@pagetag}{#2}%
1794   \renewcommand*{\@glstr@savepreloctag}[2]{%
1795     \csgdef{\@glstr@preloctag@##1}{##2}%
1796   }%
1797   \renewcommand*{\@glstr@doloctag}{%
1798     \ifcsundef{\@glstr@preloctag\glscurrententrylabel}%
1799     {%
1800       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1801         Rerun required}%
1802     }%
1803     {%
1804       \csuse{\@glstr@preloctag\glscurrententrylabel}%
1805     }%
1806   }%
1807 }
1808 \@onlypreamble\GlsXtrEnablePreLocationTag

```

**glstrpreloctag**

```

1809 \newcommand*{\@glstrpreloctag}{%
1810   \let\@glstr@org@delimN\delimN
1811   \let\@glstr@org@delimR\delimR
1812   \let\@glstr@org@glsgignore\glsgignore

```

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

```

1813 \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
1814 \renewcommand*\@delimN{%
1815   \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
1816   \@glxtr@org@delimN}%
1817 \renewcommand*\@delimR{%
1818   \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
1819   \@glxtr@org@delimR}%
1820 \renewcommand*\@glsglignore}[1]{%
1821   \gdef\@glxtr@thisloctag{\relax}%
1822   \@glxtr@org@glsglignore{##1}}%
1823 \@glxtr@doloctag
1824 }

```

glxtrpreloctag

```

1825 \newcommand*\@glxtrpreloctag{}

```

@glxtr@pagetag

```

1826 \newcommand*\@glxtr@pagetag{}%

```

glxtr@pagetag

```

1827 \newcommand*\@glxtr@pagetag{}%

```

lsxtrpostloctag

```

1828 \newcommand*\@glxtrpostloctag{%
1829   \let\delimN\@glxtr@org@delimN
1830   \let\delimR\@glxtr@org@delimR
1831   \let\glsglignore\@glxtr@org@glsglignore
1832   \protected@write\@auxout{%
1833     {\string\@glxtr@savepreloctag{\glscurrententrylabel}\@glxtr@thisloctag}}%
1834 }

```

lsxtrpostloctag

```

1835 \newcommand*\@glxtrpostloctag{}

```

lsxtr@preloctag

```

1836 \newcommand*\@glxtr@savepreloctag}[2]{%
1837   \protected@write\@auxout{%
1838     \string\providecommand\string\@glxtr@savepreloctag[2]{}}

```

glxtr@doloctag

```

1839 \newcommand*\@glxtr@doloctag{}

```

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

```

1840 \renewcommand*\KV@printgloss@nonumberlist}[1]{%
1841   \XKV@plfalse
1842   \XKV@sttrue

```

```

1843 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1844 {%
1845   \csname glsnonumberlist\XKV@resa\endcsname
1846   \ifglsnonumberlist
1847     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1848   \else
1849     \def\glossaryentrynumbers##1{%
1850       \@glstrpreloctag
1851       \GlsXtrFormatLocationList{##1}%
1852       \@glstrpostloctag
1853       \gls@save@numberlist{##1}}%
1854   \fi
1855 }%
1856 }

```

### 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.

```

1857 \renewcommand*{\glsentryfmt}{%
1858   \ifglshasshort{\glslabel}{\glsssetabbrvfmt{\glscategory{\glslabel}}{}}%
1859   \glsifregular{\glslabel}%
1860   {\glstrregularfont{\glsgenentryfmt}}%
1861   {%
1862     \ifglshasshort{\glslabel}%
1863     {\glstrabbreviationfont{\glstrgenabbrvfmt}}%
1864     {\glstrregularfont{\glsgenentryfmt}}%
1865   }%
1866 }

```

**glstrregularfont** Font used for regular entries.

```
1867 \newcommand*{\glstrregularfont}[1]{#1}
```

**glstrabbreviationfont** Font used for abbreviation entries.

```
1868 \newcommand*{\glstrabbreviationfont}[1]{#1}
```

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

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

```
1869 \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).

```
1870 \@glsxtr@record{#2}{#3}{glslink}%
1871 \glsdoifexists{#3}%
1872 {%
```

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).

```
1873 \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1874 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1875 \def\glscustomtext{#4}%
1876 \@glsxtr@field@linkdefs
1877 #1%
1878 \@gls@link[#2]{#3}{#4}%
1879 \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
1880 }%
1881 \glspostlinkhook
1882 }
```

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

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

```
1883 \let\@glsxtr@org@gls@\@gls@
1884 \def\@gls@#1#2{%
1885 \@glsxtr@record{#1}{#2}{glslink}%
1886 \@glsxtr@org@gls@{#1}{#2}%
1887 }%
```

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

```
1888 \let\@glsxtr@org@glspl@\@glspl@
1889 \def\@glspl@#1#2{%
1890 \@glsxtr@record{#1}{#2}{glslink}%
1891 \@glsxtr@org@glspl@{#1}{#2}%
1892 }%
```

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

```
1893 \let\@glsxtr@org@Gls@\@Gls@
1894 \def\@Gls@#1#2{%
1895 \@glsxtr@record{#1}{#2}{glslink}%
1896 \@glsxtr@org@Gls@{#1}{#2}%
1897 }%
```

**\@Glspl@** Save the original definition and redefine.

```
1898 \let\@glsxtr@org@Glspl@\@Glspl@
1899 \def\@Glspl@#1#2{%
1900 \@glsxtr@record{#1}{#2}{glslink}%
```

```

1901 \@glxtr@org@GLspl@{#1}{#2}%
1902 }%

```

\@GLS@ Save the original definition and redefine.

```

1903 \let\@glxtr@org@GLS@\@GLS@
1904 \def\@GLS@#1#2{%
1905   \@glxtr@record{#1}{#2}{glslink}%
1906   \@glxtr@org@GLS@{#1}{#2}%
1907 }%

```

\@GLSpl@ Save the original definition and redefine.

```

1908 \let\@glxtr@org@GLSpl@\@GLSpl@
1909 \def\@GLSpl@#1#2{%
1910   \@glxtr@record{#1}{#2}{glslink}%
1911   \@glxtr@org@GLSpl@{#1}{#2}%
1912 }%

```

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

```

1913 \renewcommand*{\@glsdisp}[3][{}]{%
1914   \@glxtr@record{#1}{#2}{glslink}%
1915   \glsdoifexists{#2}{%
1916     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1917     \let\glsifplural\@secondoftwo
1918     \let\glscapscase\@firstofthree
1919     \def\glscustomtext{#3}%
1920     \def\glsinsert{}%
1921     \def\@glo@text{\csname gls@glstype @entryfmt\endcsname}%
1922     \@gl@link[#1]{#2}{\@glo@text}%
1923     \ifKV@glslink@local
1924       \glslocalunset{#2}%
1925     \else
1926       \glsunset{#2}%
1927     \fi
1928   }%
1929   \glspostlinkhook
1930 }

```

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

```

1931 \renewcommand*{\@gl@link}[3][{}]{%
1932   \@glxtr@record{#1}{#2}{glslink}%
1933   \glsdoifexistsordo{#2}%
1934   {%
1935     \let\do@gl@link@checkfirsthyper\relax
1936     \@gl@link[#1]{#2}{#3}%
1937   }%
1938   {%
1939     \glstextformat{#3}%
1940   }%

```

```

1941 \glspostlinkhook
1942 }

```

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

```

1943 \newcommand*{\glsxtrinitwrgloss}{%
1944 \glusifattribute{\glslabel}{wrgloss}{after}%
1945 {%
1946 \glsxtrinitwrglossbeforefalse
1947 }%
1948 {%
1949 \glsxtrinitwrglossbeforetrue
1950 }%
1951 }

```

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

```

1952 \newif\ifglsxtrinitwrglossbefore
1953 \glsxtrinitwrglossbeforetrue

```

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

```

1954 \define@choicekey{glslink}{wrgloss}%
1955 [\@glsxtr@wrglossval\@glsxtr@wrglossnr]%
1956 {before,after}%
1957 {%
1958 \ifcase\@glsxtr@wrglossnr\relax
1959 \glsxtrinitwrglossbeforetrue
1960 \or
1961 \glsxtrinitwrglossbeforefalse
1962 \fi
1963 }

```

```

1964 \define@key{glslink}{thevalue}{\def\@glsxtr@thevalue{#1}}

```

```

1965 \define@key{glslink}{theHvalue}{\def\@glsxtr@theHvalue{#1}}

```

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

```

1966 \define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
1967 \glsxtr@hyperoutsidettrue

```

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

```

1968 \define@key{glslink}{textformat}{%
1969 \ifcsdef{#1}
1970 {%
1971 \letcs{\@glsxtr@local@textformat}{#1}%
1972 }%
1973 {%
1974 \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
1975 }%
1976 }

```

```

1977 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

nithyperoutside Set the default if the hyperoutside is omitted.
1978 \newcommand*{\glxtrnithyperoutside}{%
1979 \glsifattribute{\glslabel}{hyperoutside}{false}%
1980 {%
1981 \glxtr@hyperoutsidefalse
1982 }%
1983 {%
1984 \glxtr@hyperoutsidetrue
1985 }%
1986 }

r@inc@linkcount Does nothing by default.
1987 \newcommand*{\glxtr@inc@linkcount}{%

slinkpresetkeys User hook performed immediately before options are set. Does nothing by default.
1988 \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.
1989 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
1990 \protected@edef\@glxtr@tmp{#2}%
1991 \expandafter#1\expandafter{\@glxtr@tmp}%
1992 }

\@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.
1993 \def\@gls@link[#1]#2#3{%
1994 \leavevmode
1995 \edef\glslabel{\glsdetoklabel{#2}}%
1996 \def\@gls@link@opts{#1}%
1997 \let\@gls@link@label\glslabel
1998 \let\@glsnumberformat\@glxtr@defaultnumberformat
1999 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2000 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
2001 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

Save current value of \glolinkprefix:
2002 \let\@glxtr@org@glolinkprefix\glolinkprefix

Initialise \@glxtr@local@textformat
2003 \let\@glxtr@local@textformat\relax

Initialise thevalue and theHvalue (v1.19).
2004 \def\@glxtr@thevalue{%
2005 \def\@glxtr@theHvalue{\@glxtr@thevalue}%

```



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

2006 \glxtrinitwrgloss

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

2007 \glxtrinithyperoutside

Note that the default link options may override \glxtrinitwrgloss.

2008 \@gls@setdefault@glslink@opts

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

2009 \glxtr@inc@linkcount

As the original definition.

2010 \do@gl:disablehyperinlist

2011 \do@gl@link@checkfirsthyper

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

2012 \glslinkpresetkeys

Set options.

2013 \setkeys{glslink}{#1}%

User hook after options are set:

2014 \glslinkpostsetkeys

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

2015 \ifdefempty{\@glxtr@thevalue}%

2016 {%

2017 \@gls@saveentrycounter

2018 }%

2019 {%

2020 \let\theglentrycounter\@glxtr@thevalue

2021 \def\theHglentrycounter{\@glxtr@theHvalue}%

2022 }%

2023 \@gls@setsort{\glslabel}%

Check if the textformat key has been used.

2024 \ifx\@glxtr@local@textformat\relax

Check textformat attribute (new to v1.21).

2025 \glshasattribute{\glslabel}{textformat}%

2026 {%

2027 \edef\@glxtr@attrval{\glsetattribute{\glslabel}{textformat}}%

2028 \ifcsdef{\@glxtr@attrval}%

2029 {%

2030 \letcs{\@glxtr@textformat}{\@glxtr@attrval}%

2031 }%

2032 {%

2033 \GlossariesExtraWarning{Unknown control sequence name

2034 ‘\@glxtr@attrval’ supplied in textformat attribute

2035 for entry ‘\glslabel’. Reverting to default \string\glstextformat}%

2036 \let\@glxtr@textformat\glstextformat

2037 }%

```

2038     }%
2039     {%
2040         \let\@glsxtr@textformat\glstextformat
2041     }%
2042 \else
2043     \let\@glsxtr@textformat\@glsxtr@local@textformat
2044 \fi

```

Do write if it should occur before the link text:

```

2045 \ifglsxtrinitwrglossbefore
2046 \do@wrglossary{#2}%
2047 \fi

```

Do the link text:

```

2048 \ifKV@glslink@hyper
2049 \ifglsxtr@hyperoutside
2050     \@glslink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
2051 \else
2052     \@glsxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2053 \fi
2054 \else
2055     \ifglsxtr@hyperoutside
2056         \glsdonohyperlink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
2057     \else
2058         \@glsxtr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
2059     \fi
2060 \fi

```

Do write if it should occur after the link text:

```

2061 \ifglsxtrinitwrglossbefore
2062 \else
2063     \do@wrglossary{#2}%
2064 \fi

```

Restore original value of \glolinkprefix:

```

2065 \let\glolinkprefix\@glsxtr@org@glolinkprefix

```

As the original definition:

```

2066 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2067 }

```

```

2068 \define@key{glossadd}{thevalue}{\def\@glsxtr@thevalue{#1}}

```

```

2069 \define@key{glossadd}{theHvalue}{\def\@glsxtr@theHvalue{#1}}

```

lsaddpresetkeys

```

2070 \newcommand*{\glsaddpresetkeys}{ }

```

saddpostsetkeys

```

2071 \newcommand*{\glsaddpostsetkeys}{ }

```

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

```
2072 \renewrobustcmd*{\glsadd}[2][]{%
2073   \glsxtrifinmark
2074   }{%
2075   {%
2076     \@gls@adjustmode
2077     \@glsxtr@record{#1}{#2}{glossadd}%
2078     \glsdoifexists{#2}%
2079     {%
2080       \let\@glsnumberformat\@glsxtr@defaultnumberformat
2081       \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
2082       \def\@glsxtr@thevalue{%
2083         \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Implement any default settings (before options are set)

```
2084   \glsaddpresetkeys
2085   \setkeys{glossadd}{#1}%
```

Implement any default settings (after options are set)

```
2086   \glsaddpostsetkeys
2087   \ifdefempty{\@glsxtr@thevalue}%
2088   {%
2089     \@gls@saveentrycounter
2090   }%
2091   {%
2092     \let\theglsentrycounter\@glsxtr@thevalue
2093     \def\theHglentrycounter{\@glsxtr@theHvalue}%
2094   }%
```

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

```
2095   \@gls@setsort{#2}%
2096   \@do@wrglossary{#2}%
2097   }%
2098   }%
2099 }
```

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

```
2100 \newrobustcmd{\glsaddeach}[2][]{%
2101   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2102 }
```

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

```
2103 \newcommand*{\@glsxtr@field@linkdefs}{%
2104   \let\glsxtrifwasfirstuse\@secondoftwo
2105   \let\glsifplural\@secondoftwo
2106   \let\glsifcaps\@firstofthree
2107   \let\glsinsert\@empty
2108 }
```

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

assignfieldfont

```

2109 \newcommand*{\glxtrassignfieldfont}[1]{%
2110   \ifglstryexists{#1}%
2111   {%
2112     \ifglshasshort{#1}%
2113     {%
2114       \glsetabbrvfmt{\glscategory{#1}}%
2115       \glrifregular{#1}%
2116       {\let\@gls@field@font\glxtrregularfont}%
2117       {\let\@gls@field@font\@firstofone}%
2118     }%
2119     {%
2120       \glrifnotregular{#1}%
2121       {\let\@gls@field@font\@firstofone}%
2122       {\let\@gls@field@font\glxtrregularfont}%
2123     }%
2124   }%
2125   {%
2126     \let\@gls@field@font\@gobble
2127   }%
2128 }

```

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

```

2129 \def\@glstext@#1#2[#3]{%
2130   \glxtrassignfieldfont{#2}%
2131   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2132 }

```

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

```

2133 \def\@GLStext@#1#2[#3]{%
2134   \glxtrassignfieldfont{#2}%
2135   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2136   {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2137 }

```

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

```

2138 \def\@Glstext@#1#2[#3]{%
2139   \glxtrassignfieldfont{#2}%
2140   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2141   {\@gls@field@font{\GLSaccesstext{#2}#3}}%
2142 }

```

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.

checknohyperfirst

```

2143 \newcommand*{\glxtrchecknohyperfirst}[1]{%
2144   \glrifattribute{#1}{nohyperfirst}{true}{\KVglslink@hyperfalse}{}%
2145 }

```

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

```
2146 \def\@glsfirst@#1#2[#3]{%
2147   \glstrassignfieldfont{#2}%
      Ensure that \glsfirst honours the nohyperfirst attribute.
2148   \@gls@field@link
2149   [\let\glstrifwasfirstuse\@firstoftwo
2150     \glstrchecknohyperfirst{#2}%
2151   ]{#1}{#2}%
2152   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2153 }
```

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

```
2154 \def\@Glsfirst@#1#2[#3]{%
2155   \glstrassignfieldfont{#2}%
      Ensure that \Glsfirst honours the nohyperfirst attribute.
2156   \@gls@field@link
2157   [\let\glstrifwasfirstuse\@firstoftwo
2158     \let\glscapscase\@secondofthree
2159     \glstrchecknohyperfirst{#2}%
2160   ]%
2161   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2162 }
```

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

```
2163 \def\@GLSfirst@#1#2[#3]{%
2164   \glstrassignfieldfont{#2}%
      Ensure that \GLSfirst honours the nohyperfirst attribute.
2165   \@gls@field@link
2166   [\let\glstrifwasfirstuse\@firstoftwo
2167     \let\glscapscase\@thirdofthree
2168     \glstrchecknohyperfirst{#2}%
2169   ]%
2170   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2171 }
```

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

```
2172 \def\@glsplural@#1#2[#3]{%
2173   \glstrassignfieldfont{#2}%
2174   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2175   {\@gls@field@font{\glsaccessplural{#2}#3}}%
2176 }
```

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

```
2177 \def\@Glsplural@#1#2[#3]{%
2178   \glstrassignfieldfont{#2}%
2179   \@gls@field@link
2180   [\let\glsifplural\@firstoftwo
```

```

2181 \let\glscapscase\@secondofthree
2182 ]%
2183 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2184 }

```

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

```

2185 \def\@GLSplural@#1#2[#3]{%
2186 \glstrassignfieldfont{#2}%
2187 \@gls@field@link
2188 [\let\glsifplural\@firstoftwo
2189 \let\glscapscase\@thirdofthree
2190 ]%
2191 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2192 }

```

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

```

2193 \def\@glsfirstplural@#1#2[#3]{%
2194 \glstrassignfieldfont{#2}%
2195 \@gls@field@link
2196 [\let\glstrifwasfirstuse\@firstoftwo
2197 \let\glsifplural\@firstoftwo
2198 \glstrchecknohyperfirst{#2}%
2199 ]%
2200 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2201 }

```

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

```

2202 \def\@Glsfirstplural@#1#2[#3]{%
2203 \glstrassignfieldfont{#2}%
2204 \@gls@field@link
2205 [\let\glstrifwasfirstuse\@firstoftwo
2206 \let\glsifplural\@firstoftwo
2207 \let\glscapscase\@secondofthree
2208 \glstrchecknohyperfirst{#2}%
2209 ]%
2210 {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2211 }

```

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

```

2212 \def\@GLSfirstplural@#1#2[#3]{%
2213 \glstrassignfieldfont{#2}%
2214 \@gls@field@link
2215 [\let\glstrifwasfirstuse\@firstoftwo
2216 \let\glsifplural\@firstoftwo

```

```

2217 \let\glscapscase\@thirdofthree
2218 \glstrchecknohyperfirst{#2}%
2219 ]%
2220 {#1}{#2}%
2221 {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2222 }

```

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

```

2223 \def\@glsname@#1#2[#3]{%
2224 \glstrassignfieldfont{#2}%
2225 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
2226 }

```

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

```

2227 \def\@Glsname@#1#2[#3]{%
2228 \glstrassignfieldfont{#2}%
2229 \@gls@field@link
2230 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2231 {\@gls@field@font{\Glsaccessname{#2}#3}}%
2232 }

```

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

```

2233 \def\@GLSname@#1#2[#3]{%
2234 \glstrassignfieldfont{#2}%
2235 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2236 {#1}{#2}%
2237 {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2238 }

```

`\@glsdesc@`

```

2239 \def\@glsdesc@#1#2[#3]{%
2240 \glstrassignfieldfont{#2}%
2241 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
2242 }

```

`\@Glsdesc@` First letter uppercase version.

```

2243 \def\@Glsdesc@#1#2[#3]{%
2244 \glstrassignfieldfont{#2}%
2245 \@gls@field@link
2246 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2247 {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
2248 }

```

`\@GLSdesc@` All uppercase version.

```

2249 \def\@GLSdesc@#1#2[#3]{%
2250 \glstrassignfieldfont{#2}%
2251 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2252 {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2253 }

```

@glsdescplural@ No case-changing version.

```
2254 \def\@glsdescplural@#1#2[#3]{%
2255   \glstrassignfieldfont{#2}%
2256   \@gls@field@link
2257   [\let\glscapscase\@secondoftwo
2258   \let\glsifplural\@firstoftwo
2259   ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
2260 }
```

@Glsdescplural@ First letter uppercase version.

```
2261 \def\@Glsdescplural@#1#2[#3]{%
2262   \glstrassignfieldfont{#2}%
2263   \@gls@field@link
2264   [\let\glscapscase\@secondoftwo
2265   \let\glsifplural\@firstoftwo
2266   ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2267 }
```

@GLSdescplural@ All uppercase version.

```
2268 \def\@GLSdesc@#1#2[#3]{%
2269   \glstrassignfieldfont{#2}%
2270   \@gls@field@link
2271   [\let\glscapscase\@thirdoftwo
2272   \let\glsifplural\@firstoftwo
2273   ]%
2274   {#1}{#2}%
2275   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2276 }
```

\@glssymbol@

```
2277 \def\@glssymbol@#1#2[#3]{%
2278   \glstrassignfieldfont{#2}%
2279   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2280 }
```

\@Glsymbol@ First letter uppercase version.

```
2281 \def\@Glsymbol@#1#2[#3]{%
2282   \glstrassignfieldfont{#2}%
2283   \@gls@field@link
2284   [\let\glscapscase\@secondoftwo]%
2285   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2286 }
```

\@GLSsymbol@ All uppercase version.

```
2287 \def\@GLSsymbol@#1#2[#3]{%
2288   \glstrassignfieldfont{#2}%
2289   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2290   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2291 }
```



lssymbolplural@ No case-changing version.

```
2292 \def\@glssymbolplural@#1#2[#3]{%
2293   \glstrassignfieldfont{#2}%
2294   \@gls@field@link
2295   [\let\glscapscase\@secondoftwo
2296    \let\glsifplural\@firstoftwo
2297   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2298 }
```

lssymbolplural@ First letter uppercase version.

```
2299 \def\@Glsymbolplural@#1#2[#3]{%
2300   \glstrassignfieldfont{#2}%
2301   \@gls@field@link
2302   [\let\glscapscase\@secondoftwo
2303    \let\glsifplural\@firstoftwo
2304   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2305 }
```

LSsymbolplural@ All uppercase version.

```
2306 \def\@GLSsymbol@#1#2[#3]{%
2307   \glstrassignfieldfont{#2}%
2308   \@gls@field@link
2309   [\let\glscapscase\@thirdoftwo
2310    \let\glsifplural\@firstoftwo
2311   ]%
2312   {#1}{#2}%
2313   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2314 }
```

\@Glsuseri@ First letter uppercase version.

```
2315 \def\@Glsuseri@#1#2[#3]{%
2316   \glstrassignfieldfont{#2}%
2317   \@gls@field@link
2318   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2319   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2320 }
```

\@GLSuseri@ All uppercase version.

```
2321 \def\@GLSuseri@#1#2[#3]{%
2322   \glstrassignfieldfont{#2}%
2323   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2324   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
2325 }
```

\@Glsuserii@ First letter uppercase version.

```
2326 \def\@Glsuserii@#1#2[#3]{%
2327   \glstrassignfieldfont{#2}%
2328   \@gls@field@link
```

```

2329 [\let\glscapscase\@secondoftwo]%
2330 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2331 }

```

\@GLSuserii@ All uppercase version.

```

2332 \def\@GLSuserii@#1#2[#3]{%
2333 \glstrassignfieldfont{#2}%
2334 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2335 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2336 }

```

\@Glsuseriii@ First letter uppercase version.

```

2337 \def\@Glsuseriii@#1#2[#3]{%
2338 \glstrassignfieldfont{#2}%
2339 \@gls@field@link
2340 [\let\glscapscase\@secondoftwo]%
2341 {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
2342 }

```

\@GLSuseriii@ All uppercase version.

```

2343 \def\@GLSuseriii@#1#2[#3]{%
2344 \glstrassignfieldfont{#2}%
2345 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2346 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2347 }

```

\@Glsuseriv@ First letter uppercase version.

```

2348 \def\@Glsuseriv@#1#2[#3]{%
2349 \glstrassignfieldfont{#2}%
2350 \@gls@field@link
2351 [\let\glscapscase\@secondoftwo]%
2352 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
2353 }

```

\@GLSuseriv@ All uppercase version.

```

2354 \def\@GLSuseriv@#1#2[#3]{%
2355 \glstrassignfieldfont{#2}%
2356 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2357 {#1}{#2}%
2358 {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2359 }

```

\@Glsuserv@ First letter uppercase version.

```

2360 \def\@Glsuserv@#1#2[#3]{%
2361 \glstrassignfieldfont{#2}%
2362 \@gls@field@link
2363 [\let\glscapscase\@secondoftwo]%
2364 {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
2365 }

```

\@GLSuserv@ All uppercase version.

```
2366 \def\@GLSuserv@#1#2[#3]{%
2367   \glstrassignfieldfont{#2}%
2368   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2369   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2370 }
```

\@Glsuservi@ First letter uppercase version.

```
2371 \def\@Glsuservi@#1#2[#3]{%
2372   \glstrassignfieldfont{#2}%
2373   \@gls@field@link
2374   [\let\glscapscase\@secondoftwo]%
2375   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2376 }
```

\@GLSuservi@ All uppercase version.

```
2377 \def\@GLSuservi@#1#2[#3]{%
2378   \glstrassignfieldfont{#2}%
2379   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2380   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2381 }
```

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

\@acrshort No case change.

```
2382 \def\@acrshort#1#2[#3]{%
2383   \glsdoifexists{#2}%
2384   {%
2385     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2386     \let\glxtrifwasfirstuse\@secondoftwo
2387     \let\glsifplural\@secondoftwo
2388     \let\glscapscase\@firstofthree
2389     \let\glsinsert\@empty
2390     \def\glscustomtext{%
2391       \acronymfont{\glsaccesssshort{#2}}#3%
2392     }%
2393     \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
2394   }%
2395   \glspostlinkhook
2396 }
```

\@Acrshort First letter uppercase.

```
2397 \def\@Acrshort#1#2[#3]{%
2398   \glsdoifexists{#2}%
2399   {%
2400     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2401     \let\glxtrifwasfirstuse\@secondoftwo
```

```

2402 \let\glsifplural\@secondoftwo
2403 \let\glscapscase\@secondofthree
2404 \let\glsinsert\@empty
2405 \def\glscustomtext{%
2406 \acronymfont{\Glsaccessshort{#2}}#3%
2407 }%
2408 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2409 }%
2410 \glspostlinkhook
2411 }

```

\@ACRshort All uppercase.

```

2412 \def\@ACRshort#1#2[#3]{%
2413 \glsdoifexists{#2}%
2414 {%
2415 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2416 \let\glstrifwasfirstuse\@secondoftwo
2417 \let\glsifplural\@secondoftwo
2418 \let\glscapscase\@thirdofthree
2419 \let\glsinsert\@empty
2420 \def\glscustomtext{%
2421 \mfirstucMakeUppercase{\acronymfont{\glsaccessshort{#2}}#3}%
2422 }%
2423 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2424 }%
2425 \glspostlinkhook
2426 }

```

\@acrshortpl No case change.

```

2427 \def\@acrshortpl#1#2[#3]{%
2428 \glsdoifexists{#2}%
2429 {%
2430 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2431 \let\glstrifwasfirstuse\@secondoftwo
2432 \let\glsifplural\@firstoftwo
2433 \let\glscapscase\@firstofthree
2434 \let\glsinsert\@empty
2435 \def\glscustomtext{%
2436 \acronymfont{\glsaccessshortpl{#2}}#3%
2437 }%
2438 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2439 }%
2440 \glspostlinkhook
2441 }

```

\@Acrshortpl First letter uppercase.

```

2442 \def\@Acrshortpl#1#2[#3]{%
2443 \glsdoifexists{#2}%
2444 {%

```

```

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

```

\@ACRshortpl All uppercase.

```

2457 \def\@ACRshortpl#1#2[#3]{%
2458   \glsdofexists{#2}%
2459   {%
2460     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2461     \let\glxtrifwasfirstuse\@secondoftwo
2462     \let\gl@sifplural\@firstoftwo
2463     \let\glscapscase\@thirdofthree
2464     \let\gl$insert\@empty
2465     \def\glscustomtext{%
2466       \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
2467     }%
2468     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2469   }%
2470   \glspostlinkhook
2471 }

```

\@acrlong No case change.

```

2472 \def\@acrlong#1#2[#3]{%
2473   \glsdofexists{#2}%
2474   {%
2475     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2476     \let\glxtrifwasfirstuse\@secondoftwo
2477     \let\gl@sifplural\@secondoftwo
2478     \let\glscapscase\@firstofthree
2479     \let\gl$insert\@empty
2480     \def\glscustomtext{%
2481       \acronymfont{\Glsaccesslong{#2}}#3%
2482     }%
2483     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2484   }%
2485   \glspostlinkhook
2486 }

```

\@Acrlong First letter uppercase.

```

2487 \def\@Acrlong#1#2[#3]{%

```

```

2488 \glsdoifexists{#2}%
2489 {%
2490   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2491   \let\glstrifwasfirstuse\@secondoftwo
2492   \let\glsifplural\@secondoftwo
2493   \let\glscapscase\@secondofthree
2494   \let\glsinsert\@empty
2495   \def\glscustomtext{%
2496     \acronymfont{\Glsaccesslong{#2}}#3%
2497   }%
2498   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2499 }%
2500 \glspostlinkhook
2501 }

```

\@ACRlong All uppercase.

```

2502 \def\@ACRlong#1#2[#3]{%
2503   \glsdoifexists{#2}%
2504   {%
2505     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2506     \let\glstrifwasfirstuse\@secondoftwo
2507     \let\glsifplural\@secondoftwo
2508     \let\glscapscase\@thirdofthree
2509     \let\glsinsert\@empty
2510     \def\glscustomtext{%
2511       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
2512     }%
2513     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2514   }%
2515   \glspostlinkhook
2516 }

```

\@acrlongpl No case change.

```

2517 \def\@acrlongpl#1#2[#3]{%
2518   \glsdoifexists{#2}%
2519   {%
2520     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2521     \let\glstrifwasfirstuse\@secondoftwo
2522     \let\glsifplural\@firstoftwo
2523     \let\glscapscase\@firstofthree
2524     \let\glsinsert\@empty
2525     \def\glscustomtext{%
2526       \acronymfont{\Glsaccesslongpl{#2}}#3%
2527     }%
2528     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2529   }%
2530   \glspostlinkhook
2531 }

```

\@Acrlongpl First letter uppercase.

```
2532 \def\@Acrlongpl#1#2[#3]{%
2533   \glsdoifexists{#2}%
2534   {%
2535     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2536     \let\glxtrifwasfirstuse\@secondoftwo
2537     \let\gl@sifplural\@firstoftwo
2538     \let\glscapscase\@secondofthree
2539     \let\gl$insert\@empty
2540     \def\glscustomtext{%
2541       \acronymfont{\Glsaccesslongpl{#2}}#3%
2542     }%
2543     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2544   }%
2545   \glspostlinkhook
2546 }
```

\@ACRlongpl All uppercase.

```
2547 \def\@ACRlongpl#1#2[#3]{%
2548   \glsdoifexists{#2}%
2549   {%
2550     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2551     \let\glxtrifwasfirstuse\@secondoftwo
2552     \let\gl@sifplural\@firstoftwo
2553     \let\glscapscase\@thirdofthree
2554     \let\gl$insert\@empty
2555     \def\glscustomtext{%
2556       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
2557     }%
2558     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2559   }%
2560   \glspostlinkhook
2561 }
```

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

\@glsaddkey

```
2562 \renewcommand*{\@glsaddkey}[7]{%
2563   \key@ifundefined{glossentry}{#1}%
2564   {%
2565     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2566     \appto\@gl@keymap{, {#1}{#1}}%
2567     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2568     \appto\@newglossaryentryposthook{%
2569       \letcs{\@glo@tmp}{@glo@#1}%
2570       \gl$assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2571     }%
2572     \newcommand*{#3}[1]{\@gl$entry@field{##1}{#1}}%
```

```
2573 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2574 \ifcsdef{@gls@user@#1@}%
2575 {%
2576   \PackageError{glossaries}%
2577   {Can't define '\string#5' as helper command
2578   '\expandafter\string\csname @gls@user@#1@\endcsname' already
2579   exists}%
2580   }%
2581 }%
2582 {%
2583   \expandafter\newcommand\expandafter*\expandafter
2584   {\csname @gls@user@#1\endcsname}[2][ ]{%
2585     \new@ifnextchar[%
2586       {\csuse{@gls@user@#1@}{##1}{##2}}%
2587       {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2588   \csdef{@gls@user@#1@}##1##2[##3]{%
2589     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2590   }%
2591   \newrobustcmd*{#5}{%
2592     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2593   }%
```

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

```
2594 \ifcsdef{@Gls@user@#1@}%
2595 {%
2596   \PackageError{glossaries}%
2597   {Can't define '\string#6' as helper command
2598   '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2599   exists}%
2600   }%
2601 }%
2602 {%
2603   \expandafter\newcommand\expandafter*\expandafter
2604   {\csname @Gls@user@#1\endcsname}[2][ ]{%
2605     \new@ifnextchar[%
2606       {\csuse{@Gls@user@#1@}{##1}{##2}}%
2607       {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
2608   \csdef{@Gls@user@#1@}##1##2[##3]{%
2609     \@gls@field@link[\let\glscaps@case\@secondofthree]%
2610     {##1}{##2}{#4{##2}##3}%
2611   }%
2612   \newrobustcmd*{#6}{%
2613     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2614   }%
```

Finally the all caps version (modified):

```
2615 \ifcsdef{@GLS@user@#1@}%
2616 {%
2617   \PackageError{glossaries}%
```



```

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

```

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

```
2641 \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.

```

2642 \let\@gls@xtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2643 \renewcommand*{\@gls@link@checkfirsthyper}{%

```

\ifglsused isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since \@gls@link@checkfirsthyper is only used by commands like \gls but not by other commands, this seems the best place to put it.

```

2644   \ifglsused{\glslabel}%
2645   {\let\gls@xtrifwasfirstuse\@secondoftwo}
2646   {\let\gls@xtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2647   \edef\gls@categorylabel{\gls@category{\glslabel}}%
2648   \ifglsused{\glslabel}%
2649   {%
2650     \glsifcategoryattribute{\gls@categorylabel}{nohypernext}{true}%
2651     {\KV@glslink@hyperfalse}{}%
2652   }%
2653   {%
2654     \glsifcategoryattribute{\gls@categorylabel}{nohyperfirst}{true}%

```

```

2655     {\KV@glslink@hyperfalse}{}%
2656 }%
2657 \glslinkcheckfirsthyperhook
2658 }

```

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.

```

2659 \ifdef\do@gl:disablehyperinlist
2660 {%
2661   \let\@gl:extr@do@gl:disablehyperinlist\do@gl:disablehyperinlist
2662   \renewcommand*{\do@gl:disablehyperinlist}{%
2663     \@gl:extr@do@gl:disablehyperinlist
2664     \gl:sifattribute{\gl:label}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
2665   }
2666 }
2667 {}

```

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

```

2668 \define@boolkey{gl:link}{noindex}[true]{}
2669 \KV@glslink@noindexfalse

```

If \@gl:s@setdefault@gl:link@opts has been defined (glossaries v4.20) use it to set the default keys in \@gl:link.

lt@gl:link@opts

```

2670 \ifdef\@gl:s@setdefault@gl:link@opts
2671 {
2672   \renewcommand*{\@gl:s@setdefault@gl:link@opts}{%
2673     \KV@glslink@noindexfalse
2674     \@gl:extr:setaliasnoindex
2675   }
2676 }
2677 {

```

Not defined so prepend it to \do@gl:disablehyperinlist to achieve the same effect.

```

2678   \newcommand*{\@gl:s@setdefault@gl:link@opts}{%
2679     \KV@glslink@noindexfalse
2680     \@gl:extr:setaliasnoindex
2681   }
2682   \pretol\do@gl:disablehyperinlist{\@gl:s@setdefault@gl:link@opts}
2683 }

```

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.)

```

2684 \providecommand*{\gl:extr:setaliasnoindex}{%
2685   \KV@glslink@noindextrue
2686 }

```

setaliasnoindex

```
2687 \newcommand*{\@glxtrsetaliasnoindex}{%
2688   \glxtrifhasfield{alias}{\glslabel}%
2689   {%
2690     \let\glxtrindexaliased\@glxtrindexaliased
2691     \glxtrsetaliasnoindex
2692     \let\glxtrindexaliased\@no@glxtrindexaliased
2693   }%
2694   }%
2695 }
```

xtrindexaliased

```
2696 \newcommand{\@glxtrindexaliased}{%
2697   \ifKV@glslink@noindex
2698   \else
2699     \begingroup
2700     \let\@glslnumberformat\@glxtr@defaultnumberformat
2701     \edef\@glsl@counter{\csname glo@glsl@detoklabel{\glslabel}@counter\endcsname}%
2702     \glxtr@saveentrycounter
2703     \@@do@wrglossary{\glxtralias{\glslabel}}%
2704     \endgroup
2705   \fi
2706 }
```

xtrindexaliased

```
2707 \newcommand{\@no@glxtrindexaliased}{%
2708   \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2709     not permitted outside definition of \string\glxtrsetaliasnoindex}%
2710   {}%
2711 }
```

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

```
2712 \let\glxtrindexaliased\@no@glxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
2713 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2714   \renewcommand*{\@glsl@setdefault@glslink@opts}{%
2715     \setkeys{glslink}{#1}%
2716     \@glxtrsetaliasnoindex
2717   }%
2718 }
```

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

```
2719 \newcommand*{\glxtrifindexing}[2]{%
2720   \ifKV@glslink@noindex #2\else #1\fi
2721 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

2722 \renewcommand*{\glswriteentry}[2]{%
2723   \glxtrifindexing
2724   {%
2725     \ifglindexonlyfirst
2726       \ifglused{#1}
2727       {\glxtrdoautoindexname{#1}{dualindex}}%
2728       {#2}%
2729     \else
2730       \glusifattribute{#1}{indexonlyfirst}{true}%
2731       {\ifglused{#1}
2732        {\glxtrdoautoindexname{#1}{dualindex}}%
2733        {#2}}%
2734       {#2}%
2735     \fi
2736   }%
2737   {}%
2738 }

```

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

```

2739 \appto@@do@@wrglossary{\@glxtr@do@@wrindex
2740   \glxtrdowrglossaryhook{\@gls@label}%
2741 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2742 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2743   \glxtrdowrglossaryhook{\@gls@label}%
2744 }

```

`xtr@do@@wrindex`

```

2745 \newcommand*{\@glxtr@do@@wrindex}{%
2746   \glxtrdoautoindexname{\@gls@label}{dualindex}%
2747 }

```

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

```

2748 \newcommand*{\glxtrdowrglossaryhook}[1]{%

```

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

```

2749 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2750   \let\glslinkvar\@firstofthree
2751   \let\@gls@hyp@opt@cs#1\relax
2752   \@ifstar{\s@gls@hyp@opt}%
2753   {\@ifnextchar+%

```

```

2754 {\@firstoftwo{\p@glshyp@opt}}%
2755 {%
2756 \expandafter\@ifnextchar\@glshalt@hyp@opt@char
2757 {\@firstoftwo{\@alt@glshyp@opt}}%
2758 {#1}%
2759 }%
2760 }%
2761 }

```

`\alt@glshyp@opt` User version

```

2762 \newcommand*{\@alt@glshyp@opt}[1][]{%
2763 \let\glslinkvar\@firstofthree
2764 \expandafter\@glshyp@opt@cs\expandafter[\@glshalt@hyp@opt@keys,#1]}

```

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

```

2765 \newcommand*{\@glshalt@hyp@opt@char}{}

```

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

```

2766 \newcommand*{\@glshalt@hyp@opt@keys}{}

```

`\rSetAltModifier`

```

2767 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2768 \let\@glshyp@opt\@glshalt@hyp@opt
2769 \def\@glshalt@hyp@opt@char{#1}%
2770 \def\@glshalt@hyp@opt@keys{#2}%
2771 }

```

`\org@dohyperlink`

```

2772 \let\glsxtr@org@dohyperlink\glsdohyperlink

```

`\glshnavhyperlink` Now that `\glsdohyperlink` (used by `\@glslink`) references `\glslabel` it's necessary to patch `\glshnavhyperlink` 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.

```

2773 \ifdef\glshnavhyperlink
2774 {
2775 \renewcommand*{\glshnavhyperlink}[3][\@glo@type]{%
2776 \edef\glsgprlabel{#2}\protected@edef\glsgprptitle{#3}%

```

Scope:

```

2777 {%
2778 \let\glsdohyperlink\glsxtr@org@dohyperlink
2779 \@glslink{\glshnavhyperlinkname{#1}{#2}}{#3}%
2780 }%
2781 }%
2782 }
2783 {}

```

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

```

2784 \renewcommand*{\glsdohyperlink}[2]{%
2785   \glsattribute{\glslabel}{targeturl}%
2786   {%
2787     \glsattribute{\glslabel}{targetname}%
2788     {%
2789       \glsattribute{\glslabel}{targetcategory}%
2790       {%
2791         \hyperref{\glsattribute{\glslabel}{targeturl}}{%
2792           \glsattribute{\glslabel}{targetcategory}}%
2793           \glsattribute{\glslabel}{targetname}}%
2794         {\glxstrprotectlinks#2}}%
2795       }%
2796     {%
2797       \hyperref{\glsattribute{\glslabel}{targeturl}}{%
2798         }%
2799         \glsattribute{\glslabel}{targetname}}%
2800       {\glxstrprotectlinks#2}}%
2801     }%
2802   }%
2803   {%
2804     \href{\glsattribute{\glslabel}{targeturl}}{%
2805       {\glxstrprotectlinks#2}}%
2806     }%
2807   }%
2808   {%

```

Check for alias.

```

2809   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2810   \ifvoid\gloaliaslabel
2811   {%
2812     \glxtrhyperlink{#1}{\glxstrprotectlinks#2}%
2813   }%
2814   {%

```

Redirect link to the alias target.

```

2815     \glxtrhyperlink
2816     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2817     {\glxstrprotectlinks#2}}%
2818   }%
2819 }%
2820 }

```

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

```
2821 \ifdef\glsshowtarget
2822 {
2823   \newcommand{\glstrhyperlink}[2]{%
2824     \@glsshowtarget{#1}%
2825     \hyperlink{#1}{#2}%
2826   }%
2827 }
2828 {
2829   \newcommand{\glstrhyperlink}[2]{\hyperlink{#1}{#2}}%
2830 }
```

`glisablehyper` 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.

```
2831 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
2832   \glsdoifexists{#2}%
2833   {%
2834     \def\@glo@label{#2}%
2835     {\edef\glslabel{#2}%
2836       \@glslink{\glslinkprefix\glslabel}{#1}}%
2837   }%
2838 }
```

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

```
2839 \renewcommand{\glisablehyper}{%
2840   \KV@glslink@hyperfalse
2841   \def\@glslink{\glsdohyperlink}%
2842   \let\@glstarget\@secondoftwo
2843 }
```

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

```
2844 \renewcommand{\glsenablehyper}{%
2845   \KV@glslink@hypertrue
2846   \def\@glslink{\glsdohyperlink}%
2847   \def\@glstarget{\glsdohypertarget}%
2848 }
```

`glsonohyperlink` 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).

```
2849 \def\glsdohyperlink#1#2{{\glstrprotectlinks #2}}
```

\@glslink Reset \@glslink with patched versions:

```
2850 \ifcsundef{hyperlink}%  
2851 {%  
2852   \def\@glslink{\glsdonohyperlink}  
2853 }%  
2854 {%  
2855   \def\@glslink{\glsdohyperlink}  
2856 }
```

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

```
2857 \newcommand*{\glsxtrprotectlinks}{%  
2858   \KV@glslink@hyperfalse  
2859   \KV@glslink@noindextrue  
2860   \let\@gls@\@glsxtr@p@text@  
2861   \let\@Gls@\@Glsxtr@p@text@  
2862   \let\@GLS@\@GLSxtr@p@text@  
2863   \let\@glspl@\@glsxtr@p@plural@  
2864   \let\@Glspl@\@Glsxtr@p@plural@  
2865   \let\@GLSpl@\@GLSxtr@p@plural@  
2866   \let\@glsxtrshort\@glsxtr@p@short@  
2867   \let\@Glsxtrshort\@Glsxtr@p@short@  
2868   \let\@GLSxtrshort\@GLSxtr@p@short@  
2869   \let\@glsxtrlong\@glsxtr@p@long@  
2870   \let\@Glsxtrlong\@Glsxtr@p@long@  
2871   \let\@GLSxtrlong\@GLSxtr@p@long@  
2872   \let\@glsxtrshortpl\@glsxtr@p@shortpl@  
2873   \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@  
2874   \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@  
2875   \let\@glsxtrlongpl\@glsxtr@p@longpl@  
2876   \let\@Glsxtrlongpl\@Glsxtr@p@longpl@  
2877   \let\@GLSxtrlongpl\@GLSxtr@p@longpl@  
2878   \let\@acrshort\@glsxtr@p@acrshort@  
2879   \let\@Acrshort\@Glsxtr@p@acrshort@  
2880   \let\@ACRshort\@GLSxtr@p@acrshort@  
2881   \let\@acrshortpl\@glsxtr@p@acrshortpl@  
2882   \let\@Acrshortpl\@Glsxtr@p@acrshortpl@  
2883   \let\@ACRshortpl\@GLSxtr@p@acrshortpl@  
2884   \let\@acrlong\@glsxtr@p@acrlong@  
2885   \let\@Acrlong\@Glsxtr@p@acrlong@  
2886   \let\@ACRlong\@GLSxtr@p@acrlong@  
2887   \let\@acrlongpl\@glsxtr@p@acrlongpl@  
2888   \let\@Acrlongpl\@Glsxtr@p@acrlongpl@  
2889   \let\@ACRlongpl\@GLSxtr@p@acrlongpl@  
2890 }
```

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

@glsxtr@p@text@



```

2891 \def\@glxstr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}}
@Glsxtr@p@text@
2892 \def\@Glsxtr@p@text@#1#2[#3]{\@GLstext@{#1}{#2}[#3]}}
@GLSxtr@p@text@
2893 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}}
lsxtr@p@plural@
2894 \def\@glxstr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}}
lsxtr@p@plural@
2895 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}}
LSxtr@p@plural@
2896 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}}
glsxtr@p@short@
2897 \def\@glxstr@p@short@#1#2[#3]{%
2898   {%
2899     \glssetabbrvfmt{\glscategory{#2}}%
2900     \glsabbrvfont{\glsentryshort{#2}}#3%
2901   }%
2902 }
Glsxtr@p@short@
2903 \def\@Glsxtr@p@short@#1#2[#3]{%
2904   {%
2905     \glssetabbrvfmt{\glscategory{#2}}%
2906     \glsabbrvfont{\Glsentryshort{#2}}#3%
2907   }%
2908 }
GLSxtr@p@short@
2909 \def\@GLSxtr@p@short@#1#2[#3]{%
2910   {%
2911     \glssetabbrvfmt{\glscategory{#2}}%
2912     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
2913   }%
2914 }
sxtr@p@shortpl@
2915 \def\@glxstr@p@shortpl@#1#2[#3]{%
2916   {%
2917     \glssetabbrvfmt{\glscategory{#2}}%
2918     \glsabbrvfont{\glsentryshortpl{#2}}#3%
2919   }%
2920 }

```

sxtr@p@shortpl@

```
2921 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2922 {%
2923   \glsetabbrvfmt{\glscategory{#2}}%
2924   \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2925 }%
2926 }
```

Sxtr@p@shortpl@

```
2927 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2928 {%
2929   \glsetabbrvfmt{\glscategory{#2}}%
2930   \mfirstucMakeUppercase{\glsabbrvfont{\Glsentryshortpl{#2}}#3}%
2931 }%
2932 }
```

@glxtr@p@long@

```
2933 \def\@glxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}}#3}}
```

@Glsxtr@p@long@

```
2934 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}}#3}}
```

@GLSxtr@p@long@

```
2935 \def\@GLSxtr@p@long@#1#2[#3]{%
2936   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
2937 \def\@glxtr@p@longpl@#1#2[#3]{\{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
2938 \def\@Glsxtr@p@longpl@#1#2[#3]{\{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
2939 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2940   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylongpl{#2}}#3}}}
```

xtr@p@acrshort@

```
2941 \def\@glxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
2942 \def\@Glsxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
2943 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2944   {\mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#2}}#3}}}
```

r@p@acrshortpl@

```
2945 \def\@glxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\Glsentryshortpl{#2}}#3}}
```

```

r@p@acrshortpl@
2946 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
2947 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2948   {\mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
2949 \def\@glxtr@p@acrlong@#1#2[#3]{\glentrylong{#2}#3}}

sxtr@p@acrlong@
2950 \def\@GLSxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
2951 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2952   {\mfirstucMakeUppercase{\glentrylong{#2}#3}}}

tr@p@acrlongpl@
2953 \def\@glxtr@p@acrlongpl@#1#2[#3]{\glentrylongpl{#2}#3}}

tr@p@acrlongpl@
2954 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
2955 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2956   {\mfirstucMakeUppercase{\glentrylongpl{#2}#3}}}

```

Commands to minimise conflict.

```

\@glxtrp@opt
2957 \newcommand*{\@glxtrp@opt}{hyper=false,noindex}

\glxtrsetpopts   Used in glossary to switch hyperlinks on for the \glxtrp type of commands.
2958 \newcommand*{\glxtrsetpopts}[1]{%
2959   \renewcommand*{\@glxtrp@opt}{#1}%
2960 }

lossxtrsetpopts  Used in glossary to switch hyperlinks on for the \glxtrp type of commands.
2961 \newcommand*{\glossxtrsetpopts}{%
2962   \glxtrsetpopts{noindex}%
2963 }

\@@glxtrp
2964 \newrobustcmd*{\@@glxtrp}[2]{%

```

Add scope.

```

2965  {%
2966    \let\glspostlinkhook\relax
2967    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
2968  }%
2969 }

```

\@glsxtrp

```

2970 \newrobustcmd*{\@glsxtrp}[2]{%
2971   \ifcsdef{gls#1}%
2972   {%
2973     \@glsxtrp{gls#1}{#2}%
2974   }%
2975   {%
2976     \ifcsdef{glsxtr#1}%
2977     {%
2978       \@glsxtrp{glsxtr#1}{#2}%
2979     }%
2980     {%
2981       \PackageError{glossaries-extra}{‘#1’ not recognised by
2982         \string\glsxtrp}{}%
2983     }%
2984   }%
2985 }

```

\@Glsxtrp

```

2986 \newrobustcmd*{\@Glsxtrp}[2]{%
2987   \ifcsdef{Gls#1}%
2988   {%
2989     \@glsxtrp{Gls#1}{#2}%
2990   }%
2991   {%
2992     \ifcsdef{Glsxtr#1}%
2993     {%
2994       \@glsxtrp{Glsxtr#1}{#2}%
2995     }%
2996     {%
2997       \PackageError{glossaries-extra}{‘#1’ not recognised by
2998         \string\Glsxtrp}{}%
2999     }%
3000   }%
3001 }

```

\@GLSxtrp

```

3002 \newrobustcmd*{\@GLSxtrp}[2]{%
3003   \ifcsdef{GLS#1}%
3004   {%
3005     \@glsxtrp{GLS#1}{#2}%
3006   }%

```

```

3007 {%
3008   \ifcsdef{GLSxtr#1}%
3009   {%
3010     \@@glsxtrp{GLSxtr#1}{#2}%
3011   }%
3012   {%
3013     \PackageError{glossaries-extra}{‘#1’ not recognised by
3014       \string\GLSxtrp}{}%
3015   }%
3016 }%
3017 }

```

\glsxtr@entry@p

```

3018 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3019   \glsifattribute{#1}{headuc}{true}%
3020   {%
3021     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3022   }%
3023   {%
3024     \@gls@entry@field{#1}{#2}%
3025   }%
3026 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3027 \ifdef\texorpdfstring
3028 {
3029   \newcommand{\glsxtrp}[2]{%
3030     \protect\NoCaseChange
3031     {%
3032       \protect\texorpdfstring
3033       {%
3034         \protect\glsxtrifinmark
3035         {%
3036           \ifcsdef{glsxtrhead#1}%
3037           {%
3038             {\protect\csuse{glsxtrhead#1}{#2}}%
3039           }%
3040           {%
3041             \glsxtr@headentry@p{#2}{#1}%
3042           }%
3043         }%
3044         {%
3045           \@glsxtrp{#1}{#2}%
3046         }%
3047       }%
3048     }%
3049     \protect\@gls@entry@field{#2}{#1}%
3050   }%
3051 }%

```

```

3052 }
3053 }
3054 {
3055   \newcommand{\glxtrp}[2]{%
3056     \protect\NoCaseChange
3057     {%
3058       \protect\glxtrifinmark
3059       {%
3060         \ifcsdef{glxtrhead#1}%
3061         {%
3062           {\protect\csuse{glxtrhead#1}}%
3063         }%
3064         {%
3065           \glxtr@headentry@p{#2}{#1}%
3066         }%
3067       }%
3068     }%
3069     \@glxtrp{#1}{#2}%
3070   }%
3071 }%
3072 }
3073 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3074 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

3075 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

3076 \ifdef\textorpdfstring
3077 {
3078   \newcommand{\Glsxtrp}[2]{%
3079     \protect\NoCaseChange
3080     {%
3081       \protect\textorpdfstring
3082       {%
3083         \protect\glxtrifinmark
3084         {%
3085           \ifcsdef{Glsxtrhead#1}%
3086           {%
3087             {\protect\csuse{Glsxtrhead#1}{#2}}%
3088           }%
3089           {%
3090             \protect\@Gls@entry@field{#2}{#1}%
3091           }%

```

```

3092     }%
3093     {%
3094         \@Glsxtrp{#1}{#2}%
3095     }%
3096 }%
3097 {%
3098     \protect\@gls@entry@field{#2}{#1}%
3099 }%
3100 }%
3101 }
3102 }
3103 {
3104     \newcommand{\Glsxtrp}[2]{%
3105         \protect\NoCaseChange
3106         {%
3107             \protect\glsxtrifinmark
3108             {%
3109                 \ifcsdef{Glsxtrhead#1}%
3110                 {%
3111                     {\protect\csuse{Glsxtrhead#1}}%
3112                 }%
3113                 {%
3114                     \protect\@Gls@entry@field{#2}{#1}%
3115                 }%
3116             }%
3117             {%
3118                 \@Glsxtrp{#1}{#2}%
3119             }%
3120         }%
3121     }
3122 }

```

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

```

3123 \ifdef\texorpdfstring
3124 {
3125     \newcommand{\GLSxtrp}[2]{%
3126         \protect\NoCaseChange
3127         {%
3128             \protect\texorpdfstring
3129             {%
3130                 \protect\glsxtrifinmark
3131                 {%
3132                     \ifcsdef{GLSxtr#1}%
3133                     {%
3134                         {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3135                     }%
3136                     {%
3137                         \protect\mfirstucMakeUppercase
3138                     }%

```

```

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

```

### 1.3.5 Entry Counting

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

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be



temporarily disabled, `\@glsunset` is let to `\@glsxtr@unset`, which performs the actual unsetting through `\@@glsunset` and then does the hook. This means that the unsetting (and the hook) can be switched off by redefining `\@glsunset` and then switched back on again by changing the definition back to `\@glsxtr@unset`.

`\@glsxtr@unset` Global unset.

```
3176 \newcommand*{\@glsxtr@unset}[1]{%
3177   \@@glsunset{#1}%
3178   \glsxtrpostunset{#1}%
3179 }
```

`\@glsunset` Global unset.

```
3180 \let\@glsunset\@glsxtr@unset
```

`glsxtrpostunset`

```
3181 \newcommand*{\glsxtrpostunset}[1]{}
```

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

`tUnsetBuffering`

```
3182 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3183   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3184 }
```

`tUnsetBuffering` Unstarred version doesn't check for duplicates.

```
3185 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3186   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3187   \def\@glsxtr@unset@buffer{}%
3188   \let\@glsunset\@glsxtrbuffer@unset
3189 }
```

`tUnsetBuffering` Starred version checks for duplicates.

```
3190 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3191   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3192   \def\@glsxtr@unset@buffer{}%
3193   \let\@glsunset\@glsxtrbuffer@nodup@unset
3194 }
```

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

```
3195 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3196   \listxadd\@glsxtr@unset@buffer{#1}%
3197 }
```

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

```

3198 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3199   \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}}%
3200   {\listxadd\@glsxtr@unset@buffer{#1}}}%
3201 }

```

pUnsetBuffering

```

3202 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3203   \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3204 }

```

pUnsetBuffering Unstarred form (global unset).

```

3205 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3206   \let\@glsunset\@glsxtr@unset
3207   \forlistloop\@glsunset\@glsxtr@unset@buffer
3208   \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3209 }

```

pUnsetBuffering Starred form (local unset).

```

3210 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3211   \forlistloop\@glslocalunset\@glsxtr@unset@buffer
3212   \let\@glsunset\@glsxtr@unset
3213 }

```

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

```

3214 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3215   \forlistloop#1\@glsxtr@unset@buffer
3216 }

```

\@glslocalunset Local unset.

```

3217 \renewcommand*{\@glslocalunset}[1]{%
3218   \@glslocalunset{#1}%
3219   \glsxtrpostlocalunset{#1}%
3220 }%

```

rpostlocalunset

```

3221 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

\@glsreset Global reset.

```

3222 \renewcommand*{\@glsreset}[1]{%
3223   \@glsreset{#1}%
3224   \glsxtrpostreset{#1}%
3225 }%

```

glsxtrpostreset

```

3226 \newcommand*{\glsxtrpostreset}[1]{%

```

`\@glslocalreset` Local reset.

```
3227 \renewcommand*{\@glslocalreset}[1]{%
3228   \@glslocalreset{#1}%
3229   \glstrpostlocalreset{#1}%
3230 }%
```

`rpostlocalreset`

```
3231 \newcommand*{\glstrpostlocalreset}[1]{}
```

`slocalreseteach` Locally reset a list of entries.

```
3232 \newcommand*{\glslocalreseteach}[1]{%
3233   \gls@ifnotmeasuring
3234   {%
3235     \@for\@gls@thislabel:=#1\do{%
3236       \glsdoifexists{\@gls@thislabel}%
3237       {%
3238         \@glslocalreset{\@gls@thislabel}%
3239       }%
3240     }%
3241   }%
3242 }
```

`slocalunseteach` Locally unset a list of entries.

```
3243 \newcommand*{\glslocalunseteach}[1]{%
3244   \gls@ifnotmeasuring
3245   {%
3246     \@for\@gls@thislabel:=#1\do{%
3247       \glsdoifexists{\@gls@thislabel}%
3248       {%
3249         \@glslocalunset{\@gls@thislabel}%
3250       }%
3251     }%
3252   }%
3253 }
```

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

```
3254 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
3255   \glsenableentrycount
```

Redefine `\gls` etc:

```
3256   \renewcommand*{\gls}{\cgl}%
3257   \renewcommand*{\Gls}{\cGls}%
3258   \renewcommand*{\glspl}{\cglspl}%
3259   \renewcommand*{\Glspl}{\cGlspl}%
3260   \renewcommand*{\GLS}{\cGLS}%
3261   \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
3262 \@glstr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
3263 \let\GlsXtrEnableEntryCounting\@glstr@setentrycountunsetattr
3264 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3265 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3266 can't be used with \string\GlsXtrEnableEntryCounting}%
3267 {Use one or other but not both commands}}%
3268 }
```

ycountunsetattr

```
3269 \newcommand*{\@glstr@setentrycountunsetattr}[2]{%
3270 \@for\@glstr@cat:=#1\do
3271 {%
3272 \ifdefempty{\@glstr@cat}{}%
3273 {%
3274 \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
3275 }%
3276 }%
3277 }
```

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

nableentrycount

```
3278 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
3279 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
3280 \renewcommand*{\gls@defdocnewglossaryentry}{%
3281 \renewcommand*{\newglossaryentry}[2]{%
3282 \PackageError{glossaries}{\string\newglossaryentry\space
3283 may only be used in the preamble when entry counting has
3284 been activated}{If you use \string\glsenableentrycount\space
3285 you must place all entry definitions in the preamble not in
3286 the document environment}%
3287 }%
3288 }%
```

New commands to access new fields:

```
3289 \newcommand*{\glsentrycurrcount}[1]{%
3290 \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
3291 {0}{\@gls@entry@field{##1}{currcount}}%
3292 }%
3293 \newcommand*{\glsentryprevcount}[1]{%
3294 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
3295 {0}{\@gls@entry@field{##1}{prevcount}}%
3296 }%
```

Adjust post unset and reset:

```

3297 \let\@glxtr@entrycount@org@unset\glxtrpostunset
3298 \renewcommand*{\glxtrpostunset}[1]{%
3299   \@glxtr@entrycount@org@unset{##1}%
3300   \@gls@increment@currcount{##1}%
3301 }%
3302 \let\@glxtr@entrycount@org@localunset\glxtrpostlocalunset
3303 \renewcommand*{\glxtrpostlocalunset}[1]{%
3304   \@glxtr@entrycount@org@localunset{##1}%
3305   \@gls@local@increment@currcount{##1}%
3306 }%
3307 \let\@glxtr@entrycount@org@reset\glxtrpostreset
3308 \renewcommand*{\glxtrpostreset}[1]{%
3309   \@glxtr@entrycount@org@reset{##1}%
3310   \csgdef{glo\@glsdetoklabel{##1}@currcount}{0}%
3311 }%
3312 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
3313 \renewcommand*{\glxtrpostlocalreset}[1]{%
3314   \@glxtr@entrycount@org@localreset{##1}%
3315   \csdef{glo\@glsdetoklabel{##1}@currcount}{0}%
3316 }%

```

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

```

3317 \let\@cgl@@cgl@
3318 \let\@cgl@pl@@cgl@pl@

3319 \let\@cGls@@cGls@
3320 \let\@cGlspl@@cGlspl@
3321 \let\@cGLS@@cGLS@
3322 \let\@cGLSpl@@cGLSpl@

```

The rest is as the original definition.

```

3323 \AtEndDocument{\@gls@write@entrycounts}%
3324 \renewcommand*{\@gls@entry@count}[2]{%
3325   \csgdef{glo\@glsdetoklabel{##1}@prevcount}{##2}%
3326 }%
3327 \let\glsenableentrycount\relax
3328 \renewcommand*{\glsenableentryunitcount}{%
3329   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3330     can't be used with \string\glsenableentrycount}%
3331   {Use one or other but not both commands}%
3332 }%
3333 }

```

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

```

3334 \renewcommand*{\@gls@write@entrycounts}{%
3335   \immediate\write\@auxout
3336     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%

```

```

3337 \count@=0\relax
3338 \forallglentries{\@glentry}{%
3339   \glshasattribute{\@glentry}{entrycount}%
3340   {%
3341     \ifglused{\@glentry}%
3342     {%
3343       \immediate\write\@auxout
3344       {\string\@gl@entry@count{\@glentry}{\glentrycurrcount{\@glentry}}}%
3345     }%
3346   }%
3347   \advance\count@ by \@ne
3348 }%
3349 }%
3350 }%
3351 \ifnum\count@=0
3352   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3353   \MessageBreak with \string\glsenableentrycount\space but the
3354   \MessageBreak attribute 'entrycount' hasn't
3355   \MessageBreak been assigned to any of the defined
3356   \MessageBreak entries}%
3357 \fi
3358 }

```

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

```

3359 \newcommand*{\glxtrifcounttrigger}[3]{%
3360   \glshasattribute{#1}{entrycount}%
3361   {%
3362     \ifnum\glentryprevcount{#1}>\glsetattribute{#1}{entrycount}\relax
3363     #3%
3364   \else
3365     #2%
3366   \fi
3367 }%
3368 {#3}%
3369 }

```

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

`\@@cgl@`

```

3370 \def\@@cgl@#1#2[#3]{%
3371   \glxtrifcounttrigger{#2}%
3372   {%
3373     \cglformat{#2}{#3}%
3374     \glunset{#2}%
3375   }%

```

```

3376  {%
3377    \@gls@{#1}{#2}[#3]%
3378  }%
3379 }%

```

\@@cglsp1@

```

3380 \def\@@cglsp1@#1#2[#3]{%
3381   \glstrifcounttrigger{#2}%
3382   {%
3383     \cglsp1format{#2}{#3}%
3384     \glunset{#2}%
3385   }%
3386   {%
3387     \@glsp1@{#1}{#2}[#3]%
3388   }%
3389 }%

```

\@@cGls@

```

3390 \def\@@cGls@#1#2[#3]{%
3391   \glstrifcounttrigger{#2}%
3392   {%
3393     \cGlsformat{#2}{#3}%
3394     \glunset{#2}%
3395   }%
3396   {%
3397     \@Gls@{#1}{#2}[#3]%
3398   }%
3399 }%

```

\@@cGlsp1@

```

3400 \def\@@cGlsp1@#1#2[#3]{%
3401   \glstrifcounttrigger{#2}%
3402   {%
3403     \cGlsp1format{#2}{#3}%
3404     \glunset{#2}%
3405   }%
3406   {%
3407     \@Glsp1@{#1}{#2}[#3]%
3408   }%
3409 }%

```

\@@cGLS@

```

3410 \def\@@cGLS@#1#2[#3]{%
3411   \glstrifcounttrigger{#2}%
3412   {%
3413     \cGLSformat{#2}{#3}%
3414     \glunset{#2}%
3415   }%
3416   {%

```

```

3417 \cGLS@{#1}{#2}[#3]%
3418 }%
3419 }%

```

\cGLSpl@

```

3420 \def\cGLSpl@#1#2[#3]{%
3421 \glstrifcounttrigger{#2}%
3422 {%
3423 \cGLSplformat{#2}{#3}%
3424 \glset{#2}%
3425 }%
3426 {%
3427 \cGLSpl@{#1}{#2}[#3]%
3428 }%
3429 }%

```

Remove default warnings from \cglset etc so that it can be used interchangeable with \glset etc.

\cglset@

```

3430 \def\cglset@#1#2[#3]{\cglset@{#1}{#2}[#3]}

```

\cGLset@

```

3431 \def\cGLset@#1#2[#3]{\cGLset@{#1}{#2}[#3]}

```

\cglspl@

```

3432 \def\cglspl@#1#2[#3]{\cglspl@{#1}{#2}[#3]}

```

\cGLspl@

```

3433 \def\cGLspl@#1#2[#3]{\cGLspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

\cGLS

```

3434 \newrobustcmd*\cGLS{\cglsethyp@opt\cGLS}

```

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

```

3435 \newcommand*\cGLS[2][{}]{%
3436 \new@ifnextchar{\cGLS@{#1}{#2}}{\cGLS@{#1}{#2}}[{}]}%
3437 }

```

\cGLS@

```

3438 \def\cGLS@#1#2[#3]{\cGLS@{#1}{#2}[#3]}

```

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

```

3439 \newcommand*\cGLSformat[2]{%
3440 \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsetformat{#1}{#2}}}%
3441 }

```



```
\cGLSp1
3442 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\cGLSp1}
```

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

```
3443 \newcommand*{\cGLSp1}[2][\%
3444 \new@ifnextchar[\cGLSp1@{#1}{#2}]{\cGLSp1@{#1}{#2}[ ]}%
3445 }
```

\cGLSp1@

```
3446 \def\cGLSp1@#1#2[#3]{\cGLSp1@{#1}{#2}[#3]}
```

\cGLSplformat Format used by \cGLSp1 if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
3447 \newcommand*{\cGLSplformat}[2]{\%
3448 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}}%
3449 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3450 \renewcommand*{\cglformat}[2]{\%
3451 \glsifregular{#1}
3452 {\glsentryfirst{#1}}%
3453 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3454 }
```

\cGlsformat

```
3455 \renewcommand*{\cGlsformat}[2]{\%
3456 \glsifregular{#1}
3457 {\Glsentryfirst{#1}}%
3458 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3459 }
```

\cglsplformat

```
3460 \renewcommand*{\cglsplformat}[2]{\%
3461 \glsifregular{#1}
3462 {\glsentryfirstplural{#1}}%
3463 {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3464 }
```

\cGlsplformat

```
3465 \renewcommand*{\cGlsplformat}[2]{\%
3466 \glsifregular{#1}
3467 {\Glsentryfirstplural{#1}}%
3468 {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%#2%
3469 }
```

New code similar to above for unit counting.

defunitcounters

```
3470 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3471   \edef\@glo@countunit{\csuse{@glxstr@categoryattr@@\@glo@category @unitcount}}%
3472   \ifdefvoid\@glo@countunit
3473   {}%
3474   {%
3475     \@glxstr@ifunitcounter{\@glo@countunit}%
3476     {}%
3477     {\expandafter\@glxstr@addunitcounter\expandafter{\@glo@countunit}}%
3478   }%
3479 }
```

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

```
3480 \newcommand*{\@glxstr@unitcountlist}{}
```

@addunitcounter

```
3481 \newcommand*{\@glxstr@addunitcounter}[1]{%
3482   \listadd{\@glxstr@unitcountlist}{#1}%
3483   \ifcsundef{glxstr@theunit@#1}
3484   {%
3485     \ifcsdef{theH#1}%
3486     {\csdef{glxstr@theunit@#1}{\csuse{theH#1}}}%
3487     {\csdef{glxstr@theunit@#1}{\csuse{the#1}}}%
3488   }%
3489   {}%
3490 }
```

r@ifunitcounter

```
3491 \newcommand*{\@glxstr@ifunitcounter}[3]{%
3492   \xifinlist{#1}{\@glxstr@unitcountlist}{#2}{#3}%
3493 }
```

urrentunitcount

```
3494 \newcommand*\@glxstr@currentunitcount[1]{%
3495   glo@\glsdetoklabel{#1}@currunit@glsggetattribute{#1}{unitcount}.%
3496   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3497 }
```

eviousunitcount

```
3498 \newcommand*\@glxstr@previousunitcount[1]{%
3499   glo@\glsdetoklabel{#1}@prevunit@glsggetattribute{#1}{unitcount}.%
3500   \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}%
3501 }
```

t@currunitcount

```
3502 \newcommand*{\@gls@increment@currunitcount}[1]{%
3503   \glshasattribute{#1}{unitcount}%
3504   {%
```

```

3505 \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3506 \ifcsundef{\@glxtr@csname}%
3507 {%
3508 \csgdef{\@glxtr@csname}{1}%
3509 \listcsxadd
3510 {glo@glsdetoklabel{#1}@unitlist}%
3511 {\glsggetattribute{#1}{unitcount}.%
3512 \csuse{glxtr@theunit@glsggetattribute{#1}{unitcount}}}%
3513 }%
3514 }%
3515 {%
3516 \csxdef{\@glxtr@csname}%
3517 {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3518 }%
3519 }%
3520 {}%
3521 }

```

t@currunitcount

```

3522 \newcommand*{\@glx@local@increment@currunitcount}[1]{%
3523 \glshasattribute{#1}{unitcount}%
3524 {%
3525 \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
3526 \ifcsundef{\@glxtr@csname}%
3527 {%
3528 \csdef{\@glxtr@csname}{1}%
3529 \listcseadd
3530 {glo@glsdetoklabel{#1}@unitlist}%
3531 {\glsggetattribute{#1}{unitcount}.%
3532 \csuse{glxtr@theunit@glsggetattribute{#1}{unitcount}}}%
3533 }%
3534 }%
3535 {%
3536 \csedef{\@glxtr@csname}%
3537 {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
3538 }%
3539 }%
3540 {}%
3541 }

```

r@currunitcount

```

3542 \newcommand*{\@glxtr@currunitcount}[2]{%
3543 \ifcsundef
3544 {glo@glsdetoklabel{#1}@currunit@#2}%
3545 {0}%
3546 {\csuse{glo@glsdetoklabel{#1}@currunit@#2}}%
3547 }%

```

r@prevunitcount

```

3548 \newcommand*{\@glxstr@prevunitcount}[2]{%
3549 \ifcsundef
3550 {glo@\glsdetoklabel{##1}@prevunit@#2}%
3551 {0}%
3552 {\csuse{glo@\glsdetoklabel{##1}@prevunit@#2}}%
3553 }%

```

entryunitcount

```

3554 \newcommand*{\glsenableentryunitcount}{%
    Enable new fields:
3555 \appto@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
    Just in case the user has switched on the docdef option.
3556 \renewcommand*{\gl@defdocnewglossaryentry}{%
3557 \renewcommand*\newglossaryentry[2]{%
3558 \PackageError{glossaries}{\string\newglossaryentry\space
3559 may only be used in the preamble when entry counting has
3560 been activated}{If you use \string\glsenableentryunitcount\space
3561 you must place all entry definitions in the preamble not in
3562 the document environment}%
3563 }%
3564 }%

```

New commands to access new fields:

```

3565 \newcommand*{\glentrycurrcount}[1]{%
3566 \@glxstr@currunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3567 \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3568 }%
3569 \newcommand*{\glentryprevcount}[1]{%
3570 \@glxstr@prevunitcount{##1}{\glsggetattribute{##1}{unitcount}}.%
3571 \csuse{glxstr@theunit@\glsggetattribute{##1}{unitcount}}}%
3572 }%

```

Access total count:

```

3573 \newcommand*{\glentryprevtotalcount}[1]{%
3574 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3575 {0}%
3576 {%
3577 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
3578 }%
3579 }%

```

Access max value:

```

3580 \newcommand*{\glentryprevmaxcount}[1]{%
3581 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3582 {0}%
3583 {%
3584 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
3585 }%
3586 }%

```

Adjust post unset and reset:

```

3587 \let\@glxstr@entryunitcount@org@unset\glxstrpostunset
3588 \renewcommand*{\glxstrpostunset}[1]{%
3589   \@glxstr@entryunitcount@org@unset{##1}%
3590   \@glx@increment@currunitcount{##1}%
3591 }%
3592 \let\@glxstr@entryunitcount@org@localunset\glxstrpostlocalunset
3593 \renewcommand*{\glxstrpostlocalunset}[1]{%
3594   \@glxstr@entryunitcount@org@localunset{##1}%
3595   \@glx@local@increment@currunitcount{##1}%
3596 }%
3597 \let\@glxstr@entryunitcount@org@reset\glxstrpostreset
3598 \renewcommand*{\glxstrpostreset}[1]{%
3599   \glshasattribute{##1}{unitcount}%
3600   {%
3601     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3602     \ifcsundef{\@glxstr@csname}%
3603     {%
3604       {\csgdef{\@glxstr@csname}{0}}%
3605     }%
3606   }%
3607 }%
3608 \let\@glxstr@entryunitcount@org@localreset\glxstrpostlocalreset
3609 \renewcommand*{\glxstrpostlocalreset}[1]{%
3610   \@glxstr@entryunitcount@org@localreset{##1}%
3611   \glshasattribute{##1}{unitcount}%
3612   {%
3613     \edef\@glxstr@csname{\@glxstr@currentunitcount{##1}}%
3614     \ifcsundef{\@glxstr@csname}%
3615     {%
3616       {\csdef{\@glxstr@csname}{0}}%
3617     }%
3618   }%
3619 }%

```

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

```

3620 \let\@cgl@\@cgl@
3621 \let\@cgl@pl@\@cgl@pl@

3622 \let\@cGl@\@cGl@
3623 \let\@cGl@pl@\@cGl@pl@
3624 \let\@cGL@\@cGL@
3625 \let\@cGL@pl@\@cGL@pl@

```

Write information to the aux file.

```

3626 \AtEndDocument{\@glx@write@entryunitcounts}%
3627 \renewcommand*{\@glx@entry@unitcount}[3]{%
3628   \csgdef{glo@glx@detoklabel{##1}@prevunit@##3}{##2}%
3629   \ifcsundef{glo@glx@detoklabel{##1}@prevunittotal}%

```

```

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

```

entry@unitcount

```

3651 \newcommand*{\@gls@entry@unitcount}[3]{%

```

ryunitcounts@do

```

3652 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3653   \immediate\write\@auxout
3654   {\string\@gls@entry@unitcount
3655     {\@glsentry}%
3656     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3657     }%
3658     {#1}}%
3659 }

```

entryunitcounts

```

3660 \newcommand*{\@gls@write@entryunitcounts}{%
3661   \immediate\write\@auxout
3662   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3663   \count@=0\relax
3664   \forallglsentries{\@glsentry}{%
3665     \glsattribute{\@glsentry}{unitcount}%
3666     {%
3667       \ifglsused{\@glsentry}%
3668       {%
3669         \forlistcsloop
3670         {\@gls@write@entryunitcounts@do}%
3671         {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3672       }%

```

```

3673     {}%
3674     \advance\count@ by \@ne
3675   }%
3676   {}%
3677 }%
3678 \ifnum\count@=0
3679   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3680     \MessageBreak with \string\glsenableentryunitcount\space but the
3681     \MessageBreak attribute ‘unitcount’ hasn’t
3682     \MessageBreak been assigned to any of the defined
3683     \MessageBreak entries}%
3684 \fi
3685 }

```

`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.

```

3686 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3687 \glsenableentryunitcount

```

Redefine \gls etc:

```

3688 \renewcommand*{\gls}{\cgl{s}}%
3689 \renewcommand*{\Gls}{\cGls}%
3690 \renewcommand*{\glspl}{\cgl{s}pl}%
3691 \renewcommand*{\Glspl}{\cGlspl}%
3692 \renewcommand*{\GLS}{\cGLS}%
3693 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3694 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3695 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
3696 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3697   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3698     can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3699   {Use one or other but not both commands}}%
3700 }

```

`tcountunsetattr`

```

3701 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3702   \@for\@glsxtr@cat:=#1\do
3703   {%
3704     \ifdefempty{\@glsxtr@cat}{}%
3705     {%
3706       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3707       \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3708     }%
3709   }%
3710 }

```

### 1.3.6 Acronym Modifications

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

`nericNewAcronym`

```

3711 \renewcommand*{\SetGenericNewAcronym}{%
3712   \let\@Gls@entryname\@Gls@acrentryname
3713   \renewcommand{\newacronym}[4][]{%
3714     \ifdefempty{\@glsacronymlists}%
3715     {%
3716       \def\@glo@type{\acronymtype}%
3717       \setkeys{glossentry}{##1}%
3718       \DeclareAcronymList{\@glo@type}%
3719     }%
3720   }%
3721   \glskeylisttok{##1}%
3722   \glslabeltok{##2}%
3723   \glsshorttok{##3}%
3724   \glslongtok{##4}%
3725   \newacronymhook
3726   \protected@edef\@do@newglossaryentry{%
3727     \noexpand\newglossaryentry{\the\glslabeltok}%
3728     {%
3729       type=\acronymtype,%
3730       name={\expandonce{\acronymentry{##2}}},%
3731       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3732       text={\the\glsshorttok},%
3733       short={\the\glsshorttok},%
3734       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3735       long={\the\glslongtok},%
3736       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3737       category=acronym,%
3738       \GenericAcronymFields,%
3739       \the\glskeylisttok
3740     }%
3741   }%
3742   \@do@newglossaryentry
3743 }%
3744 \renewcommand*{\acrfullfmt}[3]{%
3745   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}%
3746 \renewcommand*{\Acrfullfmt}[3]{%
3747   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}%
3748 \renewcommand*{\ACRfullfmt}[3]{%
3749   \glslink[##1]{##2}{%

```



```

3750     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3751 \renewcommand*{\acrfullplfmt}[3]{%
3752   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3753 \renewcommand*{\Acrfullplfmt}[3]{%
3754   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3755 \renewcommand*{\ACRfullplfmt}[3]{%
3756   \glslink[##1]{##2}{%
3757     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3758 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
3759 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
3760 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
3761 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
3762 }

```

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:

```

3763 \let\@glsxtr@org@setacronymstyle\setacronymstyle
3764 \let\@glsxtr@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`.

```

3765 \newcommand*{\MakeAcronymsAbbreviations}{%
3766   \renewcommand*{\newacronym}[4][]{%
3767     \glsxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3768   }%
3769   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
3770   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
3771   \renewcommand*{\setacronymstyle}[1]{%
3772     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
3773       unavailable.
3774       Use \string\setabbreviationstyle\space instead.
3775       The original acronym interface can be restored with
3776       \string\RestoreAcronyms}{}%
3777   }%
3778   \renewcommand*{\newacronymstyle}[1]{%
3779     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3780       available unless you restore the original acronym interface with
3781       \string\RestoreAcronyms}%
3782     \@glsxtr@org@newacronymstyle{##1}%
3783   }%
3784 }

```

Switch acronyms to abbreviations:

```

3785 \MakeAcronymsAbbreviations

```

**RestoreAcronyms** Restore acronyms to glossaries interface.

```

3786 \newcommand*\RestoreAcronyms}{%
3787   \SetGenericNewAcronym
3788   \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3789   \renewcommand{\acronymfont}[1]{##1}%
3790   \let\setacronymstyle\@glxtr@org@setacronymstyle
3791   \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.

```

3792   \renewcommand*\@gls@link@checkfirsthyper{%
3793     \ifglused{\glslabel}%
3794     {\let\glxtrifwasfirstuse\@secondoftwo}
3795     {\let\glxtrifwasfirstuse\@firstoftwo}%
3796     \@glxtr@org@checkfirsthyper
3797   }
3798   \glssetcategoryattribute{acronym}{regular}{false}%
3799   \setacronymstyle{long-short}%
3800 }

```

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

```

3801 \renewcommand*\glsacspace}[1]{%
3802   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3803   \ifdim\dimen@<\glsacspacemax~\else\space\fi
3804 }

```

`\glsacspacemax` Value used in the above.

```

3805 \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`.

`\reg@glosslist`

```

3806 \newcommand*\@glxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

3807 \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`

```

3808 \renewcommand*\makeglossaries[1][]{%
3809   \ifx\@glxtr@record@setting\@glxtr@record@setting@only

```

```

3810 \PackageError{glossaries-extra}{\string\makeglossaries\space
3811 not permitted\MessageBreak with record=only package option}%
3812 {You may only use \string\makeglossaries\space with
3813 record=off or record=alsoindex options}%
3814 \else
3815 \ifblank{#1}%
3816 {\@glxtr@org@makeglossaries}%
3817 {%
3818 \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3819 \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3820 not permitted\MessageBreak with record=alsoindex package option}%
3821 {You may only use the hybrid \string\makeglossaries[...]\space with
3822 record=off option}%
3823 \else
3824 \edef\@glxtr@reg@glosslist{#1}%
3825 \ifundef{\glswrite}{\newwrite\glswrite}{}%
3826 \protected@write\@auxout{\string\providecommand
3827 \string\@glxtr@reg@glosslist{#1}}
3828 \protected@write\@auxout{\string\providecommand
3829 \string\@glxtr@reg@glosslist{#1}}
3830 \protected@write\@auxout{\string\@glxtr@reg@glosslist{#1}}
3831 \protected@write\@auxout{\string\@glxtr@reg@glosslist{#1}}
3832 \protected@write\@auxout{\string\@glxtr@reg@glosslist{#1}}
3833 \write\@auxout{\string\providecommand\string\@glxtr@reg@glosslist{#1}}

```

Iterate through each supplied glossary type and activate it.

```

3834 \for\@glo@type:=#1\do{%
3835 \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3836 }%

```

New glossaries must be created before \makeglossaries:

```

3837 \renewcommand*\newglossary[4][]{%
3838 \PackageError{glossaries}{New glossaries
3839 must be created before \string\makeglossaries}{You need
3840 to move \string\makeglossaries\space after all your
3841 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

3842 \let\@makeglossary\relax
3843 \let\makeglossary\relax
3844 \renewcommand\makeglossaries[1][]{%

```

Disable all commands that have no effect after \makeglossaries

```

3845 \@disable@onlypremakeg

```

Allow see key:

```

3846 \let\gls@checkseeallowed\relax

```

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

```

3847 \renewcommand*\@do@seeglossary[2]{%
3848 \glsdoifexists{##1}%

```

```

3849      {%
3850      \edef\@gls@label{\glsdetoklabel{##1}}%
3851      \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3852      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3853      {\@glsxtr@org@doseeglossary{##1}{##2}}%
3854      {%
3855      \@@glsxtrwrglossmark
3856      \protected@write\@auxout{}{%
3857      \string\@gls@reference
3858      {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
3859      }%
3860      }%
3861      }%
3862      }%

Adjust \@do@wrglossary
3863      \let\@glsxtr@do@wrglossary\@do@wrglossary
3864      \def\@do@wrglossary{%
3865      \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3866      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3867      {\@glsxtr@do@wrglossary}%
3868      {\gls@noidxglossary}%
3869      }%

Suppress warning about no \makeglossaries
3870      \let\warn@nomakeglossaries\relax
3871      \def\warn@noprintglossary{%
3872      \GlossariesWarningNoLine{No \string\printglossary\space
3873      or \string\printglossaries\space
3874      found.^^J(Remove \string\makeglossaries\space if you don't want
3875      any glossaries.)^^JThis document will not have a glossary}%
3876      }%

Only warn for glossaries not listed.
3877      \renewcommand{\@gls@noref@warn}[1]{%
3878      \edef\@gls@type{##1}%
3879      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3880      {%
3881      \GlossariesExtraWarning{Can't use
3882      \string\printnoidxglossary[type={\@gls@type}]
3883      when '\@gls@type' is listed in the optional argument of
3884      \string\makeglossaries}%
3885      }%
3886      {%
3887      \GlossariesWarning{Empty glossary for
3888      \string\printnoidxglossary[type={##1}].
3889      Rerun may be required (or you may have forgotten to use
3890      commands like \string\gls)}%
3891      }%
3892      }%

```

Adjust display number list to check for type:

```

3893      \renewcommand*\glsdisplaynumberlist}[1]{%
3894          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3895          {\@glsxtr@idx@displaynumberlist{##1}}%
3896          {\@glsxtr@noidx@displaynumberlist{##1}}%
3897      }%

```

Adjust entry list:

```

3898      \renewcommand*\glsentrynumberlist}[1]{%
3899          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3900          {\@glsxtr@idx@entrynumberlist{##1}}%
3901          {\@glsxtr@noidx@entrynumberlist{##1}}%
3902      }%

```

Adjust number list loop

```

3903      \renewcommand*\glsnumberlistloop}[2]{%
3904          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3905          {%
3906              \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3907                  not available for glossary ‘##1’}{}%
3908          }%
3909          {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3910      }%

```

Only sanitize sort for normal indexing glossaries.

```

3911      \renewcommand*\glsprestandardsort}[3]{%
3912          \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3913          {%
3914              \glsdosanitizesort
3915          }%
3916          {%
3917              \ifglssanitizesort
3918                  \@gls@noidx@sanitizesort
3919              \else
3920                  \@gls@noidx@nosanitizesort
3921              \fi
3922          }%
3923      }%

```

Unlike `\makenoidxglossaries` we can't automatically set `sanitizesort=false`. All entries must be defined in the preamble.

```

3924      \renewcommand*\new@glossaryentry[2]{%
3925          \PackageError{glossaries-extra}{Glossary entries must be defined
3926              in the preamble\MessageBreak when you use the optional argument
3927              of \string\makeglossaries}{Either move your definitions to the
3928              preamble or don't use the optional argument of
3929              \string\makeglossaries}%
3930      }%

```

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).

```

3931      \let\@glo@assign@sortkey\@glstr@mixed@assign@sortkey
3932      \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

3933      \expandafter\@glstr@gettype\expandafter,\@glstr@printglossopts,%
3934      type=\glsdefaulttype,\@end@glstr@gettype
3935      \def\@glo@sorttype{\@glo@default@sorttype}%
3936      }%

```

Check automake setting:

```

3937      \ifglautomake
3938      \renewcommand*{\@gls@doautomake}{%
3939      \@for\@gls@type:=\@glstr@reg@glosslist\do{%
3940      \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3941      }%
3942      }%
3943      \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3944      \ifdefined\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3945      \fi
3946      }%
3947      \fi
3948      }

```

The optional argument version of \makeglossaries needs an adjustment to \@printglossary to allow \@glo@assign@sortkey to pick up the glossary type.

rgprintglossary This no longer simply saves \@printglossary with \let but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes \provideignoredglossary to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

3949 \newcommand{\@glstr@orgprintglossary}[2]{%
3950 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3951 \def\glossarytitle{%
3952 \ifcsdef{\@glo@type\@glo@type @title}%
3953 {\csuse{\@glo@type\@glo@type @title}}%
3954 {\glossaryname}}%
3955 \def\glossarytoctitle{\glossarytitle}%
3956 \let\org@glossarytitle\glossarytitle
3957 \def\@glossarystyle{%
3958 \ifx\@glossary@default@style\relax
3959 \GlossariesWarning{No default glossary style provided \MessageBreak
3960 for the glossary '\@glo@type'. \MessageBreak
3961 Using deprecated fallback. \MessageBreak
3962 To fix this set the style with \MessageBreak
3963 \string\setglossarystyle\space or use the \MessageBreak
3964 style key=value option}%
3965 \fi

```

```

3966 }%
3967 \def\gls@dotocitle{\glssettocitle{\@glo@type}}%
3968 \let\org@glossaryentrynumbers\glossaryentrynumbers
3969 \bgroup
3970 \@printgloss@setsort
3971 \setkeys{printgloss}{#1}%
3972 \ifx\glossarytitle\org@glossarytitle
3973 \else
3974 \cslet{\@glo@type\@glo@type @title}{\glossarytitle}%
3975 \fi
3976 \let\currentglossary\@glo@type
3977 \let\org@glossaryentrynumbers\glossaryentrynumbers
3978 \let\glsnonextpages\@glsnonextpages
3979 \let\glsnextpages\@glsnextpages

3980 \glsxtractivatenopost
3981 \gls@dotocitle
3982 \@glossarystyle
3983 \let\gls@org@glossaryentryfield\glossentry
3984 \let\gls@org@glossarysubentryfield\subglossentry
3985 \renewcommand{\glossentry}[1]{%
3986 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3987 \gls@org@glossaryentryfield{##1}%
3988 }%
3989 \renewcommand{\subglossentry}[2]{%
3990 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3991 \gls@org@glossarysubentryfield{##1}{##2}%
3992 }%
3993 \@gls@preglossaryhook
3994 #2%
3995 \egroup
3996 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
3997 \global\let\warn@noprintglossary\relax
3998 }

```

`ractivatenopost` Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```

3999 \newcommand*{\glsxtractivatenopost}{%
4000 \let\nopostdesc\@nopostdesc
4001 \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4002 }

```

`lsxtrnopostpunc`

```

4003 \newrobustcmd*{\glsxtrnopostpunc}{%

```

`sxtr@nopostpunc` Provide a command that works like `\nopostdesc` but only switches of the punctuation without suppressing the post-description hook.

```

4004 \newcommand{\@glsxtr@nopostpunc}{%
4005 \let\@glsxtr@org@postdescription\glspostdescription
4006 \ifglsnopostdot

```

```

4007 \renewcommand{\glspostdescription}{%
4008 \glsnopostdottrue
4009 \let\glspostdescription\@glxtr@org@postdescription
4010 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4011 \glxtrpostdescription
4012 \@glxtr@nopostpunc@postdesc}%
4013 \else
4014 \renewcommand{\glspostdescription}{%
4015 \let\glspostdescription\@glxtr@org@postdescription
4016 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4017 \glxtrpostdescription
4018 \@glxtr@nopostpunc@postdesc}%
4019 \fi
4020 \glsnopostdotfalse
4021 }

```

stpostpunc@postdesc

```

4022 \newcommand*{\@glxtr@nopostpunc@postdesc}{}

```

restore@postpunc

```

4023 \newcommand*{\@glxtr@restore@postpunc}{%
4024 \def\@glxtr@nopostpunc@postdesc{%
4025 \@glxtr@org@postdescription
4026 \let\@glxtr@nopostpunc@postdesc\@empty
4027 \let\glxtrrestorepostpunc\@empty
4028 }%
4029 }

```

restorepostpunc Does nothing outside of glossary.

```

4030 \newcommand*{\glxtrrestorepostpunc}{}

```

\@printglossary Redefine.

```

4031 \renewcommand{\@printglossary}[2]{%
4032 \def\@glxtr@printglossopts{#1}%
4033 \@glxtr@orgprintglossary{#1}{#2}%
4034 }

```

Add a key that switches off the entry targets:

```

4035 \define@choicekey{printgloss}{target}
4036 [\@glxtr@printglossval\@glxtr@printglossnr]%
4037 {true,false}[true]%
4038 {%
4039 \ifcase\@glxtr@printglossnr
4040 \def\@glstarget{\glsdohypertarget}%
4041 \else
4042 \let\@glstarget\@secondoftwo
4043 \fi
4044 }

```



hypernameprefix

```
4045 \newcommand{\@glxtrhypernameprefix}{}

New to v1.20:
```

```
4046 \define@key{printgloss}{targetnameprefix}{%
4047   \renewcommand{\@glxtrhypernameprefix}{#1}%
4048 }
```

```
4049 \define@key{printgloss}{prefix}{%
4050   \renewcommand{\glo@linkprefix}{#1}%
4051 }
```

lsdohypertarget   Redefine to insert \@glxtrhypernameprefix before the target name.

```
4052 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4053 \renewcommand{\glsdohypertarget}[2]{%
4054   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4055 }
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
4056 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4057   \def\@glstarget{\glsdohypertarget}%
4058 \fi
4059 %\end{macro}
```

@makeglossaries   For the benefit of makeglossaries

```
4060 \newcommand*{\@glxtr@makeglossaries}[1]{}

@glxtr@gettype    Get just the type.
```

```
4061 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4062   \def\@glo@type{#2}%
4063 }
```

@assign@sortkey   Assign the sort key.

```
4064 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4065   \edef\@glo@type{\@glo@type}%
4066   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4067   {%
4068     \@glo@no@assign@sortkey{#1}%
4069   }%
4070   {%
4071     \@glo@assign@sortkey{#1}%
4072   }%
4073 }%
```

Display number list for the regular version:

splaynumberlist

```
4074 \let\@glxtr@idx@displaynumberlist\glstdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```

4075 \newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
4076   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4077   \ifdef\@gls@loclist
4078   {%
4079     \def\@gls@noidx@loclist@sep{%
4080       \def\@gls@noidx@loclist@sep{%
4081         \def\@gls@noidx@loclist@sep{%
4082           \glsnumlistsep
4083         }%
4084         \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4085       }%
4086     }%
4087     \def\@gls@noidx@loclist@finalsep{}%
4088     \def\@gls@noidx@loclist@prev{}%
4089     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4090     \@gls@noidx@loclist@finalsep
4091     \@gls@noidx@loclist@prev
4092   }%
4093   {%

4094   \glsxtrundeftag
4095   \glsdoifexists{#1}%
4096   {%
4097     \GlossariesWarning{Missing location list for ‘#1’. Either
4098       a rerun is required or you haven’t referenced the entry.}%
4099   }%
4100 }%
4101 }%
4102
```

And for the number list loop:

@numberlistloop

```

4103 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
4104   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4105   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4106   \let\@gls@org@glsseeformat\glsseeformat
4107   \let\glsnoidxdisplayloc#2\relax
4108   \let\glsseeformat#3\relax
4109   \ifdef\@gls@loclist
4110   {%
4111     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4112   }%
4113   {%

4114   \glsxtrundeftag
4115   \glsdoifexists{#1}%

```

```

4116   {%
4117       \GlossariesWarning{Missing location list for ‘##1’. Either
4118           a rerun is required or you haven’t referenced the entry.}%
4119   }%
4120 }%
4121 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4122 \let\glsseeformat\@gls@org@glsseeformat
4123 }%

```

Same for entry number list.

entrynumberlist

```

4124 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4125   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4126   \ifdef\@gls@loclist
4127   {%
4128       \glsnoidxloclist{\@gls@loclist}%
4129   }%
4130   {%
4131       \glsxtrundeftag
4132       \glsdoifexists{#1}%
4133       {%
4134           \GlossariesWarning{Missing location list for ‘##1’. Either
4135               a rerun is required or you haven’t referenced the entry.}%
4136       }%
4137   }%
4138 }%

```

entrynumberlist

```

4139 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4140 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4141   \protected@edef\@glsxtr@titlelabel{#1}%
4142   \ifdefvoid\@glsxtr@titlelabel
4143   {}%
4144   {%
4145       \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle{#1}}}%
4146   }%
4147   \ifdefvoid{\@glsxtr@titlelabel}%
4148   {%
4149       \DTLifint{#1}%
4150   {%
4151       \ifnum#1<256\relax
4152       \edef#2{\char#1\relax}%
4153       \else
4154       \edef#2{#1}%
4155       \fi
4156   }%

```

```

4157   {%
4158       \ifcsundef{#1groupname}%
4159       {\def#2{#1}}%
4160       {\letcs#2{#1groupname}}%
4161   }%
4162 }%
4163 {%
4164     \let#2\@glsxtr@titlelabel
4165 }%
4166 }

```

**g@getgrouptitle** Save original definition of \@gls@getgrouptitle

```

4167 \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.

```

4168 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4169     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4170     \@onelevel@sanitize\@glsxtr@titlelabel
4171     \ifcsdef{\@glsxtr@titlelabel}
4172     {\letcs{#2}{\@glsxtr@titlelabel}}%
4173     {\glsxtr@org@getgrouptitle{#1}{#2}}%
4174 }
4175 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

**trsetgrouptitle** Sets the title for the given group label.

```

4176 \newcommand{\glsxtrsetgrouptitle}[2]{%
4177     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4178     \@onelevel@sanitize\@glsxtr@titlelabel
4179     \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4180 }

```

**alsetgrouptitle** As above put only locally defines the title.

```

4181 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4182     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4183     \@onelevel@sanitize\@glsxtr@titlelabel
4184     \protected@csedef{\@glsxtr@titlelabel}{#2}%
4185 }

```

**\glsnavigation** Redefine to use new user-level command.

```

4186 \renewcommand*{\glsnavigation}{%
4187     \def\@gls@between{}%
4188     \ifcsundef{\@gls@hypergroup@list@{\@gls@type}}%
4189     {%
4190         \def\@gls@list{}%
4191     }%
4192     {%
4193         \expandafter\let\expandafter\@gls@list

```

```

4194     \csname @gls@hypergroup@list@\@glo@type\endcsname
4195 }%
4196 \for\@gls@tmp:=\@gls@list\do{%
4197   \@gls@between
4198   \glstrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4199   \glshyperlink{\@gls@tmp}{\@gls@grptitle}%
4200   \let\@gls@between\glshypernavsep
4201 }%
4202 }

```

@noidx@glossary

```

4203 \renewcommand*{\@print@noidx@glossary}{%
4204   \ifcsdef{@gls@ref@\@glo@type}%
4205   {%
4206     \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
4207     {%
4208       \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
4209     }%
4210   }%
4211   \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4212 }%
4213 \glossarysection[\glossarytoctitle]{\glossarytitle}%
4214 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

4215   \def\@gls@currentlettergroup{%
4216     \begin{theglossary}%
4217     \glossaryheader
4218     \glsresetentrylist
4219     \forlistcsloop{\@gls@noidx@do}{\@gls@ref@\@glo@type}%
4220     \end{theglossary}%
4221     \glossarypostamble
4222   }%
4223   {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

4224     \glstrifemptyglossary{\@glo@type}%
4225     {}%
4226     {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4227     \@gls@noref@warn{\@glo@type}%
4228   }%
4229 }

```

noidxdisplayloc Patch to check for range formations.

```

4230 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4231   \setentrycounter[#1]{#2}%
4232   \@glstr@display@loc#3\empty\end@glstr@display@loc{#4}%
4233 }

```

tr@display@loc Patch to check for range formations.

```
4234 \def\@glxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4235   \ifx#1\relax
4236     \glxtrdisplaystartloc{#2}{#3}%
4237   \else
4238     \ifx#1)\relax
4239       \glxtrdisplayendloc{#2}{#3}%
4240     \else
4241       \glxtrdisplaysingleloc{#1#2}{#3}%
4242     \fi
4243   \fi
4244 }
```

isplayingleloc Single location.

```
4245 \newcommand*\glxtrdisplaysingleloc}[2]{%
4246   \csuse{#1}{#2}%
4247 }
```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glxtrlocrangefmt`.

displaystartloc Start of a location range.

```
4248 \newcommand*\glxtrdisplaystartloc}[2]{%
4249   \edef\glxtrlocrangefmt{#1}%
4250   \ifx\glxtrlocrangefmt\empty
4251     \def\glxtrlocrangefmt{glnumberformat}%
4252   \fi
4253   \expandafter\glxtrdisplaysingleloc
4254   \expandafter{\glxtrlocrangefmt}{#2}%
4255 }
```

trdisplayendloc End of a location range.

```
4256 \newcommand*\glxtrdisplayendloc}[2]{%
4257   \edef\@glxtr@tmp{#1}%
4258   \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{glnumberformat}}{}}%
4259   \ifx\glxtrlocrangefmt\@glxtr@tmp
4260   \else
4261     \GlossariesExtraWarning{Mismatched end location range
4262       (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
4263   \fi
4264   \expandafter\glxtrdisplayendloohook\expandafter{\@glxtr@tmp}{#2}%
4265   \expandafter\glxtrdisplaysingleloc
4266   \expandafter{\glxtrlocrangefmt}{#2}%
4267   \def\glxtrlocrangefmt{}%
4268 }
```

isplayendloohook Allow the user to hook into the end of range command.

```
4269 \newcommand*\glxtrdisplayendloohook}[2]{}
```

gsxtrlocrangefmt Current range format. Empty if not in a range.

```
4270 \newcommand*{\gsxtrlocrangefmt}{}%
```

gssetentrycounter Adjust \gssetentrycounter to save the original prefix.

```
4271 \renewcommand*{\gssetentrycounter}[2] [] {%
4272   \def\gsxtrcounterprefix{#1}%
4273   \ifx\gsxtrcounterprefix\@empty
4274     \def\@glo@counterprefix{.}%
4275   \else
4276     \def\@glo@counterprefix{.#1.}%
4277   \fi
4278   \def\glsentrycounter{#2}%
4279 }
```

gs@removespaces Redefine to allow adjustments to location hyperlink.

```
4280 \def\@gls@removespaces#1 #2\@nil{%
4281   \toks@=\expandafter{\the\toks@#1}%
4282   \ifx\@#2\@
4283     \edef\x{\the\toks@}%
4284     \ifx\x\empty
4285     \else
```

Expand location (just in case \toks@ is needed for something else).

```
4286     \expandafter\gsxtrlocationhyperlink\expandafter
4287     \glsentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4288   \fi
4289 \else
4290   \@gls@ReturnAfterFi{%
4291     \@gls@removespaces#2\@nil
4292   }%
4293 \fi
4294 }
```

gslocationhyperlink

```
4295 \newcommand*{\gsxtrlocationhyperlink}[3] {%
4296   \ifdefined\gsxtrsupplocationurl
4297   {%
4298     \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4299   }%
4300   {%
4301     \hyperref{\gsxtrsupplocationurl}{#1}{#2}{#3}%
4302   }%
4303 }
```

gsuppnumber

```
4304 \newcommand*{\gsxtrsupphypernumber}[1] {%
4305   {%
4306     \glsattribute{\glscurrententrylabel}{externallocation}%
4307   }%
```

```

4308     \def\glxtrsupplocationurl{%
4309         \glxgetattribute{\glscurrententrylabel}{externallocation}}%
4310     }%
4311     {%
4312         \def\glxtrsupplocationurl{%
4313             }%
4314         \glshypernumber{#1}%
4315     }%
4316 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4317 \renewcommand{\@print@glossary}{%
4318     \makeatletter
4319     \@input{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
4320     \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
4321     {}%
4322     {\glxtrNoGlossaryWarning{\@glo@type}}%
4323     \ifglxindy
4324         \ifcsundef{@xdy\@glo@type @language}%
4325         {%
4326             \edef\@do@auxoutstuff{%
4327                 \noexpand\AtEndDocument{%
4328                     \noexpand\immediate\noexpand\write\@auxout{%
4329                         \string\providecommand\string\@xdylanguage[2]{}}%
4330                     \noexpand\immediate\noexpand\write\@auxout{%
4331                         \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4332                 }%
4333             }%
4334         }%
4335         {%
4336             \edef\@do@auxoutstuff{%
4337                 \noexpand\AtEndDocument{%
4338                     \noexpand\immediate\noexpand\write\@auxout{%
4339                         \string\providecommand\string\@xdylanguage[2]{}}%
4340                     \noexpand\immediate\noexpand\write\@auxout{%
4341                         \string\@xdylanguage{\@glo@type}{\csname @xdy\@glo@type
4342                             @language\endcsname}}%
4343                 }%
4344             }%
4345         }%
4346         \@do@auxoutstuff
4347         \edef\@do@auxoutstuff{%
4348             \noexpand\AtEndDocument{%
4349                 \noexpand\immediate\noexpand\write\@auxout{%
4350                     \string\providecommand\string\@gls@codepage[2]{}}%
4351                 \noexpand\immediate\noexpand\write\@auxout{%
4352                     \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%

```



```

4353     }%
4354 }%
4355 \do@auxoutstuff
4356 \fi
4357 \renewcommand*{\@warn@nomakeglossaries}{%
4358   \GlossariesWarningNoLine{\string\makeglossaries\space
4359     hasn't been used,^^Jthe glossaries will not be updated}%
4360 }%
4361 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

4362 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4363   This document is incomplete. The external file associated with
4364   the glossary '#1' (which should be called \texttt{#2})
4365   hasn't been created.%
4366 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

4367 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4368   This has probably happened because there are no entries defined
4369   in this glossary.%
4370 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

4371 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4372   If you don't want this glossary,
4373   add \texttt{nomain} to your package option list when you load
4374   \texttt{glossaries-extra.sty}. For example:%
4375 }

```

`\GlsWarningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4376 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4377   Did you forget to use \texttt{type=#1} when you defined your
4378   entries? If you tried to load entries into this glossary with
4379   \texttt{\string\loadglsentries} did you remember to use
4380   \texttt{[#1]} as the optional argument? If you did, check that
4381   the definitions in the file you loaded all had the type set
4382   to \texttt{\string\glsdefaulttype}.%
4383 }

```

`\GlsWarningCheckFile` Advisory message to check the file contents.

```

4384 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4385   Check the contents of the file \texttt{#1}. If
4386   it's empty, that means you haven't indexed any of your entries in this
4387   glossary (using commands like \texttt{\string\gls} or
4388   \texttt{\string\glsadd}) so this list can't be generated.
4389   If the file isn't empty, the document build process hasn't been

```

```

4390 completed.%
4391 }

```

WarningAutoMake Message when automake option has been used.

```

4392 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4393   You may need to rerun \LaTeX. If you already have, it may be that
4394   \TeX's shell escape doesn't allow you to run
4395   \ifglxindy xindy\else makeindex\fi. Check the
4396   transcript file \texttt{\jobname.log}. If the shell escape is
4397   disabled, try one of the following:
4398
4399   \begin{itemize}
4400     \item Run the external (Lua) application:
4401
4402       \texttt{makeglossaries-lite.lua \string\jobname\string}
4403
4404     \item Run the external (Perl) application:
4405
4406       \texttt{makeglossaries \string\jobname\string}
4407   \end{itemize}
4408
4409   Then rerun \LaTeX\ on this document.
4410   \GlossariesExtraWarning{Rerun required to build the
4411     glossary '#1' or check TeX's shell escape allows
4412     you to run \ifglxindy xindy\else makeindex\fi}%
4413 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4414 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4415   You need to either replace \texttt{\string\makenoidxglossaries}
4416   with \texttt{\string\makeglossaries} or replace
4417   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4418   \texttt{\string\printnoidxglossary}
4419   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4420   this document.%
4421 }

```

WarningBuildInfo Build advice.

```

4422 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4423   Try one of the following:
4424   \begin{itemize}
4425     \item Add \texttt{automake} to your package option list when you load
4426       \texttt{glossaries-extra.sty}. For example:
4427
4428       \texttt{\string\usepackage[automake]%
4429         \glsopenbrace glossaries-extra\glsclosebrace}
4430
4431     \item Run the external (Lua) application:

```

```

4432
4433         \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4434
4435     \item Run the external (Perl) application:
4436
4437         \texttt{makeglossaries \string"\jobname\string"}
4438 \end{itemize}
4439
4440 Then rerun \LaTeX\ on this document.%
4441 }

trRecordWarning Paragraph for record=only.
4442 \newcommand{\GlsXtrRecordWarning}[1]{%
4443 \texttt{\string\printglossary} doesn't work
4444 with the \texttt{record=only} package option
4445 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4446 instead (or change the package option).%
4447 }

oGlsWarningTail Final paragraph.
4448 \newcommand{\GlsXtrNoGlsWarningTail}{%
4449 This message will be removed once the problem has been fixed.%
4450 }

GlsWarningNoOut No out file created. Build advice.
4451 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4452 The file \texttt{#1} doesn't exist. This most likely means you haven't used
4453 \texttt{\string\makeglossaries} or you have used
4454 \texttt{\string\nofiles}. If this is just a draft version of the
4455 document, you can suppress this message using the
4456 \texttt{nomissingglstext} package option.%
4457 }

glossarywarning
4458 \newcommand*{\@glsxtr@defaultnoglossarywarning}[1]{%
4459 \glossarysection[\glossarytoctitle]{\glossarytitle}
4460 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4461 \par
4462 \glsxtrifemptyglossary{#1}%
4463 {%
4464 \GlsXtrNoGlsWarningEmptyStart\space
4465 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4466 \medskip
4467 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]}%
4468 \glsopenbrace glossaries-extra\glsclosebrace}
4469 \medskip
4470 }%
4471 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4472 }%

```

```

4473 {%
4474   \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4475   {%
4476     \GlsXtrNoGlsWarningCheckFile
4477     {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4478
4479     \ifglautomake
4480
4481     \GlsXtrNoGlsWarningAutoMake{#1}
4482
4483   \else
4484
4485     \ifthenelse{\equal{#1}{main}}{%
4486       {%
4487         \GlsXtrNoGlsWarningEmptyMain\par
4488         \medskip
4489         \noindent\texttt{\string\usepackage[nomain]%
4490           \glsopenbrace glossaries-extra\glsclosebrace}
4491         \medskip
4492       }%
4493     }%
4494
4495     \ifdefequal\makeglossaries\@no@makeglossaries
4496     {%
4497       \GlsXtrNoGlsWarningMisMatch
4498     }%
4499     {%
4500       \GlsXtrNoGlsWarningBuildInfo
4501     }%
4502   \fi
4503 }%
4504 {%
4505   \GlsXtrNoGlsWarningNoOut
4506   {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4507 }%
4508 }%
4509 \par
4510 \GlsXtrNoGlsWarningTail
4511 }

```

**glossarywarning** Warn about using `\printglossary` with `record`

```

4512 \newcommand*{\@glstr@record@noglossarywarning}[1]{%
4513   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4514     with record=only package option\MessageBreak(use
4515     \string\printunsrtglossary[type=#1])\MessageBreak
4516     instead (or change the package option)}%
4517   \glossarysection[\glossarytoctitle]{\glossarytitle}
4518   \GlsXtrRecordWarning{#1}
4519   \GlsXtrNoGlsWarningTail

```

4520 }

Provide some commands to accompany the record option for use with **bib2gls**.

**glxtrresourcefile** 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.

4521 \newcommand\*{\glxtrresourcefile}[2] [] {%

The record option can't be set after this command.

4522 \disable@keys{glossaries-extra.sty}{record}%

4523 \glxtr@writefields

4524 \protected@write\@auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%

4525 \let\@glxtr@org@see@noindex\@glxtr@see@noindex

4526 \let\@glxtr@see@noindex\relax

4527 \IfFileExists{#2.glstex}%

4528 {%

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

4529 \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%

4530 \makeatletter

4531 \@input{#2.glstex}%

4532 \@bibgls@restreat

4533 }%

4534 {%

4535 \GlossariesExtraWarning{No file '#2.glstex'}%

4536 }%

4537 \let\@glxtr@see@noindex\@glxtr@org@see@noindex

4538 }

4539 \@onlypreamble\glxtrresourcefile

**glxtrresourceinit** Code used during the protected write operation.

4540 \newcommand\*{\glxtrresourceinit}{}%

**glxtrresourcecount**

4541 \newcount\glxtrresourcecount

**glxtrLoadResources** Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

4542 \newcommand\*{\GlsXtrLoadResources}[1] [] {%

4543 \ifnum\glxtrresourcecount=0\relax

4544 \glxtrresourcefile[#1]{\jobname}%

4545 \else

4546 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%

4547 \fi

4548 \advance\glxtrresourcecount by 1\relax

4549 }

**glxtr@resource**

4550 \newcommand\*{\glxtr@resource}[2] {}%

```

\glxtr@fields
4551 \newcommand*{\glxtr@fields}[1]{}

xtr@texencoding
4552 \newcommand*{\glxtr@texencoding}[1]{}

\glxtr@langtag
4553 \newcommand*{\glxtr@langtag}[1]{}

@pluralsuffixes
4554 \newcommand*{\glxtr@pluralsuffixes}[4]{}

tr@shortcutsval
4555 \newcommand*{\glxtr@shortcutsval}[1]{}

sxtr@linkprefix
4556 \newcommand*{\glxtr@linkprefix}[1]{}

xtr@writefields This information only needs to be written once, so disable it after it's been used.
4557 \newcommand*{\glxtr@writefields}{%

4558 \protected@write\@auxout{}%
4559 {\string\providecommand*{\string\glxtr@fields}[1]{}}%
4560 \protected@write\@auxout{}%
4561 {\string\providecommand*{\string\glxtr@resource}[2]{}}%
4562 \protected@write\@auxout{}%
4563 {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}}%
4564 \protected@write\@auxout{}%
4565 {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}%
4566 \protected@write\@auxout{}%
4567 {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}%
4568 \protected@write\@auxout{}{\string\glxtr@fields{\@gls@keymap}}%

4569 \protected@write\@auxout{}%
4570 {\string\providecommand*{\string\glxtr@record}[5]{}}%

If any languages have been loaded, the language tag will be available in \CurrentTrackedLanguageTag
(provided by tracklang). For multilingual documents, the required locale will have to be indi-
cated in the sort key when using \glxtrresourcefile.

4571 \ifdef\CurrentTrackedLanguageTag
4572 {%
4573 \protected@write\@auxout{}{%
4574 \string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
4575 }%
4576 {}%
4577 \protected@write\@auxout{}{\string\glxtr@pluralsuffixes
4578 {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4579 {\glxtrabbrvpluralsuffix}}%

```

```

4580 \ifdef\inputencodingname
4581 {%
4582   \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
4583 }%
4584 {%

```

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.)

```

4585   \@ifpackageloaded{fontspec}%
4586   {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}}%
4587   {}%
4588 }%
4589 \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.

```

4590 \AtBeginDocument
4591 {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}}%
4592 \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.

```

4593 \ifglautomake
4594 \IfFileExists{\jobname.aux}%
4595 {\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.

```

4596 \ifx\@gls@doautomake\@gls@doautomake@err
4597 \let\@gls@doautomake\relax
4598 \fi
4599 \fi
4600 }

```

do@automake@err

```

4601 \newcommand*\@gls@doautomake@err{%
4602   \PackageError{glossaries}{You must use
4603     \string\makeglossaries\space with automake=true}
4604   {%
4605     Either remove the automake=true setting or
4606     add \string\makeglossaries\space to your document preamble.%
4607   }%
4608 }

```

Allow locations specific to a particular counter to be recorded.

\glxtr@record

```

4609 \newcommand*\glxtr@record}[5]{%

```

r@counterrecord Aux file command.

```
4610 \newcommand*{\glxtr@counterrecord}[3]{%
4611   \glxtrfieldlistgadd{#1}{record.#2}{#3}%
4612 }
```

unterrecordhook Hook used by \@glxtr@dorecord.

```
4613 \newcommand*{\@glxtr@counterrecordhook}{}
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4614 \newcommand*{\GlsXtrRecordCounter}[1]{%
4615   \@glxtr@recordcounter{#1}%
4616 }
4617 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4618 \newcommand*{\@glxtr@docounterrecord}[1]{%
4619   \protected@write\@auxout{}\string\glxtr@counterrecord
4620     {\@glslabel}{#1}{\csuse{the#1}}}%
4621 }
```

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.

```
4622 \newcommand*{\glxtrglossentry}[1]{%
4623   \glxtrtitleorpdforheading
4624   {\@glxtrglossentry{#1}}%
4625   {\glsenryname{#1}}%
4626   {\glxtrheadname{#1}}%
4627 }
```

lsxtrglossentry Another test is needed in case \@glxtrglossentry has been written to the table of contents.

```
4628 \newrobustcmd*{\@glxtrglossentry}[1]{%
4629   \glxtrtitleorpdforheading
4630   {%
4631     \glsdodefexists{#1}%
4632     {%
4633       \begingroup
4634         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4635         \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4636         \ifglshasparent{#1}%
4637           {\GlsXtrStandaloneSubEntryItem{#1}}%
4638           {\glsenryitem{#1}}%
4639           \glstarget{#1}{\glossentryname{#1}}%
4640         \endgroup
```



```

4641 }%
4642 }%
4643 {\glsentryname{#1}}%
4644 {\glsxtrheadname{#1}}%
4645 }

```

**oneGlossaryType** To make it easier to adjust the definition of `\currentglossary` within `\glsxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4646 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}

```

**oneSubEntryItem** Used for sub-entries in standalone format. The argument is the entry's label.

```

4647 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4648   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}}%
4649 }

```

**glossentryother** As `\glsxtrglossentry` 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.

```

4650 \newcommand*{\glsxtrglossentryother}[3]{%
4651   \ifstrempy{#1}%
4652   {%
4653     \ifcsdef{glsxtrhead#3}%
4654     {%
4655       \glsxtrtitleorpdforheading
4656       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4657       {\@gls@entry@field{#2}{#3}}%
4658       {\csuse{glsxtrhead#3}{#2}}%
4659     }%
4660     {%
4661       \glsxtrtitleorpdforheading
4662       {\@glsxtrglossentryother{#2}{#3}{#1}}%
4663       {\@gls@entry@field{#2}{#3}}%
4664       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4665     }%
4666   }%
4667   {%
4668     \glsxtrtitleorpdforheading
4669     {\@glsxtrglossentryother{#2}{#3}{#1}}%
4670     {\@gls@entry@field{#2}{#3}}%
4671     {#1}%
4672   }%
4673 }

```

**glossentryother** As `\@glsxtrglossentry` but uses a different field.

```

4674 \newrobustcmd*{\@glsxtrglossentryother}[3]{%
4675   \glsxtrtitleorpdforheading

```

```

4676 {%
4677   \glsdoifexists{#1}%
4678   {%
4679     \begingroup
4680     \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4681     \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4682     \ifglshasparent{#1}%
4683     {\GlsXtrStandaloneSubEntryItem{#1}}%
4684     {\gl Sentryitem{#1}}%
4685     \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4686     \endgroup
4687   }%
4688 }%
4689 {\@gls@entry@field{#1}{#2}}%
4690 {#3}%
4691 }

```

`\printunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4692 \newcommand*{\printunsrtglossary}{%
4693   \ifstar\s@printunsrtglossary\@printunsrtglossary
4694 }

```

`\printunsrtglossary` Unstarred version.

```

4695 \newcommand*{\@printunsrtglossary}[1][ ]{%
4696   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4697 }

```

`\printunsrtglossary` Starred version.

```

4698 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4699   \begingroup
4700     #2%
4701     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4702   \endgroup
4703 }

```

`\printunsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```

4704 \newcommand*{\printunsrtglossaries}{%
4705   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4706 }

```

`\@unsrt@glossary`

```

4707 \newcommand*{\@print@unsrt@glossary}{%
4708   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4709   \glossarypreamble
4710   check for empty list
4710   \glxtrifemptyglossary{\@glo@type}%

```

```

4711 {%
4712   \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4713 }%
4714 {%

4715   \key@ifundefined{glossentry}{group}%
4716   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4717   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4718   \def\@gls@currentlettergroup{}}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

4719   \def\@glsxtr@doglossary{%
4720     \begin{theglossary}%
4721     \glossaryheader
4722     \glsresetentrylist
4723   }%
4724   \expandafter\@for\expandafter\@gls@currententrylabel\expandafter
4725   : \expandafter=\csname glist@\@glo@type\endcsname\do{%
4726     \ifdefempty{\@gls@currententrylabel}
4727     {}%
4728     {%

```

Provide a hook (for example to measure width).

```

4729     \let\@glsxtr@process\@firstofone
4730     \let\@printunsrtglossaryskipentry
4731     \@glsxtr@printunsrtglossaryskipentry
4732     \printunsrtglossaryentryprocesshook{\@gls@currententrylabel}%

```

Don't check group for child entries.

```

4733     \glsxtr@process
4734     {%
4735     \ifglshasparent{\@gls@currententrylabel}{}%
4736     {%
4737       \@glsxtr@checkgroup\@gls@currententrylabel
4738       \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
4739       {\@glsxtr@groupheading}%
4740     }%
4741     \eappto\@glsxtr@doglossary{%
4742       \noexpand\@printunsrt@glossary@handler{\@gls@currententrylabel}}%
4743     }%
4744   }%
4745 }%
4746 \appto\@glsxtr@doglossary{\end{theglossary}}%
4747 \printunsrtglossarypredoglossary
4748 \@glsxtr@doglossary
4749 }%
4750 \glossarypostamble
4751 }

```

entryprocesshook

```

4752 \newcommand*{\@printunsrtglossaryentryprocesshook}[1]{%

```

ossaryskipentry

```
4753 \newcommand*{\printunsrtglossaryskipentry}{%
4754   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4755   can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4756 }
```

entryprocesshook

```
4757 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
4758   \let\glxtr@process\@gobble
4759 }
```

rypredoglossary

```
4760 \newcommand*{\printunsrtglossarypredoglossary}{}%
```

lossary@handler

```
4761 \newcommand{\@printunsrt@glossary@handler}[1]{%
4762   \xdef\glscurrententrylabel{#1}%
4763   \printunsrtglossaryhandler\glscurrententrylabel
4764 }
```

glossaryhandler

```
4765 \newcommand{\printunsrtglossaryhandler}[1]{%
4766   \glxtrunsrtdo{#1}%
4767 }
```

xtriflabelinlist

```
\glxtriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@glx@ifinlist` which ensures the label and list are fully expanded.

```
4768 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4769   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@glx@ifinlist{#1}{#2}}%
4770   \@glxtr@doiflabelinlist{#3}{#4}%
4771 }
```

srtglossaryunit

```
4772 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
4773   \s@printunsrtglossary[type=\gldefaulttype,#1]{%
4774     \printunsrtglossaryunitsetup{#2}%
4775   }%
4776 }
```

ossaryunitsetup

```
4777 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4778   \renewcommand{\printunsrtglossaryhandler}[1]{%

```

```

4779 \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4780 {\glxtrunsrtdo{##1}}%
4781 {}%
4782 }%

```

Only the target names should have the prefixes adjusted as \gls etc need the original \glolinkprefix. The \@gobble part discards \glolinkprefix.

```

4783 \ifcsundef{theH#1}%
4784 {%
4785 \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4786 }%
4787 {%
4788 \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4789 }%
4790 \renewcommand*{\glossarysection}[2][{}]%
4791 \appto\glossarypostamble{\glspare\medskip\glspare}%
4792 }

```

srtglossaryunit

```

4793 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]%
4794 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4795 requires the record=only or record=alsoindex package option}{}%
4796 }

```

t@getgrouptitle

```

4797 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4798 \protected@edef\@glxtr@titlelabel{\glxtr@grouptitle@#1}%
4799 \@onelevel@sanitize\@glxtr@titlelabel
4800 \ifcsdef{\@glxtr@titlelabel}
4801 {\letcs{#2}{\@glxtr@titlelabel}}%
4802 {\def#2{#1}}%
4803 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

4804 \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).

```

4805 \newcommand*{\glxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glxtr@doglossary is being constructed rather than in the handler.

sxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glxtr@groupheading, which will be empty if no heading is required.

```

4806 \newcommand*{\@glsxtr@checkgroup}[1]{%
4807   \def\@glsxtr@groupheading{}%
4808   \key@ifundefined{glossentry}{group}%
4809   {%
4810     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
4811     \expandafter\glo@grabfirst\@gls@sort{}\@nil
4812   }%
4813   {%
4814     \protected@edef\@glo@thislettergrp{%
4815       \csuse{glo@\glsdetoklabel{#1}@\glsxtrgroupfield}}%
4816     }%
4817     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4818     {}%
4819     {%
4820       \ifdefempty{\@gls@currentlettergroup}{}%
4821       {\def\@glsxtr@groupheading{\gls@groupskip}}%
4822       \eappto\@glsxtr@groupheading{%
4823         \noexpand\gls@groupheading{\expandonce\@glo@thislettergrp}%
4824       }%
4825     }%
4826     \let\@gls@currentlettergroup\@glo@thislettergrp
4827 }

```

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.

```

4828 \newcommand{\@glsxtr@noidx@do}[1]{%
4829   \ifglsentryexists{#1}%
4830   {%
4831     \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4832     \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
4833     \ifgls@hasparent{#1}%
4834     {%
4835       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
4836       \ifdefvoid{\@gls@location}%
4837       {%
4838         \ifdefvoid{\@gls@loclist}%
4839         {%
4840           \subglossentry{\gls@level}{#1}{}%
4841         }%
4842         {%
4843           \subglossentry{\gls@level}{#1}%
4844           {%
4845             \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4846           }%
4847         }%
4848       }%
4849     }%
4850     \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4851   }%

```

```

4852 }%
4853 {%
4854 \ifdefvoid{\@gls@location}%
4855 {%
4856 \ifdefvoid{\@gls@loclist}
4857 {%
4858 \glossentry{#1}{}%
4859 }%
4860 {%
4861 \glossentry{#1}%
4862 {%
4863 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}}%
4864 }%
4865 }%
4866 }%
4867 {%
4868 \glossentry{#1}%
4869 {%
4870 \glossaryentrynumbers{\@gls@location}%
4871 }%
4872 }%
4873 }%
4874 }%
4875 {}%
4876 }

```

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`

```
4877 \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>}
```

```

4878 \newcommand*{\@glsxtrnewgls}[4]{%
4879 \ifdef{#3}%
4880 {%
4881 \PackageError{glossaries-extra}{Command \string#3\space already
4882 defined}{}%
4883 }%
4884 {%

```

```

4885 \ifcsdef{@#4like@#2}%
4886 {%
4887 \advance\@glxstrnewgls@inner by \@ne
4888 \def\@glxstrnewgls@innercsname{@#4like\number\@glxstrnewgls@inner @#2}%
4889 }%
4890 {\def\@glxstrnewgls@innercsname{@#4like@#2}}%
4891 \expandafter\newrobustcmd\expandafter*\expandafter
4892 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glxstrnewgls@innercsname\endcsname}%
4893 \ifstrempy{#1}%
4894 {%
4895 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][ ]{%
4896 \new@ifnextchar[%
4897 {\csname @#4@\endcsname{##1}{#2##2}}%
4898 {\csname @#4@\endcsname{##1}{#2##2}[ ]}%
4899 }%
4900 }%
4901 {%
4902 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][ ]{%
4903 \new@ifnextchar[%
4904 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4905 {\csname @#4@\endcsname{#1,##1}{#2##2}[ ]}%
4906 }%
4907 }%
4908 }%
4909 }

```

`\glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4910 \newrobustcmd*{\glxstrnewgls}[3][ ]{%
4911 \@glxstrnewgls{#1}{#2}{#3}{gls}%
4912 }

```

`\glxstrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glsp1`, `\Gls` and `\Glsp1` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4913 \newrobustcmd*{\glxstrnewglslike}[6][ ]{%
4914 \@glxstrnewgls{#1}{#2}{#3}{gls}%
4915 \@glxstrnewgls{#1}{#2}{#4}{glsp1}%
4916 \@glxstrnewgls{#1}{#2}{#5}{Gls}%
4917 \@glxstrnewgls{#1}{#2}{#6}{Glsp1}%
4918 }

```

`\glxstrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSp1` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4919 \newrobustcmd*{\glxstrnewGLSlike}[4][ ]{%

```



```

4920 \@glxstrnewgls{#1}{#2}{#3}{GLS}%
4921 \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
4922 }

```

\glxstrnewrgls As \glxstrnewgls but for \rgls.

```

4923 \newrobustcmd*{\glxstrnewrgls}[3][]{%
4924 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4925 }

```

sxtrnewrglslike As \glxstrnewrglslike but for \rgls etc.

```

4926 \newrobustcmd*{\glxstrnewrglslike}[6][]{%
4927 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4928 \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
4929 \@glxstrnewgls{#1}{#2}{#5}{rGLs}%
4930 \@glxstrnewgls{#1}{#2}{#6}{rGLspl}%
4931 }

```

sxtrnewrGLSlike As \glxstrnewrGLSlike but for \rGLS etc.

```

4932 \newrobustcmd*{\glxstrnewrGLSlike}[4][]{%
4933 \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
4934 \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
4935 }

```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```

4936 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
4937 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount}%
4938 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
4939 {0}%
4940 }

```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```

4941 \newcommand*{\GlsXtrRecordCount}[2]{%
4942 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
4943 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
4944 {0}%
4945 }

```

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.

```

4946 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4947 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}}%
4948 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}\endcsname}%
4949 {0}%
4950 }

```

trdetoklocation

```
4951 \newcommand*{\glxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
4952 \newcommand*{\glxtrenablerecordcount}{%
4953   \renewcommand*{\gls}{\rgls}%
4954   \renewcommand*{\Gls}{\rGls}%
4955   \renewcommand*{\glspl}{\rglspl}%
4956   \renewcommand*{\Glspl}{\rGlspl}%
4957   \renewcommand*{\GLS}{\rGLS}%
4958   \renewcommand*{\GLSpl}{\rGLSpl}%
4959 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
4960 \newcommand*{\glxtrrecordtriggervalue}[1]{%
4961   \GlsXtrTotalRecordCount{#1}%
4962 }
```

dCountAttribute

```
4963 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4964   \@for\@glxtr@cat:=#1\do
4965   {%
4966     \ifdefempty{\@glxtr@cat}{}%
4967     {%
4968       \glssetcategoryattribute{\@glxtr@cat}{recordcount}{#2}%
4969     }%
4970   }%
4971 }
```

trifrecordtrigger

`\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}`

```
4972 \newcommand*{\glxtrifrecordtrigger}[3]{%
4973   \glshasattribute{#1}{recordcount}%
4974   {%
4975     \ifnum\glxtrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
4976     #3%
4977   \else
4978     #2%
4979   \fi
4980 }%
4981 {#3}%
4982 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
4983 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
```

```

4984 \edef\glslabel{\glsdetoklabel{#2}}%
4985 \let\@gls@link@label\glslabel
4986 \def\@glsxtr@thevalue{%
4987 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4988 \def\@glsnumberformat{glstriggerrecordformat}%
4989 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
4990 \edef\gls@type{\csname glo@\glslabel @type\endcsname}%
4991 \def\@glsxtr@thevalue{%
4992 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4993 \glsxtrinitwrgloss
4994 \glslinkpresetkeys
4995 \setkeys{glslink}{#1}%
4996 \glslinkpostsetkeys
4997 \ifdefempty{\@glsxtr@thevalue}%
4998 {%
4999 \@gls@saveentrycounter
5000 }%
5001 {%
5002 \let\theglsentrycounter\@glsxtr@thevalue
5003 \def\theHglentrycounter{\@glsxtr@theHvalue}%
5004 }%
5005 \ifglsxtrinitwrglossbefore
5006 \do@wrglossary{#2}%
5007 \fi
5008 #3%
5009 \ifglsxtrinitwrglossbefore
5010 \else
5011 \do@wrglossary{#2}%
5012 \fi
5013 \ifKV@glslink@local
5014 \glslocalunset{#2}%
5015 \else
5016 \glsunset{#2}%
5017 \fi
5018 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5019 \newcommand*{\glstriggerrecordformat}[1]{

```

```

\rgls

```

```

5020 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rgls

```

```

5021 \newcommand*{\@rgls}[2][{}]{
5022 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5023 }

```

```

\@rgls@

```

```

5024 \def\@rgls@#1#2[#3]{%
5025   \glxtrifrecordtrigger{#2}%
5026   {%
5027     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5028   }%
5029   {%
5030     \@gl@s@{#1}{#2}[#3]%
5031   }%
5032 }%

```

\rglspl

```

5033 \newrobustcmd*{\rglspl}{\@gl@s@hyp@opt\@rglspl}

```

\@rglspl

```

5034 \newcommand*{\@rglspl}[2][{}]{%
5035   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5036 }

```

\@rglspl@

```

5037 \def\@rglspl@#1#2[#3]{%
5038   \glxtrifrecordtrigger{#2}%
5039   {%
5040     \@glxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5041   }%
5042   {%
5043     \@glspl@{#1}{#2}[#3]%
5044   }%
5045 }%

```

\rGls

```

5046 \newrobustcmd*{\rGls}{\@gl@s@hyp@opt\@rGls}

```

\@rGls

```

5047 \newcommand*{\@rGls}[2][{}]{%
5048   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5049 }

```

\@rGls@

```

5050 \def\@rGls@#1#2[#3]{%
5051   \glxtrifrecordtrigger{#2}%
5052   {%
5053     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5054   }%
5055   {%
5056     \@Gls@{#1}{#2}[#3]%
5057   }%
5058 }%

```

```

\rGlspl
5059 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

\@rGlspl
5060 \newcommand*{\@rGlspl}[2][\%
5061   \new@ifnextchar[\@rGlspl@{#1}{#2}]{\@rGlspl@{#1}{#2}[]}%
5062 }

\@rGlspl@
5063 \def\@rGlspl@#1#2[#3]{%
5064   \glsxtrifrecordtrigger{#2}%
5065   {%
5066     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5067   }%
5068   {%
5069     \@Glspl@{#1}{#2}[#3]%
5070   }%
5071 }%

\rGLS
5072 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

\@rGLS
5073 \newcommand*{\@rGLS}[2][\%
5074   \new@ifnextchar[\@rGLS@{#1}{#2}]{\@rGLS@{#1}{#2}[]}%
5075 }

\@rGLS@
5076 \def\@rGLS@#1#2[#3]{%
5077   \glsxtrifrecordtrigger{#2}%
5078   {%
5079     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
5080   }%
5081   {%
5082     \@GLS@{#1}{#2}[#3]%
5083   }%
5084 }%

\rGLSpl
5085 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

\@rGLSpl
5086 \newcommand*{\@rGLSpl}[2][\%
5087   \new@ifnextchar[\@rGLSpl@{#1}{#2}]{\@rGLSpl@{#1}{#2}[]}%
5088 }

```

```

\@rGLSpl@
5089 \def \@rGLSpl@#1#2[#3]{%
5090   \glxtrifrecordtrigger{#2}%
5091   {%
5092     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5093   }%
5094   {%
5095     \@GLSpl@{#1}{#2}[#3]%
5096   }%
5097 }%

\rglsformat
5098 \newcommand*{\rglsformat}[2]{%
5099   \glsifregular{#1}
5100   {\glsentryfirst{#1}}%
5101   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5102 }

\rglsplformat
5103 \newcommand*{\rglsplformat}[2]{%
5104   \glsifregular{#1}
5105   {\glsentryfirstplural{#1}}%
5106   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5107 }

\rGlsformat
5108 \newcommand*{\rGlsformat}[2]{%
5109   \glsifregular{#1}
5110   {\Glsentryfirst{#1}}%
5111   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5112 }

\rGlsplformat
5113 \newcommand*{\rGlsplformat}[2]{%
5114   \glsifregular{#1}
5115   {\Glsentryfirstplural{#1}}%
5116   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5117 }

\rGLSformat
5118 \newcommand*{\rGLSformat}[2]{%
5119   \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
5120 }

\rGLSplformat
5121 \newcommand*{\rGLSplformat}[2]{%
5122   \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
5123 }

```

## 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`.

```
5124 \newcommand{\@glsxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
5125 \glsifattribute{\glslabel}{linkcount}{true}%
```

```
5126 {%
```

Does the counter exist?

```
5127 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
```

```
5128 {%
```

Counter doesn’t exist, so define it.

```
5129 \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
5130 \glshasattribute{\glslabel}{linkcountmaster}%
```

```
5131 {%
```

Need to ensure values are fully expanded.

```
5132 \begingroup
```

```
5133 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
```

```
5134 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
```

```
5135 \x
```

```
5136 }%
```

```
5137 {}%
```

```
5138 }%
```

Increment counter:

```
5139 \glsxtrinlinkcounter{glsxtr@linkcount@\glslabel}%
```

```
5140 }%
```

```
5141 {}%
```

```
5142 }
```

`\rinc@linkcounter` May be redefined to use `\refstepcounter` if required.

```
5143 \newcommand*{\glsxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
5144 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
```

```
5145 \ifcsundef{c@glsxtr@linkcount@#1}{0}{\csname c@glsxtr@linkcount@#1\endcsname}%
```

```
5146 }
```

`rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5147 \newcommand*\GlsXtrTheLinkCounter}[1]{%
5148 \ifcsundef{theglsxtr@linkcount@#1}{0}%
5149 {\csname theglsextr@linkcount@#1\endcsname}%
5150 }
```

`fLinkCounterDef` Tests if the counter has been defined

```
5151 \newcommand*\GlsXtrIfLinkCounterDef}[3]{%
5152 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5153 }
```

`LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
5154 \newcommand*\GlsXtrLinkCounterName}[1]{glsextr@linkcount@#1}
```

`ableLinkCounting` `\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
5155 \newcommand*\GlsXtrEnableLinkCounting}[2][{}]{%
5156 \let\glsextr@inc@linkcount\@glsextr@do@inc@linkcount
5157 \@for\@glsextr@label:=#2\do
5158 {%
5159 \glsssetcategoryattribute{\@glsextr@label}{linkcount}{true}%
5160 \ifstrempy{#1}{}%
5161 {%
5162 \ifcsundef{c@#1}%
5163 {\@nocounterr{#1}}%
5164 {\glsssetcategoryattribute{\@glsextr@label}{linkcountmaster}{#1}}%
5165 }%
5166 }%
5167 }
5168 \@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.

```
5169 \@ifpackageloaded{glossaries-accsupp}
5170 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5171 \newcommand*\glsaccessname}[1]{%
```



```

5172   \glsnameaccessdisplay
5173   {%
5174       \glstryname{#1}%
5175   }%
5176   {#1}%
5177 }

```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

5178   \newcommand*{\Glsaccessname}[1]{%
5179       \glsnameaccessdisplay
5180       {%
5181           \Glsentryname{#1}%
5182       }%
5183       {#1}%
5184   }

```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```

5185   \newcommand*{\GLSaccessname}[1]{%
5186       \glsnameaccessdisplay
5187       {%
5188           \mfirstucMakeUppercase{\glstryname{#1}}%
5189       }%
5190       {#1}%
5191   }

```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

5192   \newcommand*{\glsaccesstext}[1]{%
5193       \glstextaccessdisplay
5194       {%
5195           \glstrytext{#1}%
5196       }%
5197       {#1}%
5198   }

```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

5199   \newcommand*{\Glsaccesstext}[1]{%
5200       \glstextaccessdisplay
5201       {%
5202           \Glsentrytext{#1}%
5203       }%
5204       {#1}%
5205   }

```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5206   \newcommand*{\GLSaccesstext}[1]{%
5207       \glstextaccessdisplay

```

```

5208    {%
5209        \mfirstucMakeUppercase{\glentrytext{#1}}}%
5210    }%
5211    {#1}%
5212 }

```

**glsaccessplural** Display the plural value (no link and no check for existence).

```

5213 \newcommand*{\glsaccessplural}[1]{%
5214     \glspluralaccessdisplay
5215     {%
5216         \glentryplural{#1}%
5217     }%
5218     {#1}%
5219 }

```

**Glsaccessplural** Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5220 \newcommand*{\Glsaccessplural}[1]{%
5221     \glspluralaccessdisplay
5222     {%
5223         \Glentryplural{#1}%
5224     }%
5225     {#1}%
5226 }

```

**GLSaccessplural** Display the plural value (no link and no check for existence) converted to upper case.

```

5227 \newcommand*{\GLSaccessplural}[1]{%
5228     \glspluralaccessdisplay
5229     {%
5230         \mfirstucMakeUppercase{\glentryplural{#1}}}%
5231     }%
5232     {#1}%
5233 }

```

**\glsaccessfirst** Display the first value (no link and no check for existence).

```

5234 \newcommand*{\glsaccessfirst}[1]{%
5235     \glsfirstaccessdisplay
5236     {%
5237         \glentryfirst{#1}%
5238     }%
5239     {#1}%
5240 }

```

**\Glsaccessfirst** Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5241 \newcommand*{\Glsaccessfirst}[1]{%
5242     \glsfirstaccessdisplay
5243     {%

```

```

5244     \Glentryfirst{#1}%
5245 }%
5246 {#1}%
5247 }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5248 \newcommand*{\GLSaccessfirst}[1]{%
5249   \glsfirstaccessdisplay
5250   {%
5251     \mfirstucMakeUppercase{\glentryfirst{#1}}%
5252   }%
5253   {#1}%
5254 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5255 \newcommand*{\glsaccessfirstplural}[1]{%
5256   \glsfirstpluralaccessdisplay
5257   {%
5258     \glentryfirstplural{#1}%
5259   }%
5260   {#1}%
5261 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5262 \newcommand*{\Glsaccessfirstplural}[1]{%
5263   \glsfirstpluralaccessdisplay
5264   {%
5265     \Glentryfirstplural{#1}%
5266   }%
5267   {#1}%
5268 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5269 \newcommand*{\GLSaccessfirstplural}[1]{%
5270   \glsfirstpluralaccessdisplay
5271   {%
5272     \mfirstucMakeUppercase{\glentryfirstplural{#1}}%
5273   }%
5274   {#1}%
5275 }

```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

5276 \newcommand*{\glsaccesssymbol}[1]{%
5277   \glssymbolaccessdisplay
5278   {%
5279     \glentrysymbol{#1}%
5280   }%

```

```

5281     {#1}%
5282 }

```

**GLsaccesssymbol** Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5283 \newcommand*{\GLsaccesssymbol}[1]{%
5284   \glssymbolaccessdisplay
5285   {%
5286     \Glentrysymbol{#1}%
5287   }%
5288   {#1}%
5289 }

```

**GLSaccesssymbol** Display the symbol value (no link and no check for existence) converted to upper case.

```

5290 \newcommand*{\GLSaccesssymbol}[1]{%
5291   \glssymbolaccessdisplay
5292   {%
5293     \mfirstucMakeUppercase{\Glentrysymbol{#1}}%
5294   }%
5295   {#1}%
5296 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence).

```

5297 \newcommand*{\glaccesssymbolplural}[1]{%
5298   \glssymbolpluralaccessdisplay
5299   {%
5300     \glentrysymbolplural{#1}%
5301   }%
5302   {#1}%
5303 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5304 \newcommand*{\GLsaccesssymbolplural}[1]{%
5305   \glssymbolpluralaccessdisplay
5306   {%
5307     \Glentrysymbolplural{#1}%
5308   }%
5309   {#1}%
5310 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5311 \newcommand*{\GLSaccesssymbolplural}[1]{%
5312   \glssymbolpluralaccessdisplay
5313   {%
5314     \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5315   }%
5316   {#1}%
5317 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```
5318 \newcommand*{\glsaccessdesc}[1]{%
5319   \glsdescriptionaccessdisplay
5320   {%
5321     \glsentrydesc{#1}%
5322   }%
5323   {#1}%
5324 }
```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
5325 \newcommand*{\Glsaccessdesc}[1]{%
5326   \glsdescriptionaccessdisplay
5327   {%
5328     \Glsentrydesc{#1}%
5329   }%
5330   {#1}%
5331 }
```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```
5332 \newcommand*{\GLSaccessdesc}[1]{%
5333   \glsdescriptionaccessdisplay
5334   {%
5335     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5336   }%
5337   {#1}%
5338 }
```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```
5339 \newcommand*{\glsaccessdescplural}[1]{%
5340   \glsdescriptionpluralaccessdisplay
5341   {%
5342     \glsentrydescplural{#1}%
5343   }%
5344   {#1}%
5345 }
```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
5346 \newcommand*{\Glsaccessdescplural}[1]{%
5347   \glsdescriptionpluralaccessdisplay
5348   {%
5349     \Glsentrydescplural{#1}%
5350   }%
5351   {#1}%
5352 }
```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5353 \newcommand*{\GLSaccessdescplural}[1]{%
5354   \glsdescriptionpluralaccessdisplay
5355   {%
5356     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5357   }%
5358   {#1}%
5359 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5360 \newcommand*{\glsaccessshort}[1]{%
5361   \glsshortaccessdisplay
5362   {%
5363     \glsentryshort{#1}%
5364   }%
5365   {#1}%
5366 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5367 \newcommand*{\Glsaccessshort}[1]{%
5368   \glsshortaccessdisplay
5369   {%
5370     \Glsentryshort{#1}%
5371   }%
5372   {#1}%
5373 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5374 \newcommand*{\GLSaccessshort}[1]{%
5375   \glsshortaccessdisplay
5376   {%
5377     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5378   }%
5379   {#1}%
5380 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5381 \newcommand*{\lsaccessshortpl}[1]{%
5382   \glsshortpluralaccessdisplay
5383   {%
5384     \glsentryshortpl{#1}%
5385   }%
5386   {#1}%
5387 }

```

`\Lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5388 \newcommand*{\Lsaccessshortpl}[1]{%

```

```

5389 \glsshortpluralaccessdisplay
5390 {%
5391 \Glsentryshortpl{#1}%
5392 }%
5393 {#1}%
5394 }

```

**LSaccessshortpl** Display the shortplural value (no link and no check for existence) converted to upper case.

```

5395 \newcommand*{\LSaccessshortpl}[1]{%
5396 \glsshortpluralaccessdisplay
5397 {%
5398 \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5399 }%
5400 {#1}%
5401 }

```

**\glsaccesslong** Display the long form (no link and no check for existence).

```

5402 \newcommand*{\glsaccesslong}[1]{%
5403 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5404 }

```

**\Glsaccesslong** Display the long form (no link and no check for existence).

```

5405
5406 \newcommand*{\Glsaccesslong}[1]{%
5407 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5408 }

```

**\GLSaccesslong** Display the long value (no link and no check for existence) converted to upper case.

```

5409 \newcommand*{\GLSaccesslong}[1]{%
5410 \glslongaccessdisplay
5411 {%
5412 \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5413 }%
5414 {#1}%
5415 }

```

**glsaccesslongpl** Display the long plural form (no link and no check for existence).

```

5416 \newcommand*{\glsaccesslongpl}[1]{%
5417 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5418 }

```

**Glsaccesslongpl** Display the long plural form (no link and no check for existence).

```

5419
5420 \newcommand*{\Glsaccesslongpl}[1]{%
5421 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5422 }

```

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.

```
5423 \newcommand*{\GLSaccesslongpl}[1]{%
5424   \glslongpluralaccessdisplay
5425   {%
5426     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5427   }%
5428   {#1}%
5429 }
```

Keys for accessibility support.

```
5430 \define@key{glsxtrabrv}{access}{%
5431   \def\@gls@nameaccess{#1}%
5432 }
5433 \define@key{glsxtrabrv}{textaccess}{%
5434   \def\@gls@textaccess{#1}%
5435 }
5436 \define@key{glsxtrabrv}{firstaccess}{%
5437   \def\@gls@firstaccess{#1}%
5438 }
5439 \define@key{glsxtrabrv}{shortaccess}{%
5440   \def\@gls@shortaccess{#1}%
5441 }
5442 \define@key{glsxtrabrv}{shortpluralaccess}{%
5443   \def\@gls@shortaccesspl{#1}%
5444 }
```

@initaccesskeys

```
5445 \newcommand*{\@gls@initaccesskeys}{%
5446   \def\@gls@nameaccess{}%
5447   \def\@gls@textaccess{}%
5448   \def\@gls@firstaccess{}%
5449   \def\@gls@shortaccess{}%
5450   \def\@gls@shortaccesspl{}%
5451 }
```

essattribute@set

`\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}`

```
5452 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5453   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5454   {#2}%
5455   {%
5456     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5457     {#3}%
5458   }
```



```

5458   {%
5459     \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5460     {#2}%
5461     {#3}%
5462   }%
5463 }%
5464 }

```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```

5465   \newcommand{\@gls@setup@default@short@access}[1]{%
    Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.
5466   \ifdefempty\@gls@shortaccess
5467   {%
5468     \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5469     {%
5470       \@glsxtr@insertdots\@gls@shortaccess{#1}%
5471       \eappto\ExtraCustomAbbreviationFields{%
5472         shortaccess={\expandonce\@gls@shortaccess},}%
5473     }%
5474   }%
5475 }%
5476 {}%

```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```

5477   \ifdefempty\@gls@shortaccess
5478   {}%
5479   {%
5480     \ifdefempty\@gls@shortaccesspl
5481     {%
5482       \@gls@ifaccessattribute@set{aposplural}%
5483       {%
5484         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5485           \@gls@shortaccess'\abrvpluralsuffix}%
5486       }%
5487     }%
5488     \@gls@ifaccessattribute@set{noshortplural}%
5489     {%
5490       \let\@gls@shortaccesspl\@gls@shortaccess
5491     }%
5492     {%
5493       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5494         \@gls@shortaccess\abrvpluralsuffix}%
5495     }%
5496   }%
5497   \eappto\ExtraCustomAbbreviationFields{%
5498     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5499 }%
5500 {}%

```

```

5501 }%

    If access key hasn't been set, check if the nameshortaccess attribute has been set.
5502 \ifdefempty\@gls@nameaccess
5503 {%
5504 \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5505 {%

    Do nothing if the shortaccess key hasn't been set.
5506 \ifdefempty\@gls@shortaccess
5507 {}%
5508 {%
5509 \eappto\ExtraCustomAbbreviationFields{%
5510 access={\expandonce\@gls@shortaccess},%
5511 }%
5512 }%
5513 }%
5514 {}%
5515 }%
5516 {}%

    If textaccess key hasn't been set, check if the textshortaccess attribute has been set.
5517 \ifdefempty\@gls@textaccess
5518 {%
5519 \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5520 {%

    Do nothing if the shortaccess key hasn't been set.
5521 \ifdefempty\@gls@shortaccess
5522 {}%
5523 {%
5524 \eappto\ExtraCustomAbbreviationFields{%
5525 textaccess={\expandonce\@gls@shortaccess},%
5526 }%
5527 }%
5528 }%
5529 {}%
5530 }%
5531 {}%

    If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.
5532 \ifdefempty\@gls@firstaccess
5533 {%
5534 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5535 {%

    Do nothing if the shortaccess key hasn't been set.
5536 \ifdefempty\@gls@shortaccess
5537 {}%
5538 {%
5539 \eappto\ExtraCustomAbbreviationFields{%
5540 firstaccess={\expandonce\@gls@shortaccess},%

```

```

5541         }%
5542     }%
5543 }%
5544 {}%
5545 }%
5546 {}%
5547 }

```

End of if accsupp part

```

5548 }
5549 {

```

No accessibility support. Just define these commands to do `\glsentry<xxx>`

```

\glsaccessname  Display the name value (no link and no check for existence).
5550  \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.
5551  \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname  Display the name value (no link and no check for existence). converted to upper case.
5552  \newcommand*{\GLSaccessname}[1]{%
5553  \protect\mfirstucMakeUppercase{\glsentryname{#1}}}

\glsaccesstext  Display the text value (no link and no check for existence).
5554  \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.
5555  \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext  Display the text value (no link and no check for existence). converted to upper case.
5556  \newcommand*{\GLSaccesstext}[1]{%
5557  \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}

glsaccessplural  Display the plural value (no link and no check for existence).
5558  \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.
5559  \newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}

GLSaccessplural  Display the plural value (no link and no check for existence). converted to upper case.
5560  \newcommand*{\GLSaccessplural}[1]{%
5561  \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}

\glsaccessfirst  Display the first value (no link and no check for existence).
5562  \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.

5563 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

5564 `\newcommand*{\GLSaccessfirst}[1]{%`  
5565 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).

5566 `\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.

5567 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

5568 `\newcommand*{\GLSaccessfirstplural}[1]{%`  
5569 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

5570 `\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.

5571 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

5572 `\newcommand*{\GLSaccesssymbol}[1]{%`  
5573 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

5574 `\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.

5575 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.

5576 `\newcommand*{\GLSaccesssymbolplural}[1]{%`  
5577 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).

5578 `\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}`

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.  
5579 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.  
5580 `\newcommand*{\GLSaccessdesc}[1]{%`  
5581 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`ccessdescplural` Display the descplural value (no link and no check for existence).  
5582 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.  
5583 `\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.  
5584 `\newcommand*{\GLSaccessdescplural}[1]{%`  
5585 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).  
5586 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

`\GLSaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).  
5587 `\newcommand*{\GLSaccessshort}[1]{\Glsentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.  
5588 `\newcommand*{\GLSaccessshort}[1]{%`  
5589 `\protect\mfirstucMakeUppercase{\glsentryshort{#1}}}`

`lsaccessshortpl` Display the short plural form (no link and no check for existence).  
5590 `\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}`

`lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).  
5591 `\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.  
5592 `\newcommand*{\GLSaccessshortpl}[1]{%`  
5593 `\protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).  
5594 `\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}`

`\GLSaccesslong` Display the long form (no link and no check for existence).  
5595 `\newcommand*{\GLSaccesslong}[1]{\Glsentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

```

5596 \newcommand*{\GLSaccesslong}[1]{%
5597 \protect\mfirstucMakeUppercase{\glstentrylong{#1}}}
```

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5598 \newcommand*{\glsaccesslongpl}[1]{\glstentrylongpl{#1}}
```

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5599 \newcommand*{\Glsaccesslongpl}[1]{\Glstentrylongpl{#1}}
```

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

```

5600 \newcommand*{\GLSaccesslongpl}[1]{%
5601 \protect\mfirstucMakeUppercase{\glstentrylongpl{#1}}}
```

`@initaccesskeys` This does nothing if there's no accessibility support.

```

5602 \newcommand*{\@gls@initaccesskeys}{}
```

`lt@short@access` This does nothing if there's no accessibility support.

```

5603 \newcommand{\@gls@setup@default@short@access}[1]{}%
    End of else part
5604 }
```

## 1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```

5605 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```

5606 \newcommand{\glsifcategory}[4]{%
5607 \ifglstfield{#1}{category}{#2}{#3}{#4}%
5608 }
```

Categories can have attributes.

`categoryattribute` `\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```

5609 \newcommand*{\glissetcategoryattribute}[3]{%
5610 \csdef{@glsxtr@categoryattr@#1@#2}{#3}%
5611 }
```

categoryattribute

```
\glsggetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5612 \newcommand*{\glsggetcategoryattribute}[2]{%
5613   \csuse{@glsxtr@categoryattr@#1@#2}%
5614 }
```

categoryattribute

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5615 \newcommand*{\glshascategoryattribute}[4]{%
5616   \ifcsvoid{@glsxtr@categoryattr@#1@#2}{#4}{#3}%
5617 }
```

\glsssetattribute

```
\glsssetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5618 \newcommand*{\glsssetattribute}[3]{%
5619   \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5620 }
```

\glsggetattribute

```
\glsggetattribute{<entry label>}{<attribute-label>}
```

Short cut where the category label is obtained from the entry information.

```
5621 \newcommand*{\glsggetattribute}[2]{%
5622   \glsggetcategoryattribute{\glscategory{#1}}{#2}%
5623 }
```

\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.

```
5624 \newcommand*{\glshasattribute}[4]{%
5625   \ifglssentryexists{#1}%
5626   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
```

```

5627  {#4}%
5628 }

```

categoryattribute

```
\glsifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true
part>}{<false part>}
```

True if category has the attribute with the given value.

```

5629 \newcommand{\glsifcategoryattribute}[5]{%
5630   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5631   {#5}%
5632   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5633 }

```

\glsifattribute

```
\glsifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}
```

Short cut to determine if the given entry has a category with the given attribute set.

```

5634 \newcommand{\glsifattribute}[5]{%
5635   \ifglstryexists{#1}%
5636   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5637   {#5}%
5638 }

```

Set attributes for the default general category:

```
5639 \glsssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5640 \glsssetcategoryattribute{acronym}{regular}{true}
```

regularcategory

Convenient shortcut to create add the regular attribute.

```

5641 \newcommand*{\glsssetregularcategory}[1]{%
5642   \glsssetcategoryattribute{#1}{regular}{true}%
5643 }

```

ifregularcategory

```
\glsifregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to true.

```

5644 \newcommand{\glsifregularcategory}[3]{%
5645   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5646 }

```



regularcategory

```
\glsifnotregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5647 \newcommand{\glsifnotregularcategory}[3]{%
5648   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5649 }
```

\glsifregular

```
\glsifregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to true.

```
5650 \newcommand{\glsifregular}[3]{%
5651   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
5652 }
```

\glsifnotregular

```
\glsifnotregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to false.

```
5653 \newcommand{\glsifnotregular}[3]{%
5654   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5655 }
```

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.

```
5656 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5657   \foralllglossaries[#1]{#3}%
5658   {%
5659     \forglsentries[#3]{#4}%
5660     {%
5661       \glsifcategory{#4}{#2}{#5}{}%
5662     }%
5663   }%
5664 }
```

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.

```
5665 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5666   \forallglossaries[#1]{#4}%
5667   {%
5668     \forallsentries[#4]{#5}%
5669     {%
5670       \glsifattribute{#5}{#2}{#3}{#6}{}%
5671     }%
5672   }%
5673 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glstrpostdescription`.

```
5674 \ifdef\newterm
5675 {%
```

`\newterm`

```
5676   \renewcommand*{\newterm}[2][ ]{%
5677     \newglossaryentry{#2}%
5678     {type={index},category=index,name={#2},%
5679      description={\glstrpostdescription\nopostdesc},#1}%
5680   }
```

Indexed terms are regular by default.

```
5681   \glsssetcategoryattribute{index}{regular}{true}
```

trpostdescindex

```
5682   \newcommand*{\glstrpostdescindex}{}
5683 }
5684 {}
```

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`.

```
5685 \ifdef\printsymbols
5686 {%
```

glstrnewsymbol

Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```

5687 \newcommand*{\glxtrnewsymbol}[3][]{%
5688   \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5689 }

```

Symbols are regular by default.

```

5690 \glsssetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

5691 \newcommand*{\glxtrpostdescsymbol}{}

5692 }
5693 {}

```

Similar for the numbers option.

```

5694 \ifdef\printnumbers
5695 {%

```

glxtrnewnumber

```

5696 \ifdef\printnumbers
5697   \newcommand*{\glxtrnewnumber}[3][]{%
5698     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5699   }

```

Numbers are regular by default.

```

5700 \glsssetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

5701 \newcommand*{\glxtrpostdescnumber}{}

5702 }
5703 {}

```

glxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```

5704 \newcommand*{\glxtrsetcategory}[2]{%
5705   \@for\@glxtr@label:=#1\do
5706   {%
5707     \glsfieldxddef{\@glxtr@label}{category}{#2}%
5708   }%
5709 }

```

glxtrsetcategoryforall 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.

```

5710 \newcommand*{\glxtrsetcategoryforall}[2]{%
5711   \forallglossaries[#1]{\@glxtr@type}{%
5712     \forglsentries[\@glxtr@type]{\@glxtr@label}%
5713     {%
5714       \glsfieldxddef{\@glxtr@label}{category}{#2}%

```

```

5715 }%
5716 }%
5717 }

```

trfieldtitlecase `\glxtrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```

5718 \newcommand*{\glxtrfieldtitlecase}[2]{%
5719   \expandafter\glxtrfieldtitlecasesecs\expandafter
5720   {\csname glo@\glsetoklabel{#1}@\#2\endcsname}%
5721 }

```

fieldtitlecasesecs The command used by `\glxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```

5722 \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.

```

5723 \ifpackageloaded{glossaries-accsupp}
5724 {
5725   \renewcommand*{\glossentrydesc}[1]{%
5726     \glsoifexistsorwarn{#1}%
5727     {%
5728       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossdescfont` attribute to determine the font applied.

```

5729   \glshasattribute{#1}{glossdescfont}%
5730   {%
5731     \edef\@glxtr@attrval{\glsetattribute{#1}{glossdescfont}}%
5732     \ifcsdef{\@glxtr@attrval}%
5733     {%
5734       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5735     }%
5736     {%
5737       \GlossariesExtraWarning{Unknown control sequence name
5738         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
5739         for entry ‘#1’. Ignoring}%
5740       \let\@glxtr@glossdescfont\@firstofone
5741     }%
5742   }%
5743   {\let\@glxtr@glossdescfont\@firstofone}%
5744   \glsoifattribute{#1}{glossdesc}{firstuc}%
5745   {%
5746     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%

```

```

5747 }%
5748 {%
5749   \glsifattribute{#1}{glossdesc}{title}%
5750   {%
5751     \@glsxtr@do@titlecaps@warn
5752     \glsdescriptionaccessdisplay
5753     {%
5754       \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5755     }%
5756     {#1}%
5757   }%
5758   {%
5759     \@glsxtr@glossdescfont{\glsaccessdesc{#1}}%
5760   }%
5761 }%
5762 }%
5763 }
5764 }
5765 {
5766   \renewcommand*{\glossentrydesc}[1]{%
5767     \glsdoifexistsorwarn{#1}%
5768     {%
5769       \glssetabbrvfmt{\glscategory{#1}}%
5770       \glsifattribute{#1}{glossdescfont}%
5771       {%
5772         \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5773         \ifcsdef{\@glsxtr@attrval}%
5774         {%
5775           \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
5776         }%
5777         {%
5778           \GlossariesExtraWarning{Unknown control sequence name
5779             '\@glsxtr@attrval' supplied in glossdescfont attribute
5780             for entry '#1'. Ignoring}%
5781           \let\@glsxtr@glossdescfont\@firstofone
5782         }%
5783       }%
5784       {\let\@glsxtr@glossdescfont\@firstofone}%
5785       \glsifattribute{#1}{glossdesc}{firstuc}%
5786       {%
5787         \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5788       }%
5789       {%
5790         \glsifattribute{#1}{glossdesc}{title}%
5791         {%
5792           \@glsxtr@do@titlecaps@warn
5793           \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5794         }%
5795         {%

```

```

5796         \@glsxtr@glossdescfont{\glentrydesc{#1}}%
5797     }%
5798 }%
5799 }%
5800 }
5801 }

```

`\glossentryname` If the `glossname` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5802 \@ifpackageloaded{glossaries-accsupp}
5803 {
5804   \renewcommand*{\glossentryname}[1]{%
5805     \@glsdoifexistsorwarn{#1}%
5806     {%
5807       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

5808   \glshasattribute{#1}{glossnamefont}%
5809   {%
5810     \edef\@glsxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5811     \ifcsdef{\@glsxtr@attrval}%
5812     {%
5813       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5814     }%
5815     {%
5816       \GlossariesExtraWarning{Unknown control sequence name
5817         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
5818         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5819       \let\@glsxtr@glossnamefont\glsnamefont
5820     }%
5821   }%
5822   {\let\@glsxtr@glossnamefont\glsnamefont}%
5823   \glsifattribute{#1}{glossname}{firstuc}%
5824   {%
5825     \glsnameaccessdisplay
5826     {%
5827       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5828     }%
5829     {#1}%
5830   }%
5831   {%
5832     \glsifattribute{#1}{glossname}{title}%
5833     {%
5834       \@glsxtr@do@titlecaps@warn
5835       \glsnameaccessdisplay
5836       {%
5837         \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
5838       }%
5839       {#1}%
5840     }%

```

```

5841      {%
5842      \glsifattribute{#1}{glossname}{uc}%
5843      {%
5844      \glsnameaccessdisplay
5845      {%
5846          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5847          \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5848      }%
5849      {#1}%
5850      }%
5851      {%
5852          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5853          \glsnameaccessdisplay
5854          {%
5855          \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
5856          }%
5857          {#1}%
5858      }%
5859      }%
5860      }%

    Do post-name hook:
5861      \glsxtrpostnamehook{#1}%
5862      }%
5863  }
5864 }
5865 {
5866 \renewcommand*{\glossentryname}[1]{%
5867 \@glsdoifexistsorwarn{#1}%
5868 {%
5869 \glssetabbrvfmt{\glscategory{#1}}%
5870 \glsattribute{#1}{glossnamefont}%
5871 {%
5872 \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5873 \ifcsdef{\@glsxtr@attrval}%
5874 {%
5875 \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5876 }%
5877 {%
5878 \GlossariesExtraWarning{Unknown control sequence name
5879 '\@glsxtr@attrval' supplied in glossnamefont attribute
5880 for entry '#1'. Reverting to default \string\glsnamefont}%
5881 \let\@glsxtr@glossnamefont\glsnamefont
5882 }%
5883 }%
5884 {\let\@glsxtr@glossnamefont\glsnamefont}%
5885 \glsifattribute{#1}{glossname}{firstuc}%
5886 {%

```

```

5887     \@glxtr@glossnamefont{\Glsentryname{#1}}%
5888 }%
5889 {%
5890     \glsifattribute{#1}{glossname}{title}%
5891     {%
5892         \@glxtr@do@titlecaps@warn
5893         \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5894     }%
5895     {%
5896         \glsifattribute{#1}{glossname}{uc}%
5897     }%

```

Hide the label from the upper-casing command.

```

5898     \letcs{\glo@name}{glo@glsetoklabel{#1}@name}%
5899     \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5900 }%
5901 {%

```

This little trick is used by glossaries to allow the user to redefine `\glsnamefont` to use `\makefirstuc`. Support it even though they can now use the `firstuc` attribute.

```

5902     \letcs{\glo@name}{glo@glsetoklabel{#1}@name}%
5903     \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5904 }%
5905 }%
5906 }%

```

Do post-name hook.

```

5907     \glxtrpostnamehook{#1}%
5908 }%
5909 }
5910 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```

5911 \@ifpackageloaded{glossaries-accsupp}
5912 {
5913     \renewcommand*{\Glossentryname}[1]{%
5914         \@glsetoklabel{#1}%
5915     }%
5916     \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

5917     \glshasattribute{#1}{glossnamefont}%
5918     {%
5919         \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5920         \ifcsdef{\@glxtr@attrval}%
5921         {%
5922             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5923         }%
5924         {%
5925             \GlossariesExtraWarning{Unknown control sequence name
5926             '\@glxtr@attrval' supplied in glossnamefont attribute

```



```

5927         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5928         \let\@glxtr@glossnamefont\glsnamefont
5929     }%
5930 }%
5931 {\let\@glxtr@glossnamefont\glsnamefont}%
5932 \glsnameaccessdisplay
5933 {%
5934     \@glxtr@glossnamefont{\Glsentryname{#1}}%
5935 }%
5936 {#1}%

```

Do post-name hook:

```

5937     \glxtrpostnamehook{#1}%
5938 }%
5939 }
5940 }
5941 {
5942     \renewcommand*{\Glossentryname}[1]{%
5943         \@glsdofexistsorwarn{#1}%
5944         {%
5945             \glsetabbrvfmt{\glscategory{#1}}%
5946             \glshasattribute{#1}{glossnamefont}%
5947             {%
5948                 \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5949                 \ifcsdef{\@glxtr@attrval}%
5950                 {%
5951                     \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5952                 }%
5953             }%
5954             \GlossariesExtraWarning{Unknown control sequence name
5955             ‘\@glxtr@attrval’ supplied in glossnamefont attribute
5956             for entry ‘#1’. Reverting to default \string\glsnamefont}%
5957             \let\@glxtr@glossnamefont\glsnamefont
5958         }%
5959     }%
5960     {\let\@glxtr@glossnamefont\glsnamefont}%
5961     \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

5962     \glxtrpostnamehook{#1}%
5963 }%
5964 }
5965 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`\glxtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

5966 \newcommand*{\glxtrpostnamehook}[1]{%

```

```

5967 \let\@glsnumberformat\@glsxtr@defaultnumberformat
5968 \glsxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

5969 \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

5970 \csuse{glsxtrpostname\glscategory{#1}}%
5971 }

```

trapostnamehook

```

5972 \newcommand*{\glsextrapostnamehook}[1]{}%

```

\glsdefpostname Provide a convenient command for defining the post-name hook for the given category.

```

5973 \newcommand*{\glsdefpostname}[2]{%
5974 \csdef{glsxtrpostname#1}{#2}%
5975 }

```

etaccessdisplay

```

5976 \@ifpackageloaded{glossaries-accsupp}
5977 {
5978 \newcommand*{\glsxtr@setaccessdisplay}[1]{%
5979 \ifcsdef{gls#1accessdisplay}%
5980 {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
5981 {%

```

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.

```

5982 \edef\@gls@thisval{#1}%
5983 \@for\@gls@map:=\@gls@keymap\do{%
5984 \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
5985 \ifdefequal{\@this@key}{\@gls@thisval}%
5986 {%
5987 \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
5988 \@endfortrue
5989 }%
5990 }%
5991 }%
5992 \ifcsdef{gls\@gls@thisval accessdisplay}%
5993 {\letcs\@glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
5994 {\let\@glsxtr@accessdisplay\@firstoftwo}%
5995 }%
5996 }
5997 }
5998 {%
5999 \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6000 \let\@glsxtr@accessdisplay\@firstoftwo}
6001 }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```
6002 \newrobustcmd*{\glossentrynameother}[2]{%
6003   \@glsdoifexistsorwarn{#1}%
6004   {%
```

Accessibility support:

```
6005   \glstr@setaccessdisplay{#2}%
```

Set the abbreviation format:

```
6006   \glsssetabbrvfmt{\glscategory{#1}}%
6007   \glshasattribute{#1}{glossnamefont}%
6008   {%
6009     \edef\@glstr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6010     \ifcsdef{\@glstr@attrval}%
6011     {%
6012       \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6013     }%
6014     {%
6015       \GlossariesExtraWarning{Unknown control sequence name
6016         '\@glstr@attrval' supplied in glossnamefont attribute
6017         for entry '#1'. Reverting to default \string\glsnamefont}%
6018       \let\@glstr@glossnamefont\glsnamefont
6019     }%
6020   }%
6021   {\let\@glstr@glossnamefont\glsnamefont}%
6022   \glsifattribute{#1}{glossname}{firstuc}%
6023   {%
6024     \@glstr@accessdisplay
6025     {\@glstr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6026     {#1}%
6027   }%
6028   {%
6029     \glsifattribute{#1}{glossname}{title}%
6030     {%
6031       \@glstr@do@titlecaps@warn
6032       \@glstr@accessdisplay
6033       {\@glstr@glossnamefont{\glstrfieldtitlecase{#1}{#2}}}%
6034       {#1}%
6035     }%
6036     {%
6037       \glsifattribute{#1}{glossname}{uc}%
6038       {%
6039         \letcs{\glo@name}{glo@\glsetoklabel{#1}@#2}%
6040         \@glstr@accessdisplay
6041         {\@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6042         {#1}%
6043       }%
6044       {%
6045         \letcs{\glo@name}{glo@\glsetoklabel{#1}@#2}%
```

```

6046      \@glsxtr@accessdisplay
6047      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}}%
6048      {#1}%
6049      }%
6050      }%
6051      }%

  Do post-name hook.
6052      \glsxtrpostnamehook{#1}%
6053      }%
6054  }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

6055 \newif\if@glsxtr@format@override
6056 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glsnumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6057 \@ifpackageloaded{hyperref}
6058 {
  If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
  don't add it.
6059   \ifHy@hyperindex
6060     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6061       \@glsxtr@format@override true
6062       \appto\theindex{\let\glsnumber\@firstofone}%
6063     }
6064   \else
6065     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6066       \@glsxtr@format@override true
6067       \appto\theindex{\let\glsnumber\hyperpage}%
6068     }
6069   \fi
6070 }
6071 {
6072   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6073     \@glsxtr@format@override true
6074   }
6075 }
6076 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

`doautoindexname`

```

6077 \newcommand*{\glsxtrdoautoindexname}[2]{%
6078   \glsattribute{#1}{#2}%
6079   {%

```

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.

```
6080 \glsxtr@autoindex@setname{#1}%
```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```
6081 \protected@edef\glsxtr@attrval{\glsgetattribute{#1}{#2}}%
6082 \if@glsxtr@format@override

6083 \ifx\glsnumberformat\glsxtr@defaultnumberformat
6084 \else
6085 \let\glsxtr@attrval\glsnumberformat
6086 \fi
6087 \fi
6088 \ifdefstring{\glsxtr@attrval}{true}%
6089 {}%
6090 {\eappto\glo@name{\glsxtr@autoindex@encap\glsxtr@attrval}}%
6091 \expandafter\glsxtrautoindex\expandafter{\glo@name}%
6092 }%
6093 {}%
6094 }
```

glsxtrautoindex

```
6095 \newcommand*{\glsxtrautoindex}{\index}
```

toindex@setname Assign \glo@name for use with indexname attribute.

```
6096 \newcommand*{\glsxtr@autoindex@setname}[1]{%
6097 \protected@edef\glo@name{\glsxtrautoindexentry{#1}}%
6098 \glsxtrautoindexassignsort{\glo@sort}{#1}%
6099 \gls@checkmkidxchars\glo@sort
6100 \glsxtr@autoindex@doextra@esc\glo@sort
6101 \epreto\glo@name{\glo@sort\glsxtr@autoindex@at}%
6102 }
```

autoindexentry Command used for the actual part when auto-indexing.

```
6103 \newcommand*{\glsxtrautoindexentry}[1]{\string\glstryname{#1}}
```

indexassignsort Used to assign the sort value when auto-indexing.

```
6104 \newcommand*{\glsxtrautoindexassignsort}[2]{%
6105 \glsletentryfield{#1}{#2}{sort}%
6106 }
```

dex@doextra@esc

```
6107 \newcommand*{\glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6108 \ifx\glsxtr@autoindex@esc\gls@quotechar
6109 \else
6110 \def\gls@checkedmkidx{}%
6111 \edef\@glsxtr@checkspch{%
```

```

6112 \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
6113 \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
6114 \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6115 \@@glsxtr@checkspch
6116 \let#1\@gls@checkedmkidx\relax
6117 \fi

```

Escape actual character unless it has already been escaped.

```

6118 \ifx\@glsxtr@autoindex@at\@gls@actualchar
6119 \else
6120 \def\@gls@checkedmkidx{}%
6121 \edef\@@glsxtr@checkspch{%
6122 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
6123 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
6124 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6125 \@@glsxtr@checkspch
6126 \let#1\@gls@checkedmkidx\relax
6127 \fi

```

Escape level character unless it has already been escaped.

```

6128 \ifx\@glsxtr@autoindex@level\@gls@levelchar
6129 \else
6130 \def\@gls@checkedmkidx{}%
6131 \edef\@@glsxtr@checkspch{%
6132 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
6133 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
6134 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6135 \@@glsxtr@checkspch
6136 \let#1\@gls@checkedmkidx\relax
6137 \fi

```

Escape encap character unless it has already been escaped.

```

6138 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6139 \else
6140 \def\@gls@checkedmkidx{}%
6141 \edef\@@glsxtr@checkspch{%
6142 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6143 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6144 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6145 \@@glsxtr@checkspch
6146 \let#1\@gls@checkedmkidx\relax
6147 \fi
6148 }

```

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.

```

6149 \newcommand*{\@glsxtr@autoindex@at}{\}

```

trSetActualChar Set the actual character.

```
6150 \newcommand*{\GlsXtrSetActualChar}[1]{%
6151   \gdef\@glstr@autoindex@at{#1}%
6152   \def\@glstr@autoindex@escat##1##2##3\@glstr@endescspch{%
6153     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escat}{##1}{##2}{##3}%
6154   }%
6155 }
6156 \@onlypreamble\GlsXtrSetActualChar
6157 \makeatother
6158 \GlsXtrSetActualChar{@}
6159 \makeatletter
```

autoindex@encap Encap character for use with \index.

```
6160 \newcommand*{\@glstr@autoindex@encap}{}
```

XtrSetEncapChar Set the encap character.

```
6161 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6162   \gdef\@glstr@autoindex@encap{#1}%
6163   \def\@glstr@autoindex@escencap##1##2##3\@glstr@endescspch{%
6164     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escencap}{##1}{##2}{##3}%
6165   }%
6166 }
6167 \GlsXtrSetEncapChar{||}
6168 \@onlypreamble\GlsXtrSetEncapChar
```

autoindex@level Level character for use with \index.

```
6169 \newcommand*{\@glstr@autoindex@level}{}
```

XtrSetLevelChar Set the encap character.

```
6170 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6171   \gdef\@glstr@autoindex@level{#1}%
6172   \def\@glstr@autoindex@esclevel##1##2##3\@glstr@endescspch{%
6173     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@esclevel}{##1}{##2}{##3}%
6174   }%
6175 }
6176 \GlsXtrSetLevelChar{!}
6177 \@onlypreamble\GlsXtrSetLevelChar
```

r@autoindex@esc Escape character for use with \index.

```
6178 \newcommand*{\@glstr@autoindex@esc}{""}
```

lsXtrSetEscChar Set the escape character.

```
6179 \newcommand*{\GlsXtrSetEscChar}[1]{%
6180   \gdef\@glstr@autoindex@esc{#1}%
6181   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
6182     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escquote}{##1}{##2}{##3}%
6183   }%
6184 }
```

```

6185 \GlsXtrSetEscChar{}
6186 \@onlypreamble\GlsXtrSetEscChar

```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```

6187 \ifdef\actualchar
6188 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6189 {}

```

Quote character \quotechar:

```

6190 \ifdef\quotechar
6191 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6192 {}

```

Level character \levelchar:

```

6193 \ifdef\levelchar
6194 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6195 {}

```

Encap character \encapchar:

```

6196 \ifdef\encapchar
6197 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6198 {}

```

leto@endescspch

```

6199 \def\@glxtr@gobbleto@endescspch#1\@glxtr@endescspch{}

```

toindex@esc@spch

```

\@glxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}

```

```

6200 \newcommand*{\@glxtr@autoindex@escspch}[5]{%
6201   \@glstmpb=\expandafter{\@glstmpb\the\@glstmpb}%
6202   \toks@={#3}%
6203   \ifx\@nnil#3\relax
6204     \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch#5\@glxtr@endescspch}%
6205   \else
6206     \ifx\@nnil#4\relax
6207       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
6208       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
6209         #4#5\@glxtr@endescspch}%
6210     \else
6211       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
6212       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
6213         #4#5\@glxtr@endescspch}%
6214     \fi
6215   \fi
6216   \@glxtr@checkspch
6217 }

```



`\Glossentrydesc` Redefine to set the abbreviation format and accessibility support.

```
6218 \renewcommand*{\Glossentrydesc}[1]{%
6219   \glsdoifexistsorwarn{#1}%
6220   {%
6221     \glssetabbrvfmt{\glscategory{#1}}%
6222     \Glsaccessdesc{#1}%
6223   }%
6224 }
```

`\lossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```
6225 \renewcommand*{\lossentrysymbol}[1]{%
6226   \glsdoifexistsorwarn{#1}%
6227   {%
6228     \glssetabbrvfmt{\glscategory{#1}}%
6229     \Glsaccesssymbol{#1}%
6230   }%
6231 }
```

`\lossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```
6232 \renewcommand*{\lossentrysymbol}[1]{%
6233   \glsdoifexistsorwarn{#1}%
6234   {%
6235     \glssetabbrvfmt{\glscategory{#1}}%
6236     \Glsaccesssymbol{#1}%
6237   }%
6238 }
```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\eInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```
6239 \newcommand*{\GlsXtrEnableInitialTagging}{%
6240   \@ifstar\s@glsextr@enabletagging\@glsextr@enabletagging
6241 }
6242 \@onlypreamble\GlsXtrEnableInitialTagging
```

`\r@enabletagging` Starred version undefines command.

```
6243 \newcommand*{\s@glsextr@enabletagging}[2]{%
6244   \undef#2%
6245   \@glsextr@enabletagging{#1}{#2}%
6246 }
```

`\r@enabletagging` Internal command.

```
6247 \newcommand*{\@glsextr@enabletagging}[2]{%
  Set attributes for categories given in the first argument.
6248   \@for\@glsextr@cat:=#1\do
6249   {%
```

```

6250 \ifdefempty\@glsxtr@cat
6251 {}%
6252 {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
6253 }%
6254 \newrobustcmd*#2[1]{##1}%
6255 \def\@glsxtr@taggingcs{#2}%
6256 \renewcommand*\@glsxtr@activate@initialtagging{%
6257 \let#2\@glsxtr@tag
6258 }%
6259 \ifundef\@gls@preglossaryhook
6260 {\GlossariesExtraWarning{Initial tagging requires at least
6261 glossaries.sty v4.19 to work correctly}}%
6262 {}%
6263 }

```

Are we using an old version of mfirstuc that has a bug in \capitalisewords? If so, patch it so we don't have a problem with a combination of tagging and title case.

`\mfu@checkword@do` If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```

6264 \ifundef\mfu@checkword@do
6265 {
6266 \newcommand*{\mfu@checkword@do}[1]{%
6267 \ifdefstring{\mfu@checkword@arg}{#1}%
6268 {%
6269 \let\@mfu@domakefirstuc\@firstofone
6270 \listbreak
6271 }%
6272 }%
6273 }

```

`\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.

```

6274 \ifundef\mfu@checkword
6275 {
6276 \newcommand{\@glsxtr@do@titlecaps@warn}{%
6277 \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6278 support not available}%

```

One warning should suffice.

```

6279 \let\@glsxtr@do@titlecaps@warn\relax
6280 }
6281 }
6282 {
6283 \renewcommand*{\mfu@checkword}[1]{%
6284 \def\mfu@checkword@arg{#1}%
6285 \let\@mfu@domakefirstuc\makefirstuc
6286 \forlistloop\mfu@checkword@do\@mfu@nocaplist
6287 }
6288 }
6289 }

```

```

6290 {}% no patch required

@titlecaps@warn Do warning if title case not supported.
6291 \newcommand*{\@glxtr@do@titlecaps@warn}{}

@initialtagging Used in \printglossary but at least v4.19 of glossaries required.
6292 \newcommand*\@glxtr@activate@initialtagging{}

\@glxtr@tag Definition of tagging command when used in glossary.
6293 \newrobustcmd*{\@glxtr@tag}[1]{%
6294   \glsifattribute{\glscurrententrylabel}{tagging}{true}%
6295   {\glxtrtagfont{#1}}{#1}%
6296 }

\glxtrtagfont Used in the glossary.
6297 \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.
6298 \ifdef\gls@preglossaryhook
6299 {
6300   \renewcommand*\@gls@preglossaryhook{%
6301     \@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.

6302     \ifundef\@glxtr@org@postdescription
6303     {%
6304       \let\@glxtr@org@postdescription\glspostdescription
6305       \renewcommand*\glspostdescription{%
6306         \ifglentryexists{\glscurrententrylabel}%
6307         {%
6308           \glxtrpostdescription
6309           \@glxtr@org@postdescription
6310         }%
6311         {}%
6312       }%
6313     }%
6314     {}%

    Enable the options used by \@glxtrp:
6315     \glossxtrsetopts
6316   }%
6317 }
6318 {}

```

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.

```
6319 \newcommand*{\glsxtrpostdescription}{%
6320   \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6321 }
```

postdescgeneral

```
6322 \newcommand*{\glsxtrpostdescgeneral}{}%
```

xtrpostdescterm

```
6323 \newcommand*{\glsxtrpostdescterm}{}%
```

postdescacronym

```
6324 \newcommand*{\glsxtrpostdescacronym}{}%
```

descabbreviation

```
6325 \newcommand*{\glsxtrpostdescabbreviation}{}%
```

`\glsdefpostdesc` Provide a convenient command for defining the post-description hook for the given category.

```
6326 \newcommand*{\glsdefpostdesc}[2]{%
6327   \csdef{glsxtrpostdesc#1}{#2}%
6328 }
```

`\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.

```
6329 \renewcommand*{\glspostlinkhook}{%
6330   \ifglentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6331 }
```

xtrpostlinkhook The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
6332 \newcommand*{\glsxtrpostlinkhook}{%
6333   \glsxtrdiscardperiod{\glslabel}%
6334   {\glsxtrpostlinkendsentence}%
6335   {\glsxtrifcustomdiscardperiod
6336     {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
6337   {\glsxtrpostlink}%
6338 }%
6339 }
```

omdiscardperiod Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6340 \newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

`\glstrpostlink`

```
6341 \newcommand*{\glstrpostlink}{%
6342   \csuse{\glstrpostlink\glscategory{\glslabel}}%
6343 }
```

`\glstdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glstrpostlink`.

```
6344 \newcommand*{\glstdefpostlink}[2]{%
    \ifthenelse is used to ensure that the expanded value is tested. (The category label must
    be fully expandable.)
6345   \ifthenelse{\equal{#1}{}}{%
6346     {\PackageError{glossaries-extra}
6347       {Invalid empty category label in \string\glstdefpostlink}{}}%
6348     {\csdef{\glstrpostlink#1}{#2}}%
6349 }
```

`linkendsentence` Done by `\glstrpostlinkhook` if a full stop is discarded.

```
6350 \newcommand*{\glstrpostlinkendsentence}{%
6351   \ifcsdef{\glstrpostlink\glscategory{\glslabel}}
6352   {%
6353     \csuse{\glstrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
6354     .\spacefactor\sfcode'\. \relax
6355   }%
6356   {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
6357   \spacefactor\sfcode'\. \relax
6358 }%
6359 }
```

`addDescOnFirstUse` 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.

```
6360 \newcommand*{\glstrpostlinkAddDescOnFirstUse}{%
6361   \glstrifwasfirstuse{\space\glstrparen{\glssaccessdesc{\glslabel}}}{}%
6362 }
```

`symbolOnFirstUse` Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6363 \newcommand*{\glstrpostlinkAddSymbolOnFirstUse}{%
6364   \glstrifwasfirstuse
6365   {%
6366     \ifglshassymbol{\glslabel}%
6367     {\space\glstrparen{\glssaccesssymbol{\glslabel}}}%
6368     {}%
6369   }%
```

```

6370 {}%
6371 }

```

**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.

```

6372 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
6373   \glxtrifwasfirstuse
6374   {%
6375     \space\glxtrparen
6376     {%
6377       \ifglshassymbol{\glslabel}%
6378       {\glssaccesssymbol{\glslabel}, }%
6379       }%
6380       \glssaccessdesc{\glslabel}%
6381     }%
6382   }%
6383 {}%
6384 }

```

**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).

```

6385 \newcommand*{\glxtrdiscardperiod}[3]{%
6386   \glxtrifwasfirstuse
6387   {%
6388     \glusifattribute{#1}{retainfirstuseperiod}{true}%
6389     {#3}%
6390     {%
6391       \glusifattribute{#1}{discardperiod}{true}%
6392       {%
6393         \glusifplural
6394         {%
6395           \glusifattribute{#1}{pluraldiscardperiod}{true}%
6396           {\glxtrifperiod{#2}{#3}}%
6397           {#3}%
6398         }%
6399       }%
6400       \glxtrifperiod{#2}{#3}%
6401     }%
6402   }%
6403   {#3}%
6404 }%
6405 }%
6406 {%
6407   \glusifattribute{#1}{discardperiod}{true}%
6408   {%
6409     \glusifplural
6410     {%

```

```

6411 \glsifattribute{#1}{pluraldiscardperiod}{true}%
6412 {\glsxtrifperiod{#2}{#3}}%
6413 {#3}%
6414 }%
6415 {%
6416 \glsxtrifperiod{#2}{#3}%
6417 }%
6418 }%
6419 {#3}%
6420 }%
6421 }

```

`\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`

```

6422 \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 ' ').

```

6423 \newcommand*\glsxtr@punclist{.,:;?!}

```

`punctuationmark` Add character to punctuation list.

```

6424 \newcommand*\glsxtraddpunctuationmark[1]{\appto\glsxtr@punclist{#1}}

```

`unctuationmarks` Reset the punctuation list.

```

6425 \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`.)

```

6426 \newcommand*\glsxtrifnextpunc[2]{%
6427 \def\reserved@a{#1}%
6428 \def\reserved@b{#2}%
6429 \futurelet\@glspunc@token\glsxtr@ifnextpunc
6430 }

```

`sxtr@ifnextpunc`

```

6431 \newcommand*\glsxtr@ifnextpunc{%
6432 \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
6433 \reserved@b
6434 }

```

`xtr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```

6435 \newcommand*\glsxtr@ifpunctoken[1]{%
6436 \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
6437 }

```

xtr@ifpunctoken

```
6438 \def\@glxtr@ifpunctoken#1#2{%
6439   \let\reserved@d=#2%
6440   \ifx\reserved@d\@nnil
6441     \let\glxtr@next\@glxtr@notfoundinlist
6442   \else
6443     \ifx#1\reserved@d
6444       \let\glxtr@next\@glxtr@foundinlist
6445     \else
6446       \let\glxtr@next\@glxtr@ifpunctoken
6447     \fi
6448   \fi
6449   \glxtr@next#1%
6450 }
```

xtr@foundinlist

```
6451 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
6452 \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.

```
6453 \newcommand{\glxtrdopostpunc}[1]{%
6454   \glxtr@ifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6455 }
```

@glxtr@swaptwo

```
6456 \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.

```
6457 \define@key{glxtrabbrv}{category}{%
6458   \edef\glscategorylabel{#1}%
6459   \ifcsdef{@glxabbrv@current@#1}%
6460   {%
```



Warning should already have been issued.

```

6461 \let\@glxstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6462 \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6463 \glxstr@applyabbrvstyle{\csname @glxabbrv@current@#1\endcsname}%
6464 \let\GlsXtrWarnDeprecatedAbbrStyle\@glxstr@orgwarndep
6465 }%
6466 {}%
6467 }

```

Save the short plural form. This may be needed before the entry is defined.

```

6468 \define@key{glxstrabbrv}{shortplural}{%
6469 \def\@gls@shortpl{#1}%
6470 }

```

Similarly for the long plural form.

```

6471 \define@key{glxstrabbrv}{longplural}{%
6472 \def\@gls@longpl{#1}%
6473 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

```

\glsshortpltok
6474 \newtoks\glsshortpltok

\glslongpltok
6475 \newtoks\glslongpltok

```

`\glxstr@insertdots` Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to `\newabbreviation`. Note that explicitly using the short or shortplural keys will override this.

```

6476 \newcommand*{\@glxstr@insertdots}[2]{%
6477 \def#1{%
6478 \@glxstr@insert@dots#1#2\@nnil
6479 }

```

```

\glxstr@insert@dots
6480 \newcommand*{\@glxstr@insert@dots}[2]{%
6481 \ifx\@nnil#2\relax
6482 \let\@glxstr@insert@dots@next\@gobble
6483 \else
6484 \ifx\relax#2\relax
6485 \else
6486 \appto#1{#2.}%
6487 \fi
6488 \let\@glxstr@insert@dots@next\@glxstr@insert@dots

```

```

6489 \fi
6490 \@glsxtr@insert@dots@next#1%
6491 }

```

Similarly provide a way of replacing spaces with `\glsxtrwordsep`, which first needs to be defined:

`\glsxtrwordsep`

```

6492 \newcommand*{\glsxtrwordsep}{\space}

```

Each word is marked with

`\glsxtrword`

```

6493 \newcommand*{\glsxtrword}[1]{#1}

```

`tr@markwordseps`

```

6494 \newcommand*{\@glsxtr@markwordseps}[2]{%
6495   \def#1{}%
6496   \@glsxtr@mark@wordseps#1#2 \@nnil
6497 }

```

`r@mark@wordseps`

```

6498 \def\@glsxtr@mark@wordseps#1#2 #3{%
6499   \ifdefempty{#1}%
6500   {\def#1{\protect\glsxtrword{#2}}}%
6501   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
6502   \ifx\@nnil#3\relax
6503     \let\@glsxtr@mark@wordseps@next\relax
6504   \else
6505     \def\@glsxtr@mark@wordseps@next{%
6506       \@glsxtr@mark@wordseps#1#3}%
6507   \fi
6508   \@glsxtr@mark@wordseps@next
6509 }

```

`newabbreviation` Define a new generic abbreviation.

```

6510 \newcommand*{\newabbreviation}[4][ ]{%
6511   \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
6512 }

```

`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.)

```

6513 \newcommand*{\glsxtr@newabbreviation}[4]{%
6514   \glskeylisttok{#1}%
6515   \glslabeltok{#2}%
6516   \glsshorttok{#3}%
6517   \glslongtok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```
6518 \def\glstrorgshort{#3}%
6519 \def\glstrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
6520 \def\ExtraCustomAbbreviationFields{}
```

Initialise accessibility settings if required.

```
6521 \@gls@initaccesskeys
```

Get the category.

```
6522 \def\glscategorylabel{abbreviation}%
6523 \glstr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
6524 \setkeys*{glstrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
6525 \def\@gls@longpl{#4\glspluralsuffix}%
6526 \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
6527 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6528 {%
6529   \@glstr@markwordseps\@gls@long{#4}%
6530   \expandafter\def\expandafter\@gls@longpl\expandafter
6531     {\@gls@long\glspluralsuffix}%
6532   \let\@gls@default@longpl\@gls@longpl
```

Update \glslongtok.

```
6533   \expandafter\glslongtok\expandafter{\@gls@long}%
6534 }%
6535 {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6536 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6537 {%
6538   \@glstr@markwordseps\@gls@short{#3}%
6539 }%
6540 {}%
```

Has the insertdots attribute been set?

```
6541 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6542 {%
6543   \@glstr@insertdots\@gls@short{#3}%
6544   \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6545 }%
6546 {\def\@gls@short{#3}}%
6547 }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6548 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6549 {%
```

```

6550 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6551 \abbrvpluralsuffix}%
6552 }%
6553 {%

Has the noshortplural attribute been set?
6554 \glsifcategoryattribute{\gls@categorylabel}{noshortplural}{true}%
6555 {%
6556 \let\@gls@shortpl\@gls@short
6557 }%
6558 {%
6559 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6560 \abbrvpluralsuffix}%
6561 }%
6562 }%

Update \glsshorttok:
6563 \expandafter\glsshorttok\expandafter{\@gls@short}%

Hook for further customisation if required:
6564 \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%

Get the short and long plurals provided by user in optional argument to override defaults, if
necessary. Ignore the category key (already obtained).
6565 \setkeys*{\glsxtrabbrv}[category]{#1}%

Has the plural been explicitly set?
6566 \ifx\@gls@default@longpl\@gls@longpl
6567 \else

Has the markwords attribute been set?
6568 \glsifcategoryattribute{\gls@categorylabel}{markwords}{true}%
6569 {%
6570 \expandafter\@glxtr@keywordseps\expandafter\@gls@longpl\expandafter
6571 {\@gls@longpl}%
6572 }%
6573 {}%
6574 \fi

Set the plural token registers so the values can be accessed by the abbreviation styles.
6575 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6576 \expandafter\glslongpltok\expandafter{\@gls@longpl}%

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).
6577 \@gls@setup@default@short@access{#3}%

Do any extra setup provided by hook:
6578 \newabbreviationhook

Define this entry:
6579 \protected@edef\@do@newglossaryentry{%
6580 \noexpand\newglossaryentry{\the\gls@labeltok}%
6581 {%

```

```

6582     type=\glstrabbrvtype,%
6583     category=abbreviation,%
6584     short={\the\glsshorttok},%
6585     shortplural={\the\glsshortpltok},%
6586     long={\the\glslongtok},%
6587     longplural={\the\glslongpltok},%
6588     name={\the\glsshorttok},%
6589     \CustomAbbreviationFields,%

```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```

6590     \ExtraCustomAbbreviationFields

```

Any explicit fields set in the optional argument override all other settings.

```

6591     \the\glskeylisttok
6592     }%
6593     }%
6594     \@do@newglossaryentry
6595     \GlsXtrPostNewAbbreviation
6596 }

```

**evpresetkeyhook** Hook for extra stuff in `\newabbreviation`

```

6597 \newcommand*{\glstrnewabbspresetkeyhook}[3]{}

```

**NewAbbreviation** Hook used by abbreviation styles.

```

6598 \newcommand*{\GlsXtrPostNewAbbreviation}{}

```

**bbreviationhook** Hook for use with `\newabbreviation`.

```

6599 \newcommand*{\newabbreviationhook}{}

```

**reviationFields**

```

6600 \newcommand*{\CustomAbbreviationFields}{}

```

**\glstrparen** For the parenthetical styles.

```

6601 \newcommand*{\glstrparen}[1]{(#1)}

```

**lsxtrfullformat** Full format without case change.

```

6602 \newcommand*{\glsxtrfullformat}[2]{%
6603   \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
6604   \glstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6605 }

```

**lsxtrfullformat** Full format with case change.

```

6606 \newcommand*{\Glsxtrfullformat}[2]{%
6607   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
6608   \glstrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
6609 }

```

`\glxtrfullplformat` Plural full format without case change.

```
6610 \newcommand*{\glxtrfullplformat}[2]{%
6611   \glsfirstlongfont{\glssaccesslongpl{#1}}#2\glxtrfullsep{#1}%
6612   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{#1}}}%
6613 }
```

`\glxtrfullplformat` Plural full format with case change.

```
6614 \newcommand*{\Glsxtrfullplformat}[2]{%
6615   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
6616   \glxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6617 }
```

`\glxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```
6618 \newcommand*{\glxtrfullsep}[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).

`\glxtrinlinefullformat` Full format without case change.

```
6619 \newcommand*{\glxtrinlinefullformat}{\glxtrfullformat}
```

`\Glsxtrinlinefullformat` Full format with case change.

```
6620 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\glxtrfullplformat` Plural full format without case change.

```
6621 \newcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}
```

`\Glsxtrfullplformat` Plural full format with case change.

```
6622 \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 `\glxtrfull` set of commands instead.

`\gl Sentryfull`

```
6623 \renewcommand*{\gl Sentryfull}[1]{\glxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
6624 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\gl Sentryfullpl`

```
6625 \renewcommand*{\gl Sentryfullpl}[1]{\glxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

```
6626 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\gl sfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

```
6627 \newcommand*{\gl sfirstabbrvfont}[1]{\gl sfirstabbrvdefaultfont{#1}}
```

`bbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

```
6628 \newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}
```

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.

```
6629 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}
```

`bbrvdefaultfont`

```
6630 \newcommand*{\glsabbrvdefaultfont}[1]{#1}
```

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.

```
6631 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

`longdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.

```
6632 \newcommand*{\glslongdefaultfont}[1]{#1}
```

`lsfirstlongfont` Font changing command used for the long form on first use or in the full format.

```
6633 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

`longdefaultfont`

```
6634 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

`brvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

```
6635 \newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}
```

`brvpluralsuffix` Default plural suffix.

```
6636 \newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}
```

`\glsxtrfull` Full form (no case-change).

```
6637 \newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\@ns@glsxtrfull}
6638 \newcommand*{\ns@glsxtrfull}[2][\%]
6639 \new@ifnextchar[{\@glsxtr@full{#1}{#2}}]{\%}
6640 {\@glsxtr@full{#1}{#2}[\%]}
6641 }
```

`\@glsxtr@full` Low-level macro:

```
6642 \def\@glsxtr@full#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).
6643 \@glsxtr@record{#1}{#2}{\glslink}%
6644 \glsdoifexists{#2}%
6645 {%
6646 \glssetabbrvfmt{\glscategory{#2}}%
6647 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6648 \let\glsifplural\@secondoftwo
6649 \let\glsupcase\@firstofthree
6650 \let\glsinsert\@empty
6651 \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%
```

What should `\glxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
6652 \glxtrsetupfulldefs
6653 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6654 }%
6655 \glspostlinkhook
6656 }
```

`trsetupfulldefs`

```
6657 \newcommand*\glxtrsetupfulldefs{%
6658 \let\glxtrifwasfirstuse\@firstoftwo
6659 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
6660 \newrobustcmd*\Glsxtrfull{\@gls@hyp@opt\@ns@Glsxtrfull}
6661 \newcommand*\ns@Glsxtrfull[2][{}]{%
6662 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
6663 {\@Glsxtr@full{#1}{#2}[]}%
6664 }
```

`\@Glsxtr@full` Low-level macro:

```
6665 \def\@Glsxtr@full#1#2[#3]{%
6666 \glsdoifexists{#2}%
6667 {%
6668 \glssetabbrvfmt{\glscategory{#2}}%
6669 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6670 \let\glsifplural\@secondoftwo
6671 \let\glsupcase\@secondofthree
6672 \let\glsinsert\@empty
6673 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6674 \glxtrsetupfulldefs
6675 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6676 }%
6677 \glspostlinkhook
6678 }
```

`\GLSxtrfull` Full form (all uppercase).

```
6679 \newrobustcmd*\GLSxtrfull{\@gls@hyp@opt\@ns@GLSxtrfull}
6680 \newcommand*\ns@GLSxtrfull[2][{}]{%
6681 \new@ifnextchar[\@GLSxtr@full{#1}{#2}}%
6682 {\@GLSxtr@full{#1}{#2}[]}%
6683 }
```

`\@GLSxtr@full` Low-level macro:

```
6684 \def\@GLSxtr@full#1#2[#3]{%
6685 \glsdoifexists{#2}%
6686 }
```



```

6686 {%
6687   \glsetabbrfmt{\glscategory{#2}}%
6688   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6689   \let\gl@ifplural\@secondoftwo
6690   \let\glscapscase\@thirdofthree
6691   \let\glinsert\@empty
6692   \def\glscustomtext{\mfirstucMakeUppercase{\glxtrinlinefullformat{#2}{#3}}}%
6693   \glxtrsetupfulldefs
6694   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6695 }%
6696 \glspostlinkhook
6697 }

```

`\glxtrfullpl` Plural full form (no case-change).

```

6698 \newrobustcmd*{\glxtrfullpl}{\@gl@hyp@opt\@ns@glxtrfullpl}
6699 \newcommand*\ns@glxtrfullpl[2][{}]{%
6700   \new@ifnextchar[{\@glxtr@fullpl{#1}{#2}}%
6701     {\@glxtr@fullpl{#1}{#2}[]}%
6702 }

```

`\@glxtr@fullpl` Low-level macro:

```

6703 \def\@glxtr@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).
6704   \@glxtr@record{#1}{#2}{glslink}%
6705   \gl@ifexists{#2}%
6706   {%
6707     \glsetabbrfmt{\glscategory{#2}}%
6708     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6709     \let\gl@ifplural\@firstoftwo
6710     \let\glscapscase\@firstofthree
6711     \let\glinsert\@empty
6712     \def\glscustomtext{\glxtrinlinefullplformat{#2}{#3}}%
6713     \glxtrsetupfulldefs
6714     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6715   }%
6716   \glspostlinkhook
6717 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

6718 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\@ns@Glsxtrfullpl}
6719 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
6720   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6721     {\@Glsxtr@fullpl{#1}{#2}[]}%
6722 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6723 \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).

```

6724 \@glstr@record{#1}{#2}{glslink}%
6725 \glsoifexists{#2}%
6726 {%
6727 \glsetabbrvfmt{\glscategory{#2}}%
6728 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6729 \let\gl@ifplural\@firstoftwo
6730 \let\glscapscase\@secondofthree
6731 \let\glinsert\@empty
6732 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
6733 \glstrsetupfulldefs
6734 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6735 }%
6736 \glspostlinkhook
6737 }

```

\Glsxtrfullpl Plural full form (all upper case).

```

6738 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\ns@Glsxtrfullpl}
6739 \newcommand*{\ns@Glsxtrfullpl}[2][{}]{%
6740 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6741 {\@Glsxtr@fullpl{#1}{#2}[]}%
6742 }

```

\@Glsxtr@fullpl Low-level macro:

```

6743 \def\@Glsxtr@fullpl#1#2[#3]{%
6744 \@glstr@record{#1}{#2}{glslink}%
6745 \glsoifexists{#2}%
6746 {%
6747 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6748 \let\gl@ifplural\@firstoftwo
6749 \let\glscapscase\@thirdofthree
6750 \let\glinsert\@empty
6751 \def\glscustomtext{%
6752 \mfirstucMakeUppercase{\glxtrinlinefullplformat{#2}{#3}}}%
6753 \glstrsetupfulldefs
6754 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
6755 }%
6756 \glspostlinkhook
6757 }

```

The short and long forms work in a similar way to acronyms.

\glstrshort

```

6758 \newrobustcmd*{\glstrshort}{\@gl@hyp@opt\ns@glstrshort}

```

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

```
6759 \newcommand*{\ns@glstrshort}[2] [] {%
6760   \new@ifnextchar[{\@glstrshort{#1}{#2}}{\@glstrshort{#1}{#2} []}%
6761 }
```

Read in the final optional argument:

```
6762 \def\@glstrshort#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).

```
6763   \@glstr@record{#1}{#2}{glslink}%
6764   \glstoifexists{#2}%
6765   {%
```

Need to make sure \glssabrvfont is set correctly.

```
6766     \glsssetabrvfmt{\glscategory{#2}}%
6767     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6768     \let\glstrifwasfirstuse\@secondoftwo
6769     \let\glslifplural\@secondoftwo
6770     \let\glscapscase\@firstofthree
6771     \let\glslinsert\@empty
6772     \def\glscustomtext{%
6773       \glssabrvfont{\glssaccessshort{#2}\ifglstrinsertinside#3\fi}%
6774       \ifglstrinsertinside\else#3\fi
6775     }%
6776     \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6777   }%
6778   \glspostlinkhook
6779 }
```

\Glsxtrshort

```
6780 \newrobustcmd*{\Glsxtrshort}{\@glshyp@opt\ns@Glsxtrshort}
```

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

```
6781 \newcommand*{\ns@Glsxtrshort}[2] [] {%
6782   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
6783 }
```

Read in the final optional argument:

```
6784 \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).

```
6785   \@glstr@record{#1}{#2}{glslink}%
6786   \glstoifexists{#2}%
6787   {%
6788     \glsssetabrvfmt{\glscategory{#2}}%
6789     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
6790     \let\glstrifwasfirstuse\@secondoftwo
6791     \let\glslifplural\@secondoftwo
6792     \let\glscapscase\@secondofthree
```

```

6793 \let\glsinsert\@empty
6794 \def\glscustomtext{%
6795     \glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6796     \ifglxtrinsertinside\else#3\fi
6797 }%
6798 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6799 }%
6800 \glspostlinkhook
6801 }

```

## \GLSxtrshort

```

6802 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}

Define the un-starred form. Need to determine if there is a final optional argument
6803 \newcommand*{\ns@GLSxtrshort}[2][{}]{%
6804     \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[]}%
6805 }

Read in the final optional argument:
6806 \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).
6807     \@glsxtr@record{#1}{#2}{glslink}%
6808     \glsdoifexists{#2}%
6809     {%
6810         \glssetabbrvfmt{\glscategory{#2}}%
6811         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6812         \let\glsxtrifwasfirstuse\@secondoftwo
6813         \let\glsifplural\@secondoftwo
6814         \let\glscapscase\@thirdofthree
6815         \let\glsinsert\@empty
6816         \def\glscustomtext{%
6817             \mfirstucMakeUppercase
6818             {\glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6819             \ifglxtrinsertinside\else#3\fi
6820         }%
6821     }%
6822     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6823 }%
6824 \glspostlinkhook
6825 }

```

## \glsxtrlong

```

6826 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glsxtrlong}

Define the un-starred form. Need to determine if there is a final optional argument
6827 \newcommand*{\ns@glsxtrlong}[2][{}]{%
6828     \new@ifnextchar[\@glsxtrlong{#1}{#2}]{\@glsxtrlong{#1}{#2}[]}%
6829 }

```

Read in the final optional argument:

```
6830 \def\@glxstrlong#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).

```
6831 \@glxstr@record{#1}{#2}{glslink}%
6832 \glstoifexists{#2}%
6833 {%
6834 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6835 \let\glxstrifwasfirstuse\@secondoftwo
6836 \let\gl@ifplural\@secondoftwo
6837 \let\glscapscase\@firstofthree
6838 \let\glinsert\@empty
6839 \def\glscustomtext{%
6840 \glslongfont{\glaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6841 \ifglxtrininsertinside\else#3\fi
6842 }%
6843 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6844 }%
6845 \glspostlinkhook
6846 }
```

\Glsxtrlong

```
6847 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\@ns@Glsxtrlong}
```

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

```
6848 \newcommand*{\ns@Glsxtrlong}[2][]{%
6849 \new@ifnextchar[\@Glsxtrlong{#1}{#2}]{\@Glsxtrlong{#1}{#2}[]}%
6850 }
```

Read in the final optional argument:

```
6851 \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).

```
6852 \@glxstr@record{#1}{#2}{glslink}%
6853 \glstoifexists{#2}%
6854 {%
6855 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6856 \let\glxstrifwasfirstuse\@secondoftwo
6857 \let\gl@ifplural\@secondoftwo
6858 \let\glscapscase\@secondofthree
6859 \let\glinsert\@empty
6860 \def\glscustomtext{%
6861 \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6862 \ifglxtrininsertinside\else#3\fi
6863 }%
6864 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6865 }%
6866 \glspostlinkhook
6867 }
```

\GLSxtrlong

```
6868 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}
```

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

```
6869 \newcommand*{\ns@GLSxtrlong}[2] [] {%
```

```
6870   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
```

```
6871 }
```

Read in the final optional argument:

```
6872 \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).

```
6873   \@glsxtr@record{#1}{#2}{glslink}%
```

```
6874   \glsdoifexists{#2}%
```

```
6875   {%
```

```
6876     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
6877     \let\glsxtrifwasfirstuse\@secondoftwo
```

```
6878     \let\glsifplural\@secondoftwo
```

```
6879     \let\glscapscase\@thirdofthree
```

```
6880     \let\glsinsert\@empty
```

```
6881     \def\glscustomtext{%
```

```
6882       \mfirstucMakeUppercase
```

```
6883       {\glsfont{\glsaccesslong{#2}\ifglsxtrininsertinside#3\fi}%
```

```
6884       \ifglsxtrininsertinside\else#3\fi
```

```
6885     }%
```

```
6886   }%
```

```
6887   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
6888 }%
```

```
6889 \glspostlinkhook
```

```
6890 }
```

Plural short forms:

\glsxtrshortpl

```
6891 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}
```

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

```
6892 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
```

```
6893   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
```

```
6894 }
```

Read in the final optional argument:

```
6895 \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).

```
6896   \@glsxtr@record{#1}{#2}{glslink}%
```

```
6897   \glsdoifexists{#2}%
```

```
6898   {%
```

```
6899     \glssetabbrvfmt{\glscategory{#2}}%
```

```

6900 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6901 \let\glxtrifwasfirstuse\@secondoftwo
6902 \let\gl@ifplural\@firstoftwo
6903 \let\gl@scapscase\@firstofthree
6904 \let\gl@insert\@empty
6905 \def\glscustomtext{%
6906     \gl@abbrvfont{\gl@accessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6907     \ifglxtrininsertinside\else#3\fi
6908 }%
6909 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6910 }%
6911 \glspostlinkhook
6912 }

```

#### \Glsxtrshortpl

```

6913 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\ns@Glsxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6914 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
6915     \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
6916 }

```

Read in the final optional argument:

```

6917 \def\@Glsxtrshortpl#1#2[#3] {%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6918 \@glxtr@record{#1}{#2}{gl@link}%
6919 \gl@doifexists{#2}%
6920 {%
6921     \gl@setabbrvfmt{\gl@category{#2}}%
6922     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6923     \let\glxtrifwasfirstuse\@secondoftwo
6924     \let\gl@ifplural\@firstoftwo
6925     \let\gl@scapscase\@secondofthree
6926     \let\gl@insert\@empty
6927     \def\glscustomtext{%
6928         \gl@abbrvfont{\Gls@accessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6929         \ifglxtrininsertinside\else#3\fi
6930     }%
6931     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6932 }%
6933 \glspostlinkhook
6934 }

```

#### \GLSxtrshortpl

```

6935 \newrobustcmd*{\GLSxtrshortpl}{\@gl@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6936 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%

```

```

6937 \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6938 }

```

Read in the final optional argument:

```

6939 \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).

```

6940 \@glxtr@record{#1}{#2}{glslink}%
6941 \glstoifexists{#2}%
6942 {%
6943 \glsetabbrvfmt{\glscategory{#2}}%
6944 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6945 \let\glxtrifwasfirstuse\@secondoftwo
6946 \let\gl@ifplural\@firstoftwo
6947 \let\glscapscase\@thirdofthree
6948 \let\glinsert\@empty
6949 \def\glscustomtext{%
6950 \mfirstucMakeUppercase
6951 {\glabbrvfont{\glaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6952 \ifglxtrininsertinside\else#3\fi
6953 }%
6954 }%
6955 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6956 }%
6957 \glspostlinkhook
6958 }

```

Plural long forms:

\glxtrlongpl

```

6959 \newrobustcmd*{\glxtrlongpl}{\@gl@hyp@opt\@ns@glxtrlongpl}

```

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

```

6960 \newcommand*{\ns@glxtrlongpl}[2] [] {%
6961 \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
6962 }

```

Read in the final optional argument:

```

6963 \def\@glxtrlongpl#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).

```

6964 \@glxtr@record{#1}{#2}{glslink}%
6965 \glstoifexists{#2}%
6966 {%
6967 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6968 \let\glxtrifwasfirstuse\@secondoftwo
6969 \let\gl@ifplural\@firstoftwo
6970 \let\glscapscase\@firstofthree
6971 \let\glinsert\@empty

```



```

6972 \def\glscustomtext{%
6973 \glslongfont{\glssaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
6974 \ifglsxtrinsertinside\else#3\fi
6975 }%
6976 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6977 }%
6978 \glspostlinkhook
6979 }

```

\Glsxtrlongpl

```

6980 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6981 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
6982 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
6983 }

```

Read in the final optional argument:

```

6984 \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).
6985 \@glsxtr@record{#1}{#2}{glslink}%
6986 \glsdoidexists{#2}%
6987 {%
6988 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6989 \let\glxtrifwasfirstuse\@secondoftwo
6990 \let\glsifplural\@firstoftwo
6991 \let\glscapscase\@secondofthree
6992 \let\glsinsert\@empty
6993 \def\glscustomtext{%
6994 \glslongfont{\glssaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
6995 \ifglsxtrinsertinside\else#3\fi
6996 }%
6997 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6998 }%
6999 \glspostlinkhook
7000 }

```

\GLSxtrlongpl

```

7001 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7002 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7003 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7004 }
    Read in the final optional argument:
7005 \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).

```

7006 \@glstr@record{#1}{#2}{glslink}%
7007 \glsoifexists{#2}%
7008 {%
7009 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7010 \let\glstrifwasfirstuse\@secondoftwo
7011 \let\gl@ifplural\@firstoftwo
7012 \let\glscapscase\@thirdofthree
7013 \let\glinsert\@empty
7014 \def\glscustomtext{%
7015 \mfirstucMakeUppercase
7016 {\glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7017 \ifglstrinsertinside\else#3\fi
7018 }%
7019 }%
7020 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
7021 }%
7022 \glspostlinkhook
7023 }

```

**\glsetabbrvfmt** Set the current format for the given category (or the abbreviation category if unset).

```

7024 \newcommand*{\glsetabbrvfmt}[1]{%
7025 \ifcsdef{@glsabbrv@current@#1}%
7026 {\glstr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7027 {\glstr@applyabbrvfmt{@glsabbrv@current@abbreviation}}%
7028 }

```

**\gluseabbrvfont** Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7029 \newrobustcmd*{\gluseabbrvfont}[2]{\glsetabbrvfmt{#2}\glsabbrvfont{#1}}

```

**\gluselongfont** Provide a way to use the long font for a given category for arbitrary text.

```

7030 \newrobustcmd*{\gluselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}

```

**\glstrgenabbrvfmt** Similar to \glsgenacfmt, but for abbreviations.

```

7031 \newcommand*{\glstrgenabbrvfmt}{%
7032 \ifdefempty\glscustomtext
7033 {%
7034 \ifglused\glslabel
7035 {%

```

Subsequent use:

```

7036 \gl@ifplural
7037 {%

```

Subsequent plural form:

```

7038 \glscapscase
7039 {%

```

Subsequent plural form, don't adjust case:

```
7040      \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%  
7041      }%  
7042      {%
```

Subsequent plural form, make first letter upper case:

```
7043      \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%  
7044      }%  
7045      {%
```

Subsequent plural form, all caps:

```
7046      \mfirstucMakeUppercase  
7047      {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%  
7048      }%  
7049      }%  
7050      {%
```

Subsequent singular form

```
7051      \glscapscase  
7052      {%
```

Subsequent singular form, don't adjust case:

```
7053      \glxtrsubsequentfmt{\glslabel}{\glsinsert}%  
7054      }%  
7055      {%
```

Subsequent singular form, make first letter upper case:

```
7056      \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%  
7057      }%  
7058      {%
```

Subsequent singular form, all caps:

```
7059      \mfirstucMakeUppercase  
7060      {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%  
7061      }%  
7062      }%  
7063      }%  
7064      {%
```

First use:

```
7065      \glsifplural  
7066      {%
```

First use plural form:

```
7067      \glscapscase  
7068      {%
```

First use plural form, don't adjust case:

```
7069      \glxtrfullplformat{\glslabel}{\glsinsert}%  
7070      }%  
7071      {%
```

First use plural form, make first letter upper case:

```
7072      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
7073      }%
7074      {%
```

First use plural form, all caps:

```
7075      \mfirstucMakeUppercase
7076      {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
7077      }%
7078      }%
7079      {%
```

First use singular form

```
7080      \glscapscase
7081      {%
```

First use singular form, don't adjust case:

```
7082      \glsxtrfullformat{\glslabel}{\glsinsert}%
7083      }%
7084      {%
```

First use singular form, make first letter upper case:

```
7085      \Glsxtrfullformat{\glslabel}{\glsinsert}%
7086      }%
7087      {%
```

First use singular form, all caps:

```
7088      \mfirstucMakeUppercase
7089      {\glsxtrfullformat{\glslabel}{\glsinsert}}%
7090      }%
7091      }%
7092      }%
7093      }%
7094      {%
```

User supplied text.

```
7095      \glscustomtext
7096      }%
7097 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7098 \newcommand*{\glsxtrsubsequentfmt}[2]{%
7099   \glsabbrvfont{\glsaccessshort{#1}\ifglsxtrininsertinside #2\fi}%
7100   \ifglsxtrininsertinside \else#2\fi
7101 }
7102 \let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
7103 \newcommand*{\glsxtrsubsequentplfmt}[2]{%
7104   \glsabbrvfont{\glsaccessshortpl{#1}\ifglsxtrininsertinside #2\fi}%
7105   \ifglsxtrininsertinside \else#2\fi
```

```

7106 }
7107 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
7108 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7109   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7110   \ifglxtrininsertinside \else#2\fi
7111 }
7112 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

subsequentplfmt Subsequent use format (plural, first letter uppercase).
7113 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7114   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7115   \ifglxtrininsertinside \else#2\fi
7116 }
7117 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

### 1.7.1 Abbreviation Styles Setup

```

breviationstyle
7118 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7119   \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
7120   {%
7121     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7122   }%
7123   {%

Have abbreviations already been defined for this category?
7124   \ifcsstring{@glsabbrv@current@#1}{#2}%
7125   {%

Style already set.
7126   }%
7127   {%
7128     \def\@glxtr@dostylewarn{%
7129       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7130       {%
7131         \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7132           style has been switched \MessageBreak
7133           for category ‘#1’, \MessageBreak
7134           but there have already been entries \MessageBreak
7135           defined for this category. Unwanted \MessageBreak
7136           side-effects may result}}%
7137         \@endfortrue
7138       }%
7139       \@glxtr@dostylewarn

Set up the style for the given category.
7140       \csdef{@glsabbrv@current@#1}{#2}%
7141       \glxtr@applyabbrvstyle{#2}%

```

```

7142     }%
7143 }%
7144 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7145 \newcommand*{\glxtr@applyabbrvstyle}[1]{%
7146   \csuse{@glsabbrv@dispstyle@setup@#1}%
7147   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7148 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

7149 \newcommand*{\glxtr@applyabbrvfmt}[1]{%
7150   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7151 }

```

`\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.

```

7152 \newcommand*{\newabbreviationstyle}[3]{%
7153   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7154   {%
7155     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7156       defined}{}%
7157   }%
7158   {%
7159     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7160     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7161     #2}%
7162     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7163     \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
7164     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7165     \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
7166     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glxtrsubsequentfmt` etc in case a style changes this.

```

7167     \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
7168     \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
7169     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7170     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7171     #3}%
7172   }%
7173 }

```

`\newabbreviationstyle`

```

7174 \newcommand*{\renewabbreviationstyle}[3]{%
7175   \ifcsundef{@glsabbrv@dispstyle@setup@#1}

```

```

7176 {%
7177   \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7178 }%
7179 {%
7180   \csdef{@glsabbrv@dispstyle@setup@#1}{%
       Initialise hook to do nothing. The style may change this.
7181     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7182     #2}%
7183   \csdef{@glsabbrv@dispstyle@fmts@#1}{%
       Assume in-line form is the same as first use. The style may change this.
7184     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7185     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7186     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7187     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7188     #3}%
7189 }%
7190 }

```

**abbreviationstyle** Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```

7191 \newcommand*{\letabbreviationstyle}[2]{%
7192   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7193   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7194 }

```

**deprecated@abbrstyle** `\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```

7195 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7196   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7197     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7198     \csuse{@glsabbrv@dispstyle@setup@#2}%
7199   }%
7200   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7201 }

```

**deprecatedAbbrStyle** Generate warning for deprecated style use.

```

7202 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7203   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7204     use ‘#2’ instead}%
7205 }

```

**useAbbrStyleSetup**

```

7206 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7207   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%

```

```

7208 {%
7209     \PackageError{glossaries-extra}%
7210     {Unknown abbreviation style definitions ‘#1’}{}%
7211 }%
7212 {%
7213     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7214 }%
7215 }

```

seAbbrStyleFmts

```

7216 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7217     \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7218     {%
7219         \PackageError{glossaries-extra}%
7220         {Unknown abbreviation style formats ‘#1’}{}%
7221     }%
7222     {%
7223         \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7224     }%
7225 }

```

### 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.

```

7226 \newif\ifglsxtrinsertinside
7227 \glsxtrinsertinsidefalse

```

trlongshortname

```

7228 \newcommand*{\glsxtrlongshortname}{%
7229     \protect\glsabbrvfont{\the\glsshorttok}%
7230 }

```

long-short

```

7231 \newabbreviationstyle{long-short}%
7232 {%
7233     \renewcommand*{\CustomAbbreviationFields}{%
7234         name={\glsxtrlongshortname},
7235         sort={\the\glsshorttok},

```



```

7236 first={\protect\glsfirstlongfont{\the\glslongtok}%
7237 \protect\glsxtrfullsep{\the\glslabeltok}%
7238 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7239 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7240 \protect\glsxtrfullsep{\the\glslabeltok}%
7241 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

7242 plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
7243 description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7244 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7245 \glsasattribute{\the\glslabeltok}{regular}%
7246 {%
7247 \glssetattribute{\the\glslabeltok}{regular}{false}%
7248 }%
7249 {}%
7250 }%
7251 }%
7252 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7253 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7254 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7255 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7256 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7257 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7258 \renewcommand*{\glsxtrfullformat}[2]{%
7259 \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7260 \ifglsxtrininsertinside\else##2\fi
7261 \glsxtrfullsep{##1}%
7262 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7263 }%
7264 \renewcommand*{\glsxtrfullplformat}[2]{%
7265 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7266 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7267 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7268 }%
7269 \renewcommand*{\Glsxtrfullformat}[2]{%
7270 \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7271 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7272 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7273 }%
7274 \renewcommand*{\Glsxtrfullplformat}[2]{%
7275 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7276 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7277 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7278 }%
7279 }

```

Set this as the default style for general abbreviations:

```
7280 \setabbreviationstyle{long-short}
```

ngshortdescsort

```
7281 \newcommand*{\glxtrlongshortdescsort}{%
7282 \expandonce\glxtrorglong\space (\expandonce\glxtrorgshort)%
7283 }
```

ngshortdescname

```
7284 \newcommand*{\glxtrlongshortdescname}{%
7285 \protect\glslongfont{\the\glslongtok}
7286 \glxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}}%
7287 }
```

long-short-desc User supplies description. The long form is included in the name.

```
7288 \newabbreviationstyle{long-short-desc}%
7289 {%
7290 \renewcommand*{\CustomAbbreviationFields}{%
7291 name={\glxtrlongshortdescname},
7292 sort={\glxtrlongshortdescsort},%
7293 first={\protect\glsfirstlongfont{\the\glslongtok}%
7294 \protect\glxtrfullsep{\the\glslabeltok}%
7295 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7296 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7297 \protect\glxtrfullsep{\the\glslabeltok}%
7298 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7299 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7300 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7301 }%
```

The text key should only have the short form.

```
7299 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7300 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7301 }%
```

Unset the regular attribute if it has been set.

```
7302 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7303 \glshasattribute{\the\glslabeltok}{regular}%
7304 {%
7305 \glssetattribute{\the\glslabeltok}{regular}{false}%
7306 }%
7307 {}%
7308 }%
7309 }%
7310 {%
7311 \GlsXtrUseAbbrStyleFmts{long-short}%
7312 }
```

trshortlongname

```
7313 \newcommand*{\glxtrshortlongname}{%
7314 \protect\glsabbrvfont{\the\glsshorttok}%
7315 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7316 \newabbreviationstyle{short-long}%
7317 {%
7318   \renewcommand*{\CustomAbbreviationFields}{%
7319     name={\glxtrshortlongname},
7320     sort={\the\glsshorttok},
7321     description={\the\gslongtok},%
7322     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7323       \protect\glxtrfullsep{\the\glslabeltok}%
7324       \glxtrparen{\protect\glsfirstlongfont{\the\gslongtok}}},%
7325     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7326       \protect\glxtrfullsep{\the\glslabeltok}%
7327       \glxtrparen{\protect\glsfirstlongfont{\the\gslongpltok}}},%
7328     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7329 }
```

Unset the regular attribute if it has been set.

```
7329 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7330   \glshasattribute{\the\glslabeltok}{regular}%
7331   {%
7332     \glissetattribute{\the\glslabeltok}{regular}{false}%
7333   }%
7334   {}%
7335 }%
7336 }%
7337 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7338 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7339 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7340 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7341 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7342 \renewcommand*{\gslongfont}[1]{\gslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7343 \renewcommand*{\glxtrfullformat}[2]{%
7344   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7345   \ifglxtrininsertinside\else##2\fi
7346   \glxtrfullsep{##1}%
7347   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7348 }%
7349 \renewcommand*{\glxtrfullplformat}[2]{%
7350   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7351   \ifglxtrininsertinside\else##2\fi
7352   \glxtrfullsep{##1}%
7353   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7354 }%
7355 \renewcommand*{\Glsxtrfullformat}[2]{%
7356   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7357   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7358 }
```

```

7358 \glstrparen{\glfirstlongfont{\glaccesslong{##1}}}%
7359 }%
7360 \renewcommand*{\Glsxtrfullplformat}[2]{%
7361 \glfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7362 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7363 \glstrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
7364 }%
7365 }

```

ortlongdescsort

```

7366 \newcommand*{\glxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7367 \newcommand*{\glxtrshortlongdescname}{%
7368 \protect\glsabbrvfont{\the\glsshorttok}
7369 \glstrparen{\protect\glslongfont{\the\glslongtok}}}%
7370 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7371 \newabbreviationstyle{short-long-desc}%
7372 {%
7373 \renewcommand*{\CustomAbbreviationFields}{%
7374 name={\glxtrshortlongdescname},
7375 sort={\glxtrshortlongdescsort},
7376 first={\protect\glfirstabbrvfont{\the\glsshorttok}}%
7377 \protect\glxtrfullsep{\the\glslabeltok}}%
7378 \glstrparen{\protect\glfirstlongfont{\the\glslongtok}}},%
7379 firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}}%
7380 \protect\glxtrfullsep{\the\glslabeltok}}%
7381 \glstrparen{\protect\glfirstlongfont{\the\glslongpltok}}},%
7382 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7383 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7384 }%

```

Unset the regular attribute if it has been set.

```

7385 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7386 \glshasattribute{\the\glslabeltok}{regular}}%
7387 {%
7388 \glissetattribute{\the\glslabeltok}{regular}{false}}%
7389 }%
7390 {}%
7391 }%
7392 }%
7393 {%
7394 \GlsXtrUseAbbrStyleFmts{short-long}}%
7395 }

```

ongfootnotefont Only used by the “footnote” styles.

```
7396 \newcommand*{\glslongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
7397 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

xtrabbrvfootnote `\glsxtrabbrvfootnote{<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`).

```
7398 \newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}
```

xtrfootnotename

```
7399 \newcommand*{\glsxtrfootnotename}{%  
7400   \protect\glsabbrvfont{\the\glsshorttok}%  
7401 }
```

footnote Short form followed by long form in footnote on first use.

```
7402 \newabbreviationstyle{footnote}%  
7403 {%  
7404   \renewcommand*{\CustomAbbreviationFields}{%  
7405     name={\glsxtrfootnotename},  
7406     sort={\the\glsshorttok},  
7407     description={\the\glslongtok},%  
  
7408     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%  
7409     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%  
7410     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%  
7411     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%  
7412     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%  
7413     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%  
  
7414     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.

```
7415   \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
7416     \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%  
7417     \glshasattribute{\the\glslabeltok}{regular}%  
7418     {%  
7419       \glsssetAttribute{\the\glslabeltok}{regular}{false}%  
7420     }%  
7421     {}%  
7422   }%
```

7423 }%  
 7424 {%

In case the user wants to mix and match font styles, these are redefined here.

```
7425 \renewcommand*\abbrvpluralsuffix{\glstrabbrvpluralsuffix}%
7426 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7427 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7428 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7429 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
7430 \renewcommand*\glstrfullformat[2]{%
7431   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
7432   \ifglstrinsertinside\else##2\fi
7433   \protect\glstrabbrvfootnote{##1}%
7434   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7435 }%
7436 \renewcommand*\glstrfullplformat[2]{%
7437   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
7438   \ifglstrinsertinside\else##2\fi
7439   \protect\glstrabbrvfootnote{##1}%
7440   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7441 }%
7442 \renewcommand*\Glsxtrfullformat[2]{%
7443   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
7444   \ifglstrinsertinside\else##2\fi
7445   \protect\glstrabbrvfootnote{##1}%
7446   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7447 }%
7448 \renewcommand*\Glsxtrfullplformat[2]{%
7449   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
7450   \ifglstrinsertinside\else##2\fi
7451   \protect\glstrabbrvfootnote{##1}%
7452   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7453 }%
```

The first use full form and the inline full form use the short (long) style.

```
7454 \renewcommand*\glstrinlinefullformat[2]{%
7455   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
7456   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
7457   \glstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7458 }%
7459 \renewcommand*\glstrinlinefullplformat[2]{%
7460   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
7461   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
7462   \glstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7463 }%
7464 \renewcommand*\Glsxtrinlinefullformat[2]{%
7465   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
7466   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
7467   \glstrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7468 }
```

```

7468 }%
7469 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7470   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7471   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7472   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7473 }%
7474 }

```

#### short-footnote

```

7475 \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.

```

7476 \newabbreviationstyle{postfootnote}%
7477 {%
7478   \renewcommand*{\CustomAbbreviationFields}{%
7479     name={\glxtrfootnotename},
7480     sort={\the\glsshorttok},
7481     description={\the\glslongtok},%
7482     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7483     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7484     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.

```

7485 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7486   \csdef{glxtrpostlink\glscategorylabel}{%
7487     \glxtrifwasfirstuse
7488     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7489       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7490         {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
7491     }%
7492   }%
7493 }%
7494 \glshasattribute{\the\glslabeltok}{regular}%
7495 {%
7496   \glissetattribute{\the\glslabeltok}{regular}{false}%
7497 }%
7498 {}%
7499 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7500 \renewcommand*{\glxtrsetupfulldefs}{%
7501   \let\glxtrifwasfirstuse\@secondoftwo

```

```

7502 }%
7503 }%
7504 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7505 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7506 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7507 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7508 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7509 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

7510 \renewcommand*\glxtrfullformat[2]{%
7511   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7512   \ifglxtrininsertinside\else##2\fi
7513 }%
7514 \renewcommand*\glxtrfullplformat[2]{%
7515   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7516   \ifglxtrininsertinside\else##2\fi
7517 }%
7518 \renewcommand*\Glsxtrfullformat[2]{%
7519   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7520   \ifglxtrininsertinside\else##2\fi
7521 }%
7522 \renewcommand*\Glsxtrfullplformat[2]{%
7523   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7524   \ifglxtrininsertinside\else##2\fi
7525 }%

```

The first use full form and the inline full form use the short (long) style.

```

7526 \renewcommand*\glxtrininlinefullformat[2]{%
7527   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7528   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7529   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7530 }%
7531 \renewcommand*\glxtrininlinefullplformat[2]{%
7532   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7533   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7534   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7535 }%
7536 \renewcommand*\Glsxtrininlinefullformat[2]{%
7537   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7538   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7539   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7540 }%
7541 \renewcommand*\Glsxtrininlinefullplformat[2]{%
7542   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7543   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7544   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7545 }%
7546 }

```



rt-postfootnote

```
7547 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```
7548 \newcommand*{\glxtrshortnolongname}{%
7549   \protect\glsabbrvfont{\the\glsshorttok}%
7550 }
```

**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.

```
7551 \newabbreviationstyle{short}%
7552 {%
7553   \renewcommand*{\CustomAbbreviationFields}{%
7554     name={\glxtrshortnolongname},
7555     sort={\the\glsshorttok},
7556     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7557     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7558     text={\protect\glsabbrvfont{\the\glsshorttok}},
7559     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7560     description={\the\glslongtok}}%
7561   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7562     \glssetattribute{\the\glslabeltok}{regular}{true}}%
7563 }%
7564 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7565 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7566 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7567 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7568 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7569 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
7570 \renewcommand*{\glxtrinlinefullformat}[2]{%
7571   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7572   \ifglxtrininsertinside##2\fi}%
7573   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7574   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7575 }%
7576 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7577   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7578   \ifglxtrininsertinside##2\fi}%
7579   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7580   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7581 }%
7582 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7583   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
```

```

7584     \ifglxtrinsertinside##2\fi}%
7585     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7586     \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7587 }%
7588 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7589     \protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}%
7590         \ifglxtrinsertinside##2\fi}%
7591     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7592     \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7593 }%

```

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.

```

7594 \renewcommand*{\glxtrfullformat}[2]{%
7595     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7596     \ifglxtrinsertinside\else##2\fi
7597 }%
7598 \renewcommand*{\glxtrfullplformat}[2]{%
7599     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7600     \ifglxtrinsertinside\else##2\fi
7601 }%
7602 \renewcommand*{\Glsxtrfullformat}[2]{%
7603     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7604     \ifglxtrinsertinside\else##2\fi
7605 }%
7606 \renewcommand*{\Glsxtrfullplformat}[2]{%
7607     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7608     \ifglxtrinsertinside\else##2\fi
7609 }%
7610 }

```

Set this as the default style for acronyms:

```

7611 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7612 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7613 \newabbreviationstyle{short-nolong-noreg}%
7614 {%
7615     \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7616 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7617     \glshasattribute{\the\glslabeltok}{regular}%
7618     {%
7619         \glissetattribute{\the\glslabeltok}{regular}{false}%
7620     }%
7621     {}%
7622 }%

```

```

7623 }%
7624 {%
7625   \GlsXtrUseAbbrStyleFmts{short-nolong}%
7626 }

```

trshortdescname

```

7627 \newcommand*{\glxtrshortdescname}{%
7628   \protect\glsabbrvfont{\the\glsshorttok}%
7629 }

```

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.

```

7630 \newabbreviationstyle{short-desc}%
7631 {%
7632   \renewcommand*{\CustomAbbreviationFields}{%
7633     name={\glxtrshortdescname},
7634     sort={\the\glsshorttok},
7635     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7636     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7637     text={\protect\glsabbrvfont{\the\glsshorttok}},
7638     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7639     description={\the\glslongtok}}%
7640   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7641     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
7642 }%
7643 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7644 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7645 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7646 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7647 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7648 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7649 \renewcommand*{\glxtrinlinefullformat}[2]{%
7650   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7651   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7652   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7653 }%
7654 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7655   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7656   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7657   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7658 }%
7659 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7660   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7661   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7662   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7663 }%

```

```

7664 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7665   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7666   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7667   \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7668 }%

```

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.

```

7669 \renewcommand*{\glsxtrfullformat}[2]{%
7670   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7671   \ifglxtrinsertinside\else##2\fi
7672 }%
7673 \renewcommand*{\glsxtrfullplformat}[2]{%
7674   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7675   \ifglxtrinsertinside\else##2\fi
7676 }%
7677 \renewcommand*{\Glsxtrfullformat}[2]{%
7678   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7679   \ifglxtrinsertinside\else##2\fi
7680 }%
7681 \renewcommand*{\Glsxtrfullplformat}[2]{%
7682   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7683   \ifglxtrinsertinside\else##2\fi
7684 }%
7685 }

```

ort-nolong-desc

```
7686 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7687 \newabbreviationstyle{short-nolong-desc-noreg}%
7688 {%
7689   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7690 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7691   \glshasattribute{\the\glslabeltok}{regular}%
7692   {%
7693     \glssetattribute{\the\glslabeltok}{regular}{false}%
7694   }%
7695   {%
7696   }%
7697 }%
7698 {%
7699   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7700 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7701 \newabbreviationstyle{nolong-short}%
7702 {%
7703   \GlsXtrUseAbbrStyleSetup{short-nolong}%
7704 }%
7705 {%
7706   \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7707 \renewcommand*{\glxtrinlinefullformat}[2]{%
7708   \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7709   \ifglxtrininsertinside##2\fi}%
7710 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7711 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7712 }%
7713 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7714   \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7715   \ifglxtrininsertinside##2\fi}%
7716 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7717 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7718 }%
7719 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7720   \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7721   \ifglxtrininsertinside##2\fi}%
7722 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7723 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7724 }%
7725 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7726   \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7727   \ifglxtrininsertinside##2\fi}%
7728 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7729 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7730 }%
7731 }

```

**nolong-short-noreg** Like **nolong-short** but doesn't set the regular attribute.

```

7732 \newabbreviationstyle{nolong-short-noreg}%
7733 {%
7734   \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7735 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7736   \glshasattribute{\the\glslabeltok}{regular}%
7737   {%
7738     \glissetattribute{\the\glslabeltok}{regular}{false}%
7739   }%
7740   {}%
7741 }%
7742 }%
7743 {%
7744   \GlsXtrUseAbbrStyleFmts{nolong-short}%

```

7745 }

noshortdescname

7746 \newcommand\*{\glxtrlongnoshortdescname}{%

7747 \protect\glslongfont{\the\glslongtok}}%

7748 }

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.

7749 \newabbreviationstyle{long-desc}{%

7750 {%

7751 \renewcommand\*{\CustomAbbreviationFields}{%

7752 name={\glxtrlongnoshortdescname},

7753 sort={\the\glslongtok},

7754 first={\protect\glsfirstlongfont{\the\glslongtok}},

7755 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},

7756 text={\glslongfont{\the\glslongtok}},

7757 plural={\glslongfont{\the\glslongpltok}}%

7758 }%

7759 \renewcommand\*{\GlsXtrPostNewAbbreviation}{%

7760 \glssetattribute{\the\glslabeltok}{regular}{true}}%

7761 }%

7762 {%

In case the user wants to mix and match font styles, these are redefined here.

7763 \renewcommand\*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

7764 \renewcommand\*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%

7765 \renewcommand\*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%

7766 \renewcommand\*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%

7767 \renewcommand\*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

The format for subsequent use (not used when the regular attribute is set).

7768 \renewcommand\*{\glxtrsubsequentfmt}[2]{%

7769 \glslongfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%

7770 \ifglxtrininsertinside \else##2\fi

7771 }%

7772 \renewcommand\*{\glxtrsubsequentplfmt}[2]{%

7773 \glslongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

7774 \ifglxtrininsertinside \else##2\fi

7775 }%

7776 \renewcommand\*{\Glsxtrsubsequentfmt}[2]{%

7777 \glslongfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%

7778 \ifglxtrininsertinside \else##2\fi

7779 }%

7780 \renewcommand\*{\Glsxtrsubsequentplfmt}[2]{%

7781 \glslongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

7782 \ifglxtrininsertinside \else##2\fi

7783 }%

The inline full form displays the long format followed by the short form in parentheses.

```

7784 \renewcommand*{\glxtrinlinefullformat}[2]{%
7785   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7786   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7787   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7788 }%
7789 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7790   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7791   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7792   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7793 }%
7794 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7795   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7796   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7797   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7798 }%
7799 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7800   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7801   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7802   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7803 }%

```

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.

```

7804 \renewcommand*{\glxtrfullformat}[2]{%
7805   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7806   \ifglxtrinsertinside\else##2\fi
7807 }%
7808 \renewcommand*{\glxtrfullplformat}[2]{%
7809   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7810   \ifglxtrinsertinside\else##2\fi
7811 }%
7812 \renewcommand*{\Glsxtrfullformat}[2]{%
7813   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7814   \ifglxtrinsertinside\else##2\fi
7815 }%
7816 \renewcommand*{\Glsxtrfullplformat}[2]{%
7817   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7818   \ifglxtrinsertinside\else##2\fi
7819 }%
7820 }

```

`ng-noshort-desc` Provide a synonym that matches similar styles.

```
7821 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

`short-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```

7822 \newabbreviationstyle{long-noshort-desc-noreg}%
7823 {%
7824   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

Unset the regular attribute if it has been set.

```

7825 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7826   \glshasattribute{\the\glslabeltok}{regular}%
7827   {%
7828     \glsselattribute{\the\glslabeltok}{regular}{false}%
7829   }%
7830   {}%
7831 }%
7832 }%
7833 {%
7834   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7835 }

```

longnoshortname

```

7836 \newcommand*{\glsxtrlongnoshortname}{%
7837   \protect\glssabrvfont{\the\glsshorttok}%
7838 }

```

**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.

```

7839 \newabbreviationstyle{long}%
7840 {%
7841   \renewcommand*{\CustomAbbreviationFields}{%
7842     name={\glsxtrlongnoshortname},
7843     sort={\the\glsshorttok},
7844     first={\protect\glslongfont{\the\glslongtok}},
7845     firstplural={\protect\glslongfont{\the\glslongpltok}},
7846     text={\glslongfont{\the\glslongtok}},
7847     plural={\glslongfont{\the\glslongpltok}},%
7848     description={\the\glslongtok}%
7849   }%
7850   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7851     \glsselattribute{\the\glslabeltok}{regular}{true}}%
7852 }%
7853 {%
7854   \GlsXtrUseAbbrStyleFmts{long-desc}%
7855 }

```

**long-noshort** Provide a synonym that matches similar styles.

```

7856 \letabbreviationstyle{long-noshort}{long}

```

**g-noshort-noreg** Like long-noshort but doesn't set the regular attribute.

```

7857 \newabbreviationstyle{long-noshort-noreg}%
7858 {%
7859   \GlsXtrUseAbbrStyleSetup{long-noshort}%

```

Unset the regular attribute if it has been set.

```

7860 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```



```

7861 \glshasattribute{\the\glslabeltok}{regular}%
7862 {%
7863 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7864 }%
7865 {}%
7866 }%
7867 }%
7868 {%
7869 \GlsXtrUseAbbrStyleFmts{long-noshort}%
7870 }

```

### 1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
7871 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7872 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`\glsxtrfirstscfont` Maintained for backward-compatibility.

```
7873 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`\glstfirstabbrvscfont` Added for consistent naming.

```
7874 \newcommand*{\glstfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
7875 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

`long-short-sc`

```

7876 \newabbreviationstyle{long-short-sc}%
7877 {%
7878 \renewcommand*{\CustomAbbreviationFields}{%
7879 name={\glsxtrlongshortname},
7880 sort={\the\glsshorttok},
7881 first={\protect\glstfirstlongdefaultfont{\the\glslongtok}%
7882 \protect\glsxtrfullsep{\the\glslabeltok}%
7883 \glxtrparen{\protect\glstfirstabbrvscfont{\the\glsshorttok}}},%
7884 firstplural={\protect\glstfirstlongdefaultfont{\the\glslongpltok}%
7885 \protect\glsxtrfullsep{\the\glslabeltok}%
7886 \glxtrparen{\protect\glstfirstabbrvscfont{\the\glsshortpltok}}},%
7887 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}},%
7888 description={\the\glslongtok}}%
7889 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7890 \glshasattribute{\the\glslabeltok}{regular}%
7891 }%

```

```

7892     \glsetattribute{\the\glslabeltok}{regular}{false}%
7893   }%
7894   {}%
7895 }%
7896}%
7897{%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7898 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
7899 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7900 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

7901 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7902 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7903 \renewcommand*\glsxtrfullformat[2]{%
7904   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7905   \ifglsxtrininsertinside\else##2\fi
7906   \glsxtrfullsep{##1}%
7907   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7908 }%
7909 \renewcommand*\glsxtrfullplformat[2]{%
7910   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7911   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7912   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7913 }%
7914 \renewcommand*\Glsxtrfullformat[2]{%
7915   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7916   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7917   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7918 }%
7919 \renewcommand*\Glsxtrfullplformat[2]{%
7920   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7921   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7922   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7923 }%
7924 }

```

g-short-sc-desc

```

7925 \newabbreviationstyle{long-short-sc-desc}%
7926 {%
7927   \renewcommand*\CustomAbbreviationFields{%
7928     name={\glsxtrlongshortdescname},
7929     sort={\glsxtrlongshortdescsort},%
7930     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7931       \protect\glsxtrfullsep{\the\glslabeltok}%
7932       \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7933     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%

```

```

7934 \protect\glxtrfullsep{\the\glslabeltok}%
7935 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7936 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7937 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7938 }%

```

Unset the regular attribute if it has been set.

```

7939 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7940 \glshasattribute{\the\glslabeltok}{regular}%
7941 {%
7942 \glissetattribute{\the\glslabeltok}{regular}{false}%
7943 }%
7944 {}%
7945 }%
7946 }%
7947 {%

```

As long-short-sc style:

```

7948 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7949 }

```

Now the short (long) version

```

7950 \newabbreviationstyle{short-sc-long}%
7951 {%
7952 \renewcommand*{\CustomAbbreviationFields}{%
7953 name={\glxtrshortlongname},
7954 sort={\the\glsshorttok},
7955 description={\the\glslongtok},%
7956 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
7957 \protect\glxtrfullsep{\the\glslabeltok}%
7958 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7959 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
7960 \protect\glxtrfullsep{\the\glslabeltok}%
7961 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7962 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7963 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7964 \glshasattribute{\the\glslabeltok}{regular}%
7965 {%
7966 \glissetattribute{\the\glslabeltok}{regular}{false}%
7967 }%
7968 {}%
7969 }%
7970 }%
7971 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7972 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7973 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7974 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

```

7975 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7976 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7977 \renewcommand*{\glsxtrfullformat}[2]{%
7978   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7979   \ifglsxtrininsertinside\else##2\fi
7980   \glsxtrfullsep{##1}%
7981   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7982 }%
7983 \renewcommand*{\glsxtrfullplformat}[2]{%
7984   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7985   \ifglsxtrininsertinside\else##2\fi
7986   \glsxtrfullsep{##1}%
7987   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7988 }%
7989 \renewcommand*{\Glsxtrfullformat}[2]{%
7990   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7991   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7992   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7993 }%
7994 \renewcommand*{\Glsxtrfullplformat}[2]{%
7995   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7996   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7997   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7998 }%
7999 }

```

As before but user provides description

```

8000 \newabbreviationstyle{short-sc-long-desc}%
8001 {%
8002   \renewcommand*{\CustomAbbreviationFields}{%
8003     name={\glsxtrshortlongdescname},
8004     sort={\glsxtrshortlongdescsort},
8005     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8006       \protect\glsxtrfullsep{\the\glslabeltok}%
8007       \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8008     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8009       \protect\glsxtrfullsep{\the\glslabeltok}%
8010       \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8011     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8012     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8013 }%

```

Unset the regular attribute if it has been set.

```

8014 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8015   \glshasattribute{\the\glslabeltok}{regular}%
8016   {%
8017     \glssetattribute{\the\glslabeltok}{regular}{false}%
8018   }%

```

```

8019    {}%
8020  }%
8021 }%
8022 {%
    As short-sc-long style:
8023   \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8024 }

```

#### short-sc

```

8025 \newabbreviationstyle{short-sc}%
8026 {%
8027   \renewcommand*{\CustomAbbreviationFields}{%
8028     name={\glxtrshortnolongname},
8029     sort={\the\glsshorttok},
8030     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8031     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8032     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8033     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8034     description={\the\glslongtok}}%
8035   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8036     \glssetAttribute{\the\glslabeltok}{regular}{true}}%
8037 }%
8038 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8039   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8040   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8041   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8042   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8043   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8044   \renewcommand*{\glxtrinlinefullformat}[2]{%
8045     \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8046     \ifglxtrininsertinside##2\fi}%
8047   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8048   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8049 }%
8050   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8051     \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8052     \ifglxtrininsertinside##2\fi}%
8053   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8054   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8055 }%

8056   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8057     \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
8058     \ifglxtrininsertinside##2\fi}%
8059   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8060   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%

```

```

8061 }%
8062 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8063   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}%
8064   \ifglxtrinsertinside##2\fi}%
8065   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8066   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8067 }%

```

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.

```

8068 \renewcommand*{\glxtrfullformat}[2]{%
8069   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8070   \ifglxtrinsertinside\else##2\fi
8071 }%
8072 \renewcommand*{\glxtrfullplformat}[2]{%
8073   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8074   \ifglxtrinsertinside\else##2\fi
8075 }%
8076 \renewcommand*{\Glsxtrfullformat}[2]{%
8077   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8078   \ifglxtrinsertinside\else##2\fi
8079 }%
8080 \renewcommand*{\Glsxtrfullplformat}[2]{%
8081   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8082   \ifglxtrinsertinside\else##2\fi
8083 }%
8084 }

```

short-sc-nolong

```

8085 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8086 \newabbreviationstyle{short-sc-desc}%
8087 {%
8088   \renewcommand*{\CustomAbbreviationFields}{%
8089     name={\glxtrshortdescname},
8090     sort={\the\glsshorttok},
8091     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8092     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8093     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8094     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8095     description={\the\glslongtok}}%
8096   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8097     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8098 }%
8099 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8100 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8101 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%

```

```

8102 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8103 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8104 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8105 \renewcommand*\glsxtrinlinefullformat}[2]{%
8106   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8107   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8108   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8109 }%
8110 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8111   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8112   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8113   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8114 }%
8115 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8116   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8117   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8118   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8119 }%
8120 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8121   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8122   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8123   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8124 }%

```

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.

```

8125 \renewcommand*\glsxtrfullformat}[2]{%
8126   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8127   \ifglsxtrininsertinside\else##2\fi
8128 }%
8129 \renewcommand*\glsxtrfullplformat}[2]{%
8130   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8131   \ifglsxtrininsertinside\else##2\fi
8132 }%
8133 \renewcommand*\Glsxtrfullformat}[2]{%
8134   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8135   \ifglsxtrininsertinside\else##2\fi
8136 }%
8137 \renewcommand*\Glsxtrfullplformat}[2]{%
8138   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8139   \ifglsxtrininsertinside\else##2\fi
8140 }%
8141 }

```

-sc-nolong-desc

```

8142 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

8143 \newabbreviationstyle{nolong-short-sc}%
8144 {%
8145   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8146 }%
8147 {%
8148   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8149 \renewcommand*{\glxtrinlinefullformat}[2]{%
8150   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}}%
8151   \ifglxtrinsertinside##2\fi}%
8152 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8153 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8154 }%
8155 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8156   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}%
8157   \ifglxtrinsertinside##2\fi}%
8158 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8159 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8160 }%
8161 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8162   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}}%
8163   \ifglxtrinsertinside##2\fi}%
8164 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8165 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8166 }%
8167 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8168   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}%
8169   \ifglxtrinsertinside##2\fi}%
8170 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8171 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8172 }%
8173 }

```

`long-noshort-sc` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

8174 \newabbreviationstyle{long-noshort-sc}%
8175 {%
8176   \renewcommand*{\CustomAbbreviationFields}{%
8177     name={\glxtrlongnoshortname},
8178     sort={\the\glsshorttok},
8179     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8180     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8181     text={\protect\glslongdefaultfont{\the\glslongtok}},
8182     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8183     description={\the\glslongtok}%
8184   }%
8185   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8186     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8187 }%

```



8188 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8189 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8190 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8191 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvscfont{##1}}%
8192 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
8193 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8194 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8195   \glsslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8196   \ifglxtrininsertinside \else##2\fi
8197 }%
8198 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8199   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8200   \ifglxtrininsertinside \else##2\fi
8201 }%
8202 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8203   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8204   \ifglxtrininsertinside \else##2\fi
8205 }%
8206 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8207   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8208   \ifglxtrininsertinside \else##2\fi
8209 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
8210 \renewcommand*{\glxtrinlinefullformat}[2]{%
8211   \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8212   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8213   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshort{##1}}}%
8214 }%
8215 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8216   \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8217   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8218   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8219 }%
8220 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8221   \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8222   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8223   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshort{##1}}}%
8224 }%
8225 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8226   \glssfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8227   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8228   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshortpl{##1}}}%
8229 }%
```

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.

```

8230 \renewcommand*{\glxtrfullformat}[2]{%
8231   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8232   \ifglxtrinsertinside\else##2\fi
8233 }%
8234 \renewcommand*{\glxtrfullplformat}[2]{%
8235   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8236   \ifglxtrinsertinside\else##2\fi
8237 }%
8238 \renewcommand*{\Glsxtrfullformat}[2]{%
8239   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8240   \ifglxtrinsertinside\else##2\fi
8241 }%
8242 \renewcommand*{\Glsxtrfullplformat}[2]{%
8243   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8244   \ifglxtrinsertinside\else##2\fi
8245 }%
8246 }

```

long-sc Backward compatibility:

```
8247 \@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.

```

8248 \newabbreviationstyle{long-noshort-sc-desc}%
8249 {%
8250   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8251 }%
8252 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8253 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8254 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8255 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8256 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8257 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8258 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8259   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8260   \ifglxtrinsertinside \else##2\fi
8261 }%
8262 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8263   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8264   \ifglxtrinsertinside \else##2\fi
8265 }%
8266 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8267   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8268   \ifglxtrinsertinside \else##2\fi
8269 }%
8270 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

```

```

8271 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8272 \ifglxtrinsertinside \else##2\fi
8273 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8274 \renewcommand*{\glxtrinlinefullformat}[2]{%
8275 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8276 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8277 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8278 }%
8279 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8280 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8281 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8282 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8283 }%
8284 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8285 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8286 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8287 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8288 }%
8289 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8290 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8291 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8292 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8293 }%

```

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.

```

8294 \renewcommand*{\glxtrfullformat}[2]{%
8295 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8296 \ifglxtrinsertinside\else##2\fi
8297 }%
8298 \renewcommand*{\glxtrfullplformat}[2]{%
8299 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8300 \ifglxtrinsertinside\else##2\fi
8301 }%
8302 \renewcommand*{\Glsxtrfullformat}[2]{%
8303 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8304 \ifglxtrinsertinside\else##2\fi
8305 }%
8306 \renewcommand*{\Glsxtrfullplformat}[2]{%
8307 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8308 \ifglxtrinsertinside\else##2\fi
8309 }%
8310 }

```

long-desc-sc Backward compatibility:

```

8311 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

8312 \newabbreviationstyle{short-sc-footnote}%
8313 {%
8314   \renewcommand*{\CustomAbbreviationFields}{%
8315     name={\glxtrfootnotename},
8316     sort={\the\glsshorttok},
8317     description={\the\glslongtok},%
8318     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8319       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8320       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8321     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8322       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8323       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8324     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.

```

8325   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8326     \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8327     \glsattribute{\the\glslabeltok}{regular}%
8328     {%
8329       \glssetattribute{\the\glslabeltok}{regular}{false}%
8330     }%
8331   }%
8332 }%
8333 }%
8334 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8335   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8336   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8337   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8338   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8339   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8340   \renewcommand*{\glxtrfullformat}[2]{%
8341     \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8342     \ifglxtrinsertinside\else##2\fi
8343     \protect\glxtrabbrvfootnote{##1}%
8344     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8345   }%
8346   \renewcommand*{\glxtrfullplformat}[2]{%
8347     \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8348     \ifglxtrinsertinside\else##2\fi
8349     \protect\glxtrabbrvfootnote{##1}%
8350     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8351   }%
8352   \renewcommand*{\Glsxtrfullformat}[2]{%
8353     \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8354     \ifglxtrinsertinside\else##2\fi
8355     \protect\glxtrabbrvfootnote{##1}%

```

```

8356     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8357 }%
8358 \renewcommand*{\Glsxtrfullplformat}[2]{%
8359   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8360   \ifglxtrinsertinside\else##2\fi
8361   \protect\glsxtrabbrvfootnote{##1}%
8362   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8363 }%

```

The first use full form and the inline full form use the short (long) style.

```

8364 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8365   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8366   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8367   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8368 }%
8369 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8370   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8371   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8372   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8373 }%
8374 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8375   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8376   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8377   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8378 }%
8379 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8380   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8381   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8382   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8383 }%
8384 }

```

**footnote-sc** Backward compatibility:

```

8385 \@glsxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

**sc-postfootnote**

```

8386 \newabbreviationstyle{short-sc-postfootnote}%
8387 {%
8388   \renewcommand*{\CustomAbbreviationFields}{%
8389     name={\glsxtrfootnotename},
8390     sort={\the\glsshorttok},
8391     description={\the\glslongtok},%
8392     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8393     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8394     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.

```

8395 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8396   \csdef{glsxtrpostlink\glscategorylabel}{%

```

```

8397      \glxtrifwasfirstuse
8398      {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8399      \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8400      {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
8401      }%
8402      {%
8403      }%
8404      \glshasattribute{\the\glslabeltok}{regular}%
8405      {%
8406      \glissetattribute{\the\glslabeltok}{regular}{false}%
8407      }%
8408      {%}%
8409      }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8410  \renewcommand*{\glxtrsetupfulldefs}{%
8411  \let\glxtrifwasfirstuse\@secondoftwo
8412  }%
8413 }%
8414 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8415  \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8416  \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
8417  \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvscfont{##1}}%
8418  \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
8419  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8420  \renewcommand*{\glxtrfullformat}[2]{%
8421  \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8422  \ifglxtrininsertinside\else##2\fi
8423  }%
8424  \renewcommand*{\glxtrfullplformat}[2]{%
8425  \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8426  \ifglxtrininsertinside\else##2\fi
8427  }%
8428  \renewcommand*{\Glsxtrfullformat}[2]{%
8429  \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8430  \ifglxtrininsertinside\else##2\fi
8431  }%
8432  \renewcommand*{\Glsxtrfullplformat}[2]{%
8433  \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8434  \ifglxtrininsertinside\else##2\fi
8435  }%

```

The first use full form and the inline full form use the short (long) style.

```

8436 \renewcommand*{\glxtrinlinefullformat}[2]{%
8437   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8438   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8439   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8440 }%
8441 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8442   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8443   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8444   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8445 }%
8446 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8447   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8448   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8449   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8450 }%
8451 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8452   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8453   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8454   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8455 }%
8456 }

```

postfootnote-sc Backward compatibility:

```
8457 \@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.

```
8458 \newcommand*{\glxtrsmfont}[1]{\textsmaller{##1}}
```

`\glsabbrvsmfont` Added for consistent naming.

```
8459 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}
```

`\glxtrfirstsmfont` Maintained for backward compatibility.

```
8460 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{##1}}
```

`\glfirstabbrvsmfont` Added for consistent naming.

```
8461 \newcommand*{\glfirstabbrvsmfont}{\glxtrfirstsmfont}
```

and for the default short form suffix:

`\glxtrsmsuffix`

```
8462 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8463 \newabbreviationstyle{long-short-sm}%
8464 {%
8465   \renewcommand*{\CustomAbbreviationFields}{%
8466     name={\glxtrlongshortname},
8467     sort={\the\glsshorttok},
8468     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8469       \protect\glxtrfullsep{\the\glslabeltok}%
8470       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8471     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8472       \protect\glxtrfullsep{\the\glslabeltok}%
8473       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8474     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8475     description={\the\glslongtok}}%
8476 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8477   \glshasattribute{\the\glslabeltok}{regular}%
8478   {%
8479     \glissetattribute{\the\glslabeltok}{regular}{false}%
8480   }%
8481   }%
8482 }%
8483 }%
8484 {%
8485   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8486   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8487   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8488 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8489 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```
8490 \renewcommand*{\glxtrfullformat}[2]{%
8491   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8492   \ifglxtrininsertinside\else##2\fi
8493   \glxtrfullsep{##1}%
8494   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8495 }%
8496 \renewcommand*{\glxtrfullplformat}[2]{%
8497   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8498   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8499   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8500 }%
8501 \renewcommand*{\Glsxtrfullformat}[2]{%
8502   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8503   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8504   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8505 }%
8506 \renewcommand*{\Glsxtrfullplformat}[2]{%
8507   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

```



```

8508 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8509 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8510 }%
8511 }

```

g-short-sm-desc

```

8512 \newabbreviationstyle{long-short-sm-desc}%
8513 {%
8514 \renewcommand*{\CustomAbbreviationFields}{%
8515   name={\glxtrlongshortdescname},
8516   sort={\glxtrlongshortdescsort},%
8517   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8518     \protect\glxtrfullsep{\the\glslabeltok}%
8519     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8520   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8521     \protect\glxtrfullsep{\the\glslabeltok}%
8522     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8523   text={\protect\glssabbrvsmfont{\the\glsshorttok}},%
8524   plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%
8525 }%

```

Unset the regular attribute if it has been set.

```

8526 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8527   \glshasattribute{\the\glslabeltok}{regular}%
8528   {%
8529     \glissetattribute{\the\glslabeltok}{regular}{false}%
8530   }%
8531   {}%
8532 }%
8533 }%
8534 {%

```

As long-short-sm style:

```

8535 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8536 }

```

short-sm-long Now the short (long) version

```

8537 \newabbreviationstyle{short-sm-long}%
8538 {%
8539 \renewcommand*{\CustomAbbreviationFields}{%
8540   name={\glxtrshortlongname},
8541   sort={\the\glsshorttok},
8542   description={\the\glslongtok},%
8543   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8544     \protect\glxtrfullsep{\the\glslabeltok}%
8545     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8546   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8547     \protect\glxtrfullsep{\the\glslabeltok}%
8548     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8549   plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8550 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8551   \glshasattribute{\the\glslabeltok}{regular}%
8552   {%
8553     \glsselattribute{\the\glslabeltok}{regular}{false}%
8554   }%
8555   {}%
8556 }%
8557 }%
8558 {%
8559 \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8560 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvsmfont{##1}}%
8561 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8562 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8563 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8564 \renewcommand*{\glxtrfullformat}[2]{%
8565   \glfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8566   \ifglxtrininsertinside\else##2\fi
8567   \glxtrfullsep{##1}%
8568   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
8569 }%
8570 \renewcommand*{\glxtrfullplformat}[2]{%
8571   \glfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8572   \ifglxtrininsertinside\else##2\fi
8573   \glxtrfullsep{##1}%
8574   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8575 }%
8576 \renewcommand*{\Glsxtrfullformat}[2]{%
8577   \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8578   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8579   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8580 }%
8581 \renewcommand*{\Glsxtrfullplformat}[2]{%
8582   \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8583   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8584   \glxtrparen{\glfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8585 }%
8586 }

```

rt-sm-long-desc As before but user provides description

```

8587 \newabbreviationstyle{short-sm-long-desc}%
8588 {%
8589 \renewcommand*{\CustomAbbreviationFields}{%
8590   name={\glxtrshortlongdescname},
8591   sort={\glxtrshortlongdescsort},
8592   first={\protect\glfirstabbrvsmfont{\the\glsshorttok}}%
8593   \protect\glxtrfullsep{\the\glslabeltok}%

```

```

8594 \glxtrparen{\protect\glfirstlongdefaultfont{\the\glslongtok}}},%
8595 firstplural={\protect\glfirstabbrvsmfont{\the\glsshortpltok}%
8596 \protect\glxtrfullsep{\the\glslabeltok}%
8597 \glxtrparen{\protect\glfirstlongdefaultfont{\the\glslongpltok}}},%
8598 text={\protect\glabbrvsmfont{\the\glsshorttok}},%
8599 plural={\protect\glabbrvsmfont{\the\glsshortpltok}}%
8600 }%

```

Unset the regular attribute if it has been set.

```

8601 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8602 \glshasattribute{\the\glslabeltok}{regular}%
8603 {%
8604 \glissetattribute{\the\glslabeltok}{regular}{false}%
8605 }%
8606 {}%
8607 }%
8608 }%
8609 {%

```

As short-sm-long style:

```

8610 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8611 }

```

short-sm

```

8612 \newabbreviationstyle{short-sm}%
8613 {%
8614 \renewcommand*{\CustomAbbreviationFields}{%
8615 name={\glxtrshortnolongname},
8616 sort={\the\glsshorttok},
8617 first={\protect\glfirstabbrvsmfont{\the\glsshorttok}},
8618 firstplural={\protect\glfirstabbrvsmfont{\the\glsshortpltok}},
8619 text={\protect\glabbrvsmfont{\the\glsshorttok}},
8620 plural={\protect\glabbrvsmfont{\the\glsshortpltok}},
8621 description={\the\glslongtok}}%
8622 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8623 \glissetattribute{\the\glslabeltok}{regular}{true}}%
8624 }%
8625 {%
8626 \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8627 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvsmfont{##1}}%
8628 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8629 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
8630 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8631 \renewcommand*{\glxtrinlinefullformat}[2]{%
8632 \protect\glfirstabbrvsmfont{\glaccessshort{##1}}%
8633 \ifglxtrininsertinside##2\fi}%
8634 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8635 \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%

```

```

8636 }%
8637 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8638   \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}%
8639     \ifglxtrininsertinside##2\fi}%
8640   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8641   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8642 }%

8643 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8644   \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8645     \ifglxtrininsertinside##2\fi}%
8646   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8647   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8648 }%

8649 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8650   \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8651     \ifglxtrininsertinside##2\fi}%
8652   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8653   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8654 }%

```

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.

```

8655 \renewcommand*{\glxtrfullformat}[2]{%
8656   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8657   \ifglxtrininsertinside\else##2\fi
8658 }%
8659 \renewcommand*{\glxtrfullplformat}[2]{%
8660   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8661   \ifglxtrininsertinside\else##2\fi
8662 }%
8663 \renewcommand*{\Glsxtrfullformat}[2]{%
8664   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8665   \ifglxtrininsertinside\else##2\fi
8666 }%
8667 \renewcommand*{\Glsxtrfullplformat}[2]{%
8668   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8669   \ifglxtrininsertinside\else##2\fi
8670 }%
8671 }

```

short-sm-nolong

```
8672 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```

8673 \newabbreviationstyle{short-sm-desc}%
8674 {%
8675   \renewcommand*{\CustomAbbreviationFields}{%
8676     name={\glxtrshortdescname},

```

```

8677     sort={\the\glsshorttok},
8678     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8679     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8680     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8681     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8682     description={\the\glslongtok}}%
8683 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8684   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8685 }%
8686 {%
8687   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8688   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8689   \renewcommand*\abbrvpluralsuffix{\protect\glsxtrmsuffix}%
8690   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8691   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8692   \renewcommand*\glsxtrinlinefullformat}[2]{%
8693     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8694     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8695     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8696   }%
8697   \renewcommand*\glsxtrinlinefullplformat}[2]{%
8698     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8699     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8700     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8701   }%
8702   \renewcommand*\Glsxtrinlinefullformat}[2]{%
8703     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8704     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8705     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8706   }%
8707   \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8708     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8709     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8710     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8711   }%

```

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.

```

8712   \renewcommand*\glsxtrfullformat}[2]{%
8713     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8714     \ifglsxtrininsertinside\else##2\fi
8715   }%
8716   \renewcommand*\glsxtrfullplformat}[2]{%
8717     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8718     \ifglsxtrininsertinside\else##2\fi
8719   }%
8720   \renewcommand*\Glsxtrfullformat}[2]{%
8721     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%

```

```

8722     \ifglxtrinsertinside\else##2\fi
8723 }%
8724 \renewcommand*{\Glsxtrfullplformat}[2]{%
8725     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8726     \ifglxtrinsertinside\else##2\fi
8727 }%
8728 }

```

-sm-nolong-desc

```

8729 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8730 \newabbreviationstyle{nolong-short-sm}%
8731 {%
8732     \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8733 }%
8734 {%
8735     \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8736 \renewcommand*{\glxtrinlinefullformat}[2]{%
8737     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8738         \ifglxtrinsertinside##2\fi}%
8739     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8740     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8741 }%
8742 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8743     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8744         \ifglxtrinsertinside##2\fi}%
8745     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8746     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8747 }%
8748 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8749     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8750         \ifglxtrinsertinside##2\fi}%
8751     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8752     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8753 }%
8754 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8755     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8756         \ifglxtrinsertinside##2\fi}%
8757     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8758     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8759 }%
8760 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8761 \newabbreviationstyle{long-noshort-sm}%

```

```

8762 {%
8763   \renewcommand*{\CustomAbbreviationFields}{%
8764     name={\glxtrlongnosshortname},
8765     sort={\the\glssshorttok},
8766     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8767     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8768     text={\protect\glslongdefaultfont{\the\glslongtok}},
8769     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8770     description={\the\glslongtok}%
8771   }%
8772   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8773     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8774 }%
8775 {%
8776   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8777   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvsmfont{##1}}%
8778   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8779   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8780   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8781   \renewcommand*{\glxtrsubsequentfmt}[2]{%
8782     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8783     \ifglxtrininsertinside \else##2\fi
8784   }%
8785   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8786     \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8787     \ifglxtrininsertinside \else##2\fi
8788   }%
8789   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8790     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8791     \ifglxtrininsertinside \else##2\fi
8792   }%
8793   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8794     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8795     \ifglxtrininsertinside \else##2\fi
8796   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8797   \renewcommand*{\glxtrininlinefullformat}[2]{%
8798     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8799     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8800     \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8801   }%
8802   \renewcommand*{\glxtrininlinefullplformat}[2]{%
8803     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8804     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8805     \glxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8806   }%
8807   \renewcommand*{\Glsxtrininlinefullformat}[2]{%

```

```

8808 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8809 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8810 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8811 }%
8812 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8813 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8814 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8815 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8816 }%

```

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.

```

8817 \renewcommand*{\glxtrfullformat}[2]{%
8818 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8819 \ifglxtrinsertinside\else##2\fi
8820 }%
8821 \renewcommand*{\glxtrfullplformat}[2]{%
8822 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8823 \ifglxtrinsertinside\else##2\fi
8824 }%
8825 \renewcommand*{\Glsxtrfullformat}[2]{%
8826 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8827 \ifglxtrinsertinside\else##2\fi
8828 }%
8829 \renewcommand*{\Glsxtrfullplformat}[2]{%
8830 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8831 \ifglxtrinsertinside\else##2\fi
8832 }%
8833 }

```

long-sm Backward compatibility:

```

8834 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8835 \newabbreviationstyle{long-noshort-sm-desc}%
8836 {%
8837 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8838 }%
8839 {%
8840 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8841 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8842 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8843 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8844 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8845 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8846 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8847 \ifglxtrinsertinside \else##2\fi

```



```

8848 }%
8849 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8850   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8851   \ifglxtrinsertinside \else##2\fi
8852 }%
8853 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8854   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8855   \ifglxtrinsertinside \else##2\fi
8856 }%
8857 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8858   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8859   \ifglxtrinsertinside \else##2\fi
8860 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8861 \renewcommand*{\glxtrinlinefullformat}[2]{%
8862   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8863   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8864   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8865 }%
8866 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8867   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8868   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8869   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8870 }%
8871 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8872   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8873   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8874   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8875 }%
8876 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8877   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8878   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8879   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8880 }%

```

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.

```

8881 \renewcommand*{\glxtrfullformat}[2]{%
8882   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8883   \ifglxtrinsertinside\else##2\fi
8884 }%
8885 \renewcommand*{\glxtrfullplformat}[2]{%
8886   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8887   \ifglxtrinsertinside\else##2\fi
8888 }%
8889 \renewcommand*{\Glsxtrfullformat}[2]{%
8890   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8891   \ifglxtrinsertinside\else##2\fi
8892 }%

```

```

8893 \renewcommand*{\Glsxtrfullplformat}[2]{%
8894   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8895   \ifglxtrinsertinside\else##2\fi
8896 }%
8897 }

```

long-desc-sm Backward compatibility:

```

8898 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8899 \newabbreviationstyle{short-sm-footnote}%
8900 {%
8901   \renewcommand*{\CustomAbbreviationFields}{%
8902     name={\glxtrfootnotename},
8903     sort={\the\glsshorttok},
8904     description={\the\glslongtok},%
8905     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8906       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8907       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8908     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8909       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8910       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8911     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.

```

8912 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8913   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8914   \glsattribute{\the\glslabeltok}{regular}%
8915   {%
8916     \glssetattribute{\the\glslabeltok}{regular}{false}%
8917   }%
8918   {}%
8919 }%
8920 }%
8921 {%
8922   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8923   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8924   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8925   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8926   \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8927 \renewcommand*{\glxtrfullformat}[2]{%
8928   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8929   \ifglxtrinsertinside\else##2\fi
8930   \protect\glxtrabbrvfootnote{##1}%
8931   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8932 }%
8933 \renewcommand*{\glxtrfullplformat}[2]{%

```

```

8934 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8935 \ifglxtrinsertinside\else##2\fi
8936 \protect\glxtrabbrvfootnote{##1}%
8937 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8938 }%
8939 \renewcommand*{\Glsxtrfullformat}[2]{%
8940 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8941 \ifglxtrinsertinside\else##2\fi
8942 \protect\glxtrabbrvfootnote{##1}%
8943 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8944 }%
8945 \renewcommand*{\Glsxtrfullplformat}[2]{%
8946 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8947 \ifglxtrinsertinside\else##2\fi
8948 \protect\glxtrabbrvfootnote{##1}%
8949 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8950 }%

```

The first use full form and the inline full form use the short (long) style.

```

8951 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8952 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8953 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8954 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8955 }%
8956 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8957 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8958 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8959 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8960 }%
8961 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8962 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8963 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8964 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8965 }%
8966 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8967 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8968 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8969 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8970 }%
8971 }

```

footnote-sm Backward compatibility:

```

8972 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

8973 \newabbreviationstyle{short-sm-postfootnote}%
8974 {%
8975 \renewcommand*{\CustomAbbreviationFields}{%
8976 name={\glxtrfootnotename},
8977 sort={\the\glsshorttok},

```

```

8978   description={\the\glslongtok},%
8979   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
8980   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
8981   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.

```

8982   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8983     \csdef{glsxtrpostlink\glscategorylabel}{%
8984       \glsxtrifwasfirstuse
8985       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8986         \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
8987         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8988     }%
8989   }%
8990 }%
8991   \glsasattribute{\the\glslabeltok}{regular}%
8992   {%
8993     \glssetattribute{\the\glslabeltok}{regular}{false}%
8994   }%
8995   {}%
8996 }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

8997   \renewcommand*{\glsxtrsetupfulldefs}{%
8998     \let\glsxtrifwasfirstuse\@secondoftwo
8999   }%
9000 }%
9001 {%
9002   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9003   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9004   \renewcommand*\{abbrvpluralsuffix}{\protect\glsxtrmsuffix}%
9005   \renewcommand*\{glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9006   \renewcommand*\{glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9007   \renewcommand*{\glsxtrfullformat}[2]{%
9008     \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9009     \ifglsxtrininsertinside\else##2\fi
9010   }%
9011   \renewcommand*\{glsxtrfullplformat}[2]{%
9012     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9013     \ifglsxtrininsertinside\else##2\fi
9014   }%
9015   \renewcommand*\{Glsxtrfullformat}[2]{%
9016     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9017     \ifglsxtrininsertinside\else##2\fi

```

```

9018 }%
9019 \renewcommand*{\Glsxtrfullplformat}[2]{%
9020   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9021   \ifglsxtrininsertinside\else##2\fi
9022 }%

```

The first use full form and the inline full form use the short (long) style.

```

9023 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9024   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9025   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9026   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9027 }%
9028 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9029   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9030   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9031   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9032 }%
9033 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9034   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9035   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9036   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9037 }%
9038 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9039   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9040   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9041   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9042 }%
9043 }

```

postfootnote-sm Backward compatibility:

```

9044 \@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

### 1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9045 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

9046 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glsxtremsuffix`

```

9047 \newcommand*{\glsxtremsuffix}{\glsxtrabbrvpluralsuffix}

```

`\glsfirstlongemfont`

Only used by the “long-em” styles.

```

9048 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{##1}}%

```

`\glslongemfont` Only used by the “long-em” styles.

```
9049 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9050 \newabbreviationstyle{long-short-em}%
9051 {%
9052   \renewcommand*{\CustomAbbreviationFields}{%
9053     name={\glsxtrlongshortname},
9054     sort={\the\glsshorttok},
9055     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9056       \protect\glsxtrfullsep{\the\glslabeltok}%
9057       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9058     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9059       \protect\glsxtrfullsep{\the\glslabeltok}%
9060       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9061     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9062     description={\the\glslongtok}}%
9063   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9064     \glsattribute{\the\glslabeltok}{regular}%
9065     {%
9066       \glssetattribute{\the\glslabeltok}{regular}{false}%
9067     }%
9068   }%
9069 }%
9070 }%
9071 {%
9072   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9073   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9074   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9075   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9076   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

Use the default long fonts.

```
9075 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9076 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9077 \renewcommand*{\glsxtrfullformat}[2]{%
9078   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9079   \ifglsxtrininsertinside\else##2\fi
9080   \glsxtrfullsep{##1}%
9081   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9082 }%
9083 \renewcommand*{\glsxtrfullplformat}[2]{%
9084   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9085   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9086   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9087 }%
9088 \renewcommand*{\Glsxtrfullformat}[2]{%
9089   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9090   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9091   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9092 }
```

```

9092 }%
9093 \renewcommand*{\Glsxtrfullplformat}[2]{%
9094   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9095   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9096   \glsxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9097 }%
9098 }

```

g-short-em-desc

```

9099 \newabbreviationstyle{long-short-em-desc}%
9100 {%
9101   \renewcommand*{\CustomAbbreviationFields}{%
9102     name={\glsxtrlongshortdescname},
9103     sort={\glsxtrlongshortdescsort},%
9104     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9105       \protect\glsxtrfullsep{\the\glslabeltok}%
9106       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9107     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9108       \protect\glsxtrfullsep{\the\glslabeltok}%
9109       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9110     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9111     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9112   }%

```

Unset the regular attribute if it has been set.

```

9113 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9114   \glshasattribute{\the\glslabeltok}{regular}%
9115   {%
9116     \glissetattribute{\the\glslabeltok}{regular}{false}%
9117   }%
9118   {}%
9119 }%
9120 }%
9121 {%

```

As long-short-em style:

```

9122 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9123 }

```

ong-em-short-em

```

9124 \newabbreviationstyle{long-em-short-em}%
9125 {%
9126   \glslongemfont is used in the description since \glsdesc doesn't set the style.
9127   \renewcommand*{\CustomAbbreviationFields}{%
9128     name={\glsxtrlongshortname},
9129     sort={\the\glsshorttok},
9130     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9131       \protect\glsxtrfullsep{\the\glslabeltok}%
9132       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%

```

```

9132 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}}%
9133 \protect\glsextrfullsep{\the\glslabeltok}}%
9134 \glsextrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9135 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9136 description={\protect\glsfirstlongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9137 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9138 \glshasattribute{\the\glslabeltok}{regular}}%
9139 {%
9140 \glssetattribute{\the\glslabeltok}{regular}{false}}%
9141 }%
9142 {}%
9143 }%
9144 }%
9145 {%
9146 \renewcommand*{\abbrvpluralsuffix}{\protect\glsextremsuffix}%
9147 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9148 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9149 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9150 \renewcommand*{\glslongfont}[1]{\glsfirstlongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9151 \renewcommand*{\glsextrfullformat}[2]{%
9152 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsextrininsertinside##2\fi}%
9153 \ifglsextrininsertinside\else##2\fi
9154 \glsextrfullsep{##1}}%
9155 \glsextrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9156 }%
9157 \renewcommand*{\glsextrfullplformat}[2]{%
9158 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsextrininsertinside##2\fi}%
9159 \ifglsextrininsertinside\else##2\fi\glsextrfullsep{##1}}%
9160 \glsextrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9161 }%
9162 \renewcommand*{\Glsxtrfullformat}[2]{%
9163 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsextrininsertinside##2\fi}%
9164 \ifglsextrininsertinside\else##2\fi\glsextrfullsep{##1}}%
9165 \glsextrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9166 }%
9167 \renewcommand*{\Glsxtrfullplformat}[2]{%
9168 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsextrininsertinside##2\fi}%
9169 \ifglsextrininsertinside\else##2\fi\glsextrfullsep{##1}}%
9170 \glsextrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9171 }%
9172 }

```

m-short-em-desc

```

9173 \newabbreviationstyle{long-em-short-em-desc}%
9174 {%

```



```

9175 \renewcommand*{\CustomAbbreviationFields}{%
9176   name={\glxtrlongshortdescname},
9177   sort={\glxtrlongshortdescsort},%
9178   first={\protect\glsfirstlongemfont{\the\glslongtok}%
9179     \protect\glxtrfullsep{\the\glslabeltok}%
9180     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9181   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9182     \protect\glxtrfullsep{\the\glslabeltok}%
9183     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9184   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9185   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9186 }%

```

Unset the regular attribute if it has been set.

```

9187 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9188   \glshasattribute{\the\glslabeltok}{regular}%
9189   {%
9190     \glissetattribute{\the\glslabeltok}{regular}{false}%
9191   }%
9192   {}%
9193 }%
9194 }%
9195 {%
9196   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9197 }

```

short-em-long Now the short (long) version

```

9198 \newabbreviationstyle{short-em-long}%
9199 {%
9200   \renewcommand*{\CustomAbbreviationFields}{%
9201     name={\glxtrshortlongname},
9202     sort={\the\glsshorttok},
9203     description={\the\glslongtok},%
9204     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9205       \protect\glxtrfullsep{\the\glslabeltok}%
9206       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9207     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9208       \protect\glxtrfullsep{\the\glslabeltok}%
9209       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9210     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9211 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9212   \glshasattribute{\the\glslabeltok}{regular}%
9213   {%
9214     \glissetattribute{\the\glslabeltok}{regular}{false}%
9215   }%
9216   {}%
9217 }%
9218 }%

```

9219 {%

Mostly as short-long style:

```
9220 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9221 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9222 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9223 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9224 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9225 \renewcommand*{\glsxtrfullformat}[2]{%
9226   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9227   \ifglsxtrininsertinside\else##2\fi
9228   \glsxtrfullsep{##1}%
9229   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9230 }%
9231 \renewcommand*{\glsxtrfullplformat}[2]{%
9232   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9233   \ifglsxtrininsertinside\else##2\fi
9234   \glsxtrfullsep{##1}%
9235   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9236 }%
9237 \renewcommand*{\Glsxtrfullformat}[2]{%
9238   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9239   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9240   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9241 }%
9242 \renewcommand*{\Glsxtrfullplformat}[2]{%
9243   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9244   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9245   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9246 }%
9247 }
```

rt-em-long-desc As before but user provides description

```
9248 \newabbreviationstyle{short-em-long-desc}%
9249 {%
9250   \renewcommand*{\CustomAbbreviationFields}{%
9251     name={\glsxtrshortlongdescname},
9252     sort={\glsxtrshortlongdescsort},
9253     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9254     \protect\glsxtrfullsep{\the\glslabeltok}%
9255     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9256     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9257     \protect\glsxtrfullsep{\the\glslabeltok}%
9258     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9259     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9260     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
9261   }%
```

Unset the regular attribute if it has been set.

```

9262 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9263   \glshasattribute{\the\glslabeltok}{regular}%
9264   {%
9265     \glissetattribute{\the\glslabeltok}{regular}{false}%
9266   }%
9267   {}%
9268 }%
9269 }%
9270 {%
9271   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9272 }

```

hort-em-long-em

```

9273 \newabbreviationstyle{short-em-long-em}%
9274 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9275 \renewcommand*{\CustomAbbreviationFields}{%
9276   name={\glsxtrshortlongname},
9277   sort={\the\glsshorttok},
9278   description={\protect\glslongemfont{\the\glslongtok}},%
9279   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9280     \protect\glsxtrfullsep{\the\glslabeltok}%
9281     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9282   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9283     \protect\glsxtrfullsep{\the\glslabeltok}%
9284     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9285   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}

```

Unset the regular attribute if it has been set.

```

9286 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9287   \glshasattribute{\the\glslabeltok}{regular}%
9288   {%
9289     \glissetattribute{\the\glslabeltok}{regular}{false}%
9290   }%
9291   {}%
9292 }%
9293 }%
9294 {%
9295   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
9296   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9297   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9298   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9299   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9300 \renewcommand*{\glsxtrfullformat}[2]{%
9301   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9302   \ifglsxtrininsertinside\else##2\fi
9303   \glsxtrfullsep{##1}%

```

```

9304   \glstrparen{\glfirstlongemfont{\glaccesslong{##1}}}%
9305 }%
9306 \renewcommand*{\glstrfullplformat}[2]{%
9307   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9308   \ifglstrinsertinside\else##2\fi
9309   \glstrfullsep{##1}%
9310   \glstrparen{\glfirstlongemfont{\glaccesslongpl{##1}}}%
9311 }%
9312 \renewcommand*{\Glsxtrfullformat}[2]{%
9313   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9314   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9315   \glstrparen{\glfirstlongemfont{\glaccesslong{##1}}}%
9316 }%
9317 \renewcommand*{\Glsxtrfullplformat}[2]{%
9318   \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9319   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9320   \glstrparen{\glfirstlongemfont{\glaccesslongpl{##1}}}%
9321 }%
9322 }

```

em-long-em-desc

```

9323 \newabbreviationstyle{short-em-long-em-desc}%
9324 {%
9325   \renewcommand*{\CustomAbbreviationFields}{%
9326     name={\glstrshortlongdescname},%
9327     sort={\glstrshortlongdescsort},%
9328     first={\protect\glfirstabbrvemfont{\the\glsshorttok}}%
9329     \protect\glstrfullsep{\the\glslabeltok}%
9330     \glstrparen{\protect\glfirstlongemfont{\the\glslongtok}}},%
9331     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}}%
9332     \protect\glstrfullsep{\the\glslabeltok}%
9333     \glstrparen{\protect\glfirstlongemfont{\the\glslongpltok}}},%
9334     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9335     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9336 }%

```

Unset the regular attribute if it has been set.

```

9337 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9338   \glshasattribute{\the\glslabeltok}{regular}%
9339   {%
9340     \glissetattribute{\the\glslabeltok}{regular}{false}%
9341   }%
9342   {}%
9343 }%
9344 }%
9345 {%
9346   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9347 }

```

short-em

```

9348 \newabbreviationstyle{short-em}%
9349 {%
9350   \renewcommand*{\CustomAbbreviationFields}{%
9351     name={\glxtrshortnolongname},
9352     sort={\the\glsshorttok},
9353     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9354     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9355     text={\protect\glsabbrvemfont{\the\glsshorttok}},
9356     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9357     description={\the\glslongtok}}%
9358   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9359     \glssetAttribute{\the\glslabeltok}{regular}{true}}%
9360 }%
9361 {%
9362   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9363   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9364   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9365   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9366   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9367   \renewcommand*{\glxtrinlinefullformat}[2]{%
9368     \protect\glsfirstabbrvemfont{\glsaccesssshort{##1}}%
9369     \ifglxtrininsertinside##2\fi}%
9370   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9371   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9372 }%
9373   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9374     \protect\glsfirstabbrvemfont{\glsaccesssshortpl{##1}}%
9375     \ifglxtrininsertinside##2\fi}%
9376   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9377   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9378 }%
9379   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9380     \protect\glsfirstabbrvemfont{\Glsaccesssshort{##1}}%
9381     \ifglxtrininsertinside##2\fi}%
9382   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9383   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9384 }%
9385   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9386     \protect\glsfirstabbrvemfont{\Glsaccesssshortpl{##1}}%
9387     \ifglxtrininsertinside##2\fi}%
9388   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9389   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9390 }%

```

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.

```

9391   \renewcommand*{\glxtrfullformat}[2]{%

```

```

9392 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9393 \ifglxtrinsertinside\else##2\fi
9394 }%
9395 \renewcommand*{\glxtrfullplformat}[2]{%
9396 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9397 \ifglxtrinsertinside\else##2\fi
9398 }%
9399 \renewcommand*{\Glsxtrfullformat}[2]{%
9400 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9401 \ifglxtrinsertinside\else##2\fi
9402 }%
9403 \renewcommand*{\Glsxtrfullplformat}[2]{%
9404 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9405 \ifglxtrinsertinside\else##2\fi
9406 }%
9407 }

```

short-em-nolong

```

9408 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9409 \newabbreviationstyle{short-em-desc}%
9410 {%
9411 \renewcommand*{\CustomAbbreviationFields}{%
9412 name={\glxtrshortdescname},
9413 sort={\the\glsshorttok},
9414 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
9415 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
9416 text={\protect\glsabbrvemfont{\the\glsshorttok}},
9417 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9418 description={\the\glslongtok}}%
9419 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9420 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9421 }%
9422 {%
9423 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9424 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9425 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9426 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9427 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9428 \renewcommand*{\glxtrinlinefullformat}[2]{%
9429 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9430 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9431 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9432 }%
9433 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9434 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9435 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9436 \glstrparen{\glfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9437 }%
9438 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9439 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9440 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9441 \glstrparen{\glfirstlongdefaultfont{\glsaccesslong{##1}}}%
9442 }%
9443 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9444 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9445 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9446 \glstrparen{\glfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9447 }%

```

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.

```

9448 \renewcommand*{\glxtrfullformat}[2]{%
9449 \glfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9450 \ifglxtrinsertinside\else##2\fi
9451 }%
9452 \renewcommand*{\glxtrfullplformat}[2]{%
9453 \glfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9454 \ifglxtrinsertinside\else##2\fi
9455 }%
9456 \renewcommand*{\Glsxtrfullformat}[2]{%
9457 \glfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9458 \ifglxtrinsertinside\else##2\fi
9459 }%
9460 \renewcommand*{\Glsxtrfullplformat}[2]{%
9461 \glfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9462 \ifglxtrinsertinside\else##2\fi
9463 }%
9464 }

```

-em-nolong-desc

```

9465 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9466 \newabbreviationstyle{nolong-short-em}%
9467 {%
9468 \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9469 }%
9470 {%
9471 \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9472 \renewcommand*{\glxtrinlinefullformat}[2]{%
9473 \protect\glfirstlongdefaultfont{\glsaccesslong{##1}%
9474 \ifglxtrinsertinside##2\fi}%
9475 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9476 \glstrparen{\glfirstabbrvemfont{\glsaccessshort{##1}}}%

```

```

9477 }%
9478 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9479   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9480     \ifglxtrininsertinside##2\fi}%
9481   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9482   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9483 }%
9484 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9485   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9486     \ifglxtrininsertinside##2\fi}%
9487   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9488   \glxtrparen{\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9489 }%
9490 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9491   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9492     \ifglxtrininsertinside##2\fi}%
9493   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9494   \glxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9495 }%
9496 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9497 \newabbreviationstyle{long-noshort-em}%
9498 {%
9499   \renewcommand*{\CustomAbbreviationFields}{%
9500     name={\glxtrlongnoshortname},
9501     sort={\the\glsshorttok},
9502     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9503     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9504     text={\protect\glslongdefaultfont{\the\glslongtok}},
9505     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9506     description={\the\glslongtok}%
9507   }%
9508   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9509     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9510 }%
9511 {%
9512   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9513   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9514   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9515   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9516   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9517 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9518   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9519   \ifglxtrininsertinside \else##2\fi
9520 }%
9521 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9522   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

```



```

9523 \ifglxtrinsertinside \else##2\fi
9524 }%
9525 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9526 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9527 \ifglxtrinsertinside \else##2\fi
9528 }%
9529 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9530 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9531 \ifglxtrinsertinside \else##2\fi
9532 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9533 \renewcommand*{\glxtrinlinefullformat}[2]{%
9534 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9535 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9536 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9537 }%
9538 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9539 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9540 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9541 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9542 }%
9543 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9544 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9545 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9546 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9547 }%
9548 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9549 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9550 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9551 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9552 }%

```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```

9553 \renewcommand*{\glxtrfullformat}[2]{%
9554 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9555 \ifglxtrinsertinside\else##2\fi
9556 }%
9557 \renewcommand*{\glxtrfullplformat}[2]{%
9558 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9559 \ifglxtrinsertinside\else##2\fi
9560 }%
9561 \renewcommand*{\Glsxtrfullformat}[2]{%
9562 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9563 \ifglxtrinsertinside\else##2\fi
9564 }%
9565 \renewcommand*{\Glsxtrfullplformat}[2]{%
9566 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9567 \ifglxtrinsertinside\else##2\fi

```

```

9568 }%
9569 }

```

long-em    Backward compatibility:

```

9570 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em    The short form is explicitly invoked through commands like \glsshort.

```

9571 \newabbreviationstyle{long-em-noshort-em}%
9572 {%
9573   \renewcommand*{\CustomAbbreviationFields}{%
9574     name={\glxtrlongnoshortname},
9575     sort={\the\glsshorttok},
9576     first={\protect\glslongemfont{\the\glslongtok}},
9577     firstplural={\protect\glslongemfont{\the\glslongpltok}},
9578     text={\protect\glslongemfont{\the\glslongtok}},
9579     plural={\protect\glslongemfont{\the\glslongpltok}},%
9580     description={\protect\glslongemfont{\the\glslongtok}}%
9581   }%
9582   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9583     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9584 }%
9585 {%
9586   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9587   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9588   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9589   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongemfont{##1}}%
9590   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9591 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9592   \glslongemfont{\glaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9593   \ifglxtrininsertinside \else##2\fi
9594 }%
9595 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9596   \glslongemfont{\glaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9597   \ifglxtrininsertinside \else##2\fi
9598 }%
9599 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9600   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9601   \ifglxtrininsertinside \else##2\fi
9602 }%
9603 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9604   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9605   \ifglxtrininsertinside \else##2\fi
9606 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9607 \renewcommand*{\glxtrinlinefullformat}[2]{%
9608   \glslongemfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9609   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9610 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9611 }%
9612 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9613 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9614 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9615 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9616 }%
9617 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9618 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9619 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9620 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9621 }%
9622 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9623 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9624 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9625 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9626 }%

```

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.

```

9627 \renewcommand*{\glsxtrfullformat}[2]{%
9628 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9629 \ifglsxtrinsertinside\else##2\fi
9630 }%
9631 \renewcommand*{\glsxtrfullplformat}[2]{%
9632 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9633 \ifglsxtrinsertinside\else##2\fi
9634 }%
9635 \renewcommand*{\Glsxtrfullformat}[2]{%
9636 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9637 \ifglsxtrinsertinside\else##2\fi
9638 }%
9639 \renewcommand*{\Glsxtrfullplformat}[2]{%
9640 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9641 \ifglsxtrinsertinside\else##2\fi
9642 }%
9643 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

9644 \newabbreviationstyle{long-em-noshort-em-noreg}%
9645 {%
9646 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%

```

Unset the regular attribute if it has been set.

```

9647 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9648 \glshasattribute{\the\glslabeltok}{regular}%
9649 {%
9650 \glssetattribute{\the\glslabeltok}{regular}{false}%
9651 }%
9652 }%

```

```

9653 }%
9654 }%
9655 {%
9656 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
9657 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9658 \newabbreviationstyle{long-noshort-em-desc}%
9659 {%
9660 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9661 }%
9662 {%
9663 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9664 \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9665 \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
9666 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9667 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9668 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9669 \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9670 \ifglsxtrininsertinside \else##2\fi
9671 }%
9672 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9673 \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9674 \ifglsxtrininsertinside \else##2\fi
9675 }%
9676 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9677 \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9678 \ifglsxtrininsertinside \else##2\fi
9679 }%
9680 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9681 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9682 \ifglsxtrininsertinside \else##2\fi
9683 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9684 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9685 \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9686 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9687 \glsxtrparen{\protect\glsfirstabrvemfont{\glssaccessshort{##1}}}%
9688 }%
9689 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9690 \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9691 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9692 \glsxtrparen{\protect\glsfirstabrvemfont{\glssaccessshortpl{##1}}}%
9693 }%
9694 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9695 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%

```

```

9696     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9697     \glxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9698 }%
9699 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9700     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9701     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9702     \glxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9703 }%

```

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.

```

9704 \renewcommand*{\glxtrfullformat}[2]{%
9705     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9706     \ifglxtrinsertinside\else##2\fi
9707 }%
9708 \renewcommand*{\glxtrfullplformat}[2]{%
9709     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9710     \ifglxtrinsertinside\else##2\fi
9711 }%
9712 \renewcommand*{\Glsxtrfullformat}[2]{%
9713     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9714     \ifglxtrinsertinside\else##2\fi
9715 }%
9716 \renewcommand*{\Glsxtrfullplformat}[2]{%
9717     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9718     \ifglxtrinsertinside\else##2\fi
9719 }%
9720 }

```

long-desc-em Backward compatibility:

```

9721 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}

```

noshort-em-desc The short form is explicitly invoked through commands like `\glssshort`. The long form is emphasized.

```

9722 \newabbreviationstyle{long-em-noshort-em-desc}%
9723 {%
9724 \renewcommand*{\CustomAbbreviationFields}{%
9725     name={\glxtrlongnoshortdescname},
9726     sort={\the\glslongtok},
9727     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9728     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9729     text={\glslongemfont{\the\glslongtok}},
9730     plural={\glslongemfont{\the\glslongpltok}}%
9731 }%
9732 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9733     \glsssetattribute{\the\glslabeltok}{regular}{true}}%
9734 }%
9735 {%
9736 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%

```

```

9737 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9738 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9739 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
9740 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9741 \renewcommand*\glsxtrsubsequentfmt}[2]{%
9742   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9743   \ifglsxtrininsertinside \else##2\fi
9744 }%
9745 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
9746   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9747   \ifglsxtrininsertinside \else##2\fi
9748 }%
9749 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
9750   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9751   \ifglsxtrininsertinside \else##2\fi
9752 }%
9753 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
9754   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9755   \ifglsxtrininsertinside \else##2\fi
9756 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9757 \renewcommand*\glsxtrinlinefullformat}[2]{%
9758   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9759   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9760   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9761 }%
9762 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9763   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9764   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9765   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9766 }%
9767 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9768   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9769   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9770   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9771 }%
9772 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9773   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9774   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9775   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9776 }%

```

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.

```

9777 \renewcommand*\glsxtrfullformat}[2]{%
9778   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9779   \ifglsxtrininsertinside\else##2\fi
9780 }%

```

```

9781 \renewcommand*{\glstrfullplformat}[2]{%
9782   \glstrfirstlongemfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9783   \ifglstrinsertinside\else##2\fi
9784 }%
9785 \renewcommand*{\Glsxtrfullformat}[2]{%
9786   \glstrfirstlongemfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
9787   \ifglstrinsertinside\else##2\fi
9788 }%
9789 \renewcommand*{\Glsxtrfullplformat}[2]{%
9790   \glstrfirstlongemfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
9791   \ifglstrinsertinside\else##2\fi
9792 }%
9793 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

9794 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9795 {%
9796   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
9797   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9798     \glshasattribute{\the\glslabeltok}{regular}%
9799     {%
9800       \glsssetattribute{\the\glslabeltok}{regular}{false}%
9801     }%
9802   }%
9803 }%
9804 }%
9805 {%
9806   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9807 }

```

ort-em-footnote

```

9808 \newabbreviationstyle{short-em-footnote}%
9809 {%
9810   \renewcommand*{\CustomAbbreviationFields}{%
9811     name={\glstrfootnotename},
9812     sort={\the\glssshorttok},
9813     description={\the\glslongtok},%
9814     first={\protect\glstrfirstabbrvemfont{\the\glssshorttok}%
9815       \protect\glstrabbrvfootnote{\the\glslabeltok}%
9816       {\protect\glstrfirstlongfootnotefont{\the\glslongtok}}},%
9817     firstplural={\protect\glstrfirstabbrvemfont{\the\glssshortpltok}%
9818       \protect\glstrabbrvfootnote{\the\glslabeltok}%
9819       {\protect\glstrfirstlongfootnotefont{\the\glslongpltok}}},%
9820     plural={\protect\glstrabbrvemfont{\the\glssshortpltok}}}%
9821   \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

9822 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9823 \glsattribute{\the\glslabeltok}{regular}%
9824 {%
9825 \glssetattribute{\the\glslabeltok}{regular}{false}%
9826 }%
9827 {}%
9828 }%
9829 }%
9830 {%
9831 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9832 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9833 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9834 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9835 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9836 \renewcommand*{\glsxtrfullformat}[2]{%
9837 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9838 \ifglsxtrininsertinside\else##2\fi
9839 \protect\glsxtrabbrvfootnote{##1}%
9840 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9841 }%
9842 \renewcommand*{\glsxtrfullplformat}[2]{%
9843 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9844 \ifglsxtrininsertinside\else##2\fi
9845 \protect\glsxtrabbrvfootnote{##1}%
9846 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9847 }%
9848 \renewcommand*{\Glsxtrfullformat}[2]{%
9849 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9850 \ifglsxtrininsertinside\else##2\fi
9851 \protect\glsxtrabbrvfootnote{##1}%
9852 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9853 }%
9854 \renewcommand*{\Glsxtrfullplformat}[2]{%
9855 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9856 \ifglsxtrininsertinside\else##2\fi
9857 \protect\glsxtrabbrvfootnote{##1}%
9858 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9859 }%

```

The first use full form and the inline full form use the short (long) style.

```

9860 \renewcommand*{\glsxtrininlinefullformat}[2]{%
9861 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9862 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9863 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9864 }%
9865 \renewcommand*{\glsxtrininlinefullplformat}[2]{%
9866 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9867 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```



```

9868 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9869 }%
9870 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9871 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9872 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9873 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9874 }%
9875 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9876 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9877 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9878 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9879 }%
9880 }

```

footnote-em Backward compatibility:

```

9881 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9882 \newabbreviationstyle{short-em-postfootnote}%
9883 {%
9884 \renewcommand*{\CustomAbbreviationFields}{%
9885 name={\glxtrfootnotename},
9886 sort={\the\glsshorttok},
9887 description={\the\glslongtok},%
9888 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
9889 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
9890 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

9891 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9892 \csdef{glxtrpostlink\glscategorylabel}{%
9893 \glxtrifwasfirstuse
9894 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9895 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9896 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
9897 }%
9898 {}%
9899 }%
9900 \glshasattribute{\the\glslabeltok}{regular}%
9901 {%
9902 \glissetattribute{\the\glslabeltok}{regular}{false}%
9903 }%
9904 {}%
9905 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9906 \renewcommand*\glxtrsetupfulldefs}{%
9907 \let\glxtrifwasfirstuse\@secondoftwo
9908 }%
9909 }%
9910 {%
9911 \renewcommand*\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9912 \renewcommand*\glxabbrvfont[1]{\glxabbrvemfont{##1}}%
9913 \renewcommand*\glxfirstabbrvfont[1]{\glxfirstabbrvemfont{##1}}%
9914 \renewcommand*\glxfirstlongfont[1]{\glxfirstlongfootnotefont{##1}}%
9915 \renewcommand*\glxlongfont[1]{\glxlongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9916 \renewcommand*\glxtrfullformat}[2]{%
9917 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9918 \ifglxtrininsertinside\else##2\fi
9919 }%
9920 \renewcommand*\glxtrfullplformat}[2]{%
9921 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9922 \ifglxtrininsertinside\else##2\fi
9923 }%
9924 \renewcommand*\Glxtrfullformat}[2]{%
9925 \glxfirstabbrvemfont{\Glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9926 \ifglxtrininsertinside\else##2\fi
9927 }%
9928 \renewcommand*\Glxtrfullplformat}[2]{%
9929 \glxfirstabbrvemfont{\Glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9930 \ifglxtrininsertinside\else##2\fi
9931 }%

```

The first use full form and the inline full form use the short (long) style.

```

9932 \renewcommand*\glxtrininlinefullformat}[2]{%
9933 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9934 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9935 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslong{##1}}}%
9936 }%
9937 \renewcommand*\glxtrininlinefullplformat}[2]{%
9938 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9939 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9940 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslongpl{##1}}}%
9941 }%
9942 \renewcommand*\Glxtrininlinefullformat}[2]{%
9943 \glxfirstabbrvemfont{\Glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9944 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9945 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslong{##1}}}%
9946 }%
9947 \renewcommand*\Glxtrininlinefullplformat}[2]{%
9948 \glxfirstabbrvemfont{\Glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9949 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9950     \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9951   }%
9952 }

```

postfootnote-em    Backward compatibility:

```

9953 \@glsxtr@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:

glsxtruserfield    Default is the useri field.

```

9954 \newcommand*{\glsxtruserfield}{useri}

```

glsxtruserparen    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.

```

9955 \ifdef\glscurrentfieldvalue
9956 {
9957   \newcommand*{\glsxtruserparen}[2]{%
9958     \glsxtrfullsep{#2}%
9959     \glsxtrparen
9960     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9961   }
9962 }
9963 {
9964   \newcommand*{\glsxtruserparen}[2]{%
9965     \glsxtrfullsep{#2}%
9966     \glsxtrparen
9967     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9968   }
9969 }

```

Font used for short form:

lsabbrvuserfont

```

9970 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

9971 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```

9972 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}

```

Font used for long form on first use:

rstlonguserfont

```
9973 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
9974 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
9975 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
9976 \newabbreviationstyle{long-short-user}%
9977 {%
9978   \renewcommand*{\CustomAbbreviationFields}{%
9979     name={\glsxtrlongshortname},
9980     sort={\the\glsshorttok},
9981     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9982       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
9983     {\the\glslabeltok}},%
9984     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9985       \protect\glsxtruserparen
9986       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9987     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9988     description={\protect\glsuserdescription{\the\glslongtok}%
9989       {\the\glslabeltok}}}%
9989 }
```

Unset the regular attribute if it has been set.

```
9990 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9991   \glshasattribute{\the\glslabeltok}{regular}%
9992   {%
9993     \glissetattribute{\the\glslabeltok}{regular}{false}%
9994   }%
9995   {}%
9996 }%
9997 }%
9998 }
```

In case the user wants to mix and match font styles, these are redefined here.

```
9999 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10000 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10001 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10002 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10003 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
10004 \renewcommand*{\glsxtrfullformat}[2]{%
10005   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}}%
```

```

10006     \ifglxtrinsertinside\else##2\fi
10007     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10008 }%
10009 \renewcommand*{\glxtrfullplformat}[2]{%
10010     \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10011     \ifglxtrinsertinside\else##2\fi
10012     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10013 }%
10014 \renewcommand*{\Glsxtrfullformat}[2]{%
10015     \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10016     \ifglxtrinsertinside\else##2\fi
10017     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10018 }%
10019 \renewcommand*{\Glsxtrfullplformat}[2]{%
10020     \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10021     \ifglxtrinsertinside\else##2\fi
10022     \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10023 }%
10024 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10025 \newabbreviationstyle{long-postshort-user}%
10026 {%
10027     \renewcommand*{\CustomAbbreviationFields}{%
10028         name={\glxtrlongshortname},
10029         sort={\the\glssshorttok},
10030         first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10031         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10032         plural={\protect\glssabbrvuserfont{\the\glssshortpltok}},%
10033         description={\protect\glssuserdescription{\the\glslongtok}%
10034             {\the\glslabeltok}}}%
10035     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10036         \csdef{glxtrpostlink\glscategorylabel}{%
10037             \glxtrifwasfirstuse
10038             {%
10039                 \glxtruserparen
10040                 {\glsfirstabbrvuserfont{\glssentryshort{\glslabel}}}%
10041                 {\glslabel}%
10042             }%
10043             {}%
10044         }%
10045         \glshasattribute{\the\glslabeltok}{regular}%
10046         {%
10047             \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10048         }%
10049         {}%
10050     }%
10051 }%
10052 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```
10053 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10054 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10055 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10056 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10057 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
10058 \renewcommand*{\glxtrfullformat}[2]{%
10059   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10060   \ifglxtrininsertinside\else##2\fi
10061 }%
10062 \renewcommand*{\glxtrfullplformat}[2]{%
10063   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10064   \ifglxtrininsertinside\else##2\fi
10065 }%
10066 \renewcommand*{\Glsxtrfullformat}[2]{%
10067   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10068   \ifglxtrininsertinside\else##2\fi
10069 }%
10070 \renewcommand*{\Glsxtrfullplformat}[2]{%
10071   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10072   \ifglxtrininsertinside\else##2\fi
10073 }%
```

In-line format:

```
10074 \renewcommand*{\glxtrinlinefullformat}[2]{%
10075   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10076   \ifglxtrininsertinside\else##2\fi
10077   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10078 }%
10079 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10080   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10081   \ifglxtrininsertinside\else##2\fi
10082   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10083 }%
10084 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10085   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10086   \ifglxtrininsertinside\else##2\fi
10087   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10088 }%
10089 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10090   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10091   \ifglxtrininsertinside\else##2\fi
10092   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10093 }%
10094 }
```

ortuserdesname

```
10095 \newcommand*{\glxtrlongshortuserdesname}{%
```

```

10096 \protect\glslonguserfont{\the\glslongtok}%
10097 \protect\glsxtruserparen
10098 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10099 }

```

**short-user-desc** Like long-postshort-user but the user supplies the description.

```

10100 \newabbreviationstyle{long-postshort-user-desc}%
10101 {%
10102   \renewcommand*{\CustomAbbreviationFields}{%
10103     name={\glsxtrlongshortuserdescname},
10104     sort={\the\glslongtok},
10105     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10106     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10107     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10108     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
10109   }%
10110   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10111     \csdef{glsxtrpostlink\glscategorylabel}{%
10112       \glsxtrifwasfirstuse
10113       {%
10114         \glsxtruserparen
10115         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10116         {\glslabel}%
10117       }%
10118     }%
10119   }%
10120   \glsattribute{\the\glslabeltok}{regular}%
10121   {%
10122     \glssetattribute{\the\glslabeltok}{regular}{false}%
10123   }%
10124   {}%
10125 }%
10126 }%
10127 {%
10128   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10129 }

```

**t-postlong-user** Like short-long-user but defers the parenthetical matter to after the link.

```

10130 \newabbreviationstyle{short-postlong-user}%
10131 {%
10132   \renewcommand*{\CustomAbbreviationFields}{%
10133     name={\glsxtrshortlongname},
10134     sort={\the\glsshorttok},
10135     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10136     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10137     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10138     description={\protect\glsuserdescription{\the\glslongtok}}%
10139     {\the\glslabeltok}}%

```

```

10140 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10141   \csdef{glxtrpostlink\glscategorylabel}{%
10142     \glxtrifwasfirstuse
10143     {%
10144       \glxtruserparen
10145       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10146       {\glslabel}%
10147     }%
10148   }%
10149 }%
10150 \glshasattribute{\the\glslabeltok}{regular}%
10151 {%
10152   \glissetattribute{\the\glslabeltok}{regular}{false}%
10153 }%
10154 {}%
10155 }%
10156 }%
10157 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10158 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10159 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10160 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10161 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10162 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10163 \renewcommand*{\glxtrfullformat}[2]{%
10164   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10165   \ifglxtrininsertinside\else##2\fi
10166 }%
10167 \renewcommand*{\glxtrfullplformat}[2]{%
10168   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10169   \ifglxtrininsertinside\else##2\fi
10170 }%
10171 \renewcommand*{\Glsxtrfullformat}[2]{%
10172   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10173   \ifglxtrininsertinside\else##2\fi
10174 }%
10175 \renewcommand*{\Glsxtrfullplformat}[2]{%
10176   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10177   \ifglxtrininsertinside\else##2\fi
10178 }%

```

In-line format:

```

10179 \renewcommand*{\glxtrininlinefullformat}[2]{%
10180   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10181   \ifglxtrininsertinside\else##2\fi
10182   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10183 }%

```



```

10184 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10185   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10186   \ifglxtrininsertinside\else##2\fi
10187   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10188 }%
10189 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10190   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10191   \ifglxtrininsertinside\else##2\fi
10192   \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10193 }%
10194 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10195   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10196   \ifglxtrininsertinside\else##2\fi
10197   \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10198 }%
10199 }

```

onguserdescname

```

10200 \newcommand*{\glxtrshortlonguserdescname}{%
10201   \protect\glssabbrvuserfont{\the\glssshorttok}%
10202   \protect\glsxtruserparen
10203     {\protect\glslonguserfont{\the\glslongpltok}}}%
10204     {\the\glslabelltok}%
10205 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10206 \newabbreviationstyle{short-postlong-user-desc}%
10207 {%
10208   \renewcommand*{\CustomAbbreviationFields}{%
10209     name={\glxtrshortlonguserdescname},
10210     sort={\the\glssshorttok},
10211     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10212     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10213     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10214     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10215 }%
10216 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10217   \csdef{glxtrpostlink\glscategorylabel}{%
10218     \glsxtrifwasfirstuse
10219     {%
10220       \glsxtruserparen
10221         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10222         {\glslabel}%
10223     }%
10224     }%
10225 }%
10226 \glshasattribute{\the\glslabelltok}{regular}%
10227 }%

```

```

10228     \glsetattribute{\the\glslabeltok}{regular}{false}%
10229   }%
10230   {}%
10231 }%
10232 }%
10233 {%
10234   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10235 }

```

short-user-desc

```

10236 \newabbreviationstyle{long-short-user-desc}%
10237 {%
10238   \renewcommand*{\CustomAbbreviationFields}{%
10239     name={\glxtrlongshortuserdescname},
10240     sort={\glxtrlongshortdescsort},%

10241     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10242       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10243     {\the\glslabeltok}},%
10244     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10245       \protect\glxtruserparen
10246       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10247     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10248     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10249   }%

```

Unset the regular attribute if it has been set.

```

10250   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10251     \glshasattribute{\the\glslabeltok}{regular}%
10252     {%
10253       \glsetattribute{\the\glslabeltok}{regular}{false}%
10254     }%
10255     {}%
10256   }%
10257 }%
10258 {%
10259   \GlsXtrUseAbbrStyleFmts{long-short-user}%
10260 }

```

short-long-user

```

10261 \newabbreviationstyle{short-long-user}%
10262 {%

  \glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in
  \glsuserdescription.)

10263   \renewcommand*{\CustomAbbreviationFields}{%
10264     name={\glxtrshortlongname},
10265     sort={\the\glsshorttok},
10266     description={\protect\glsuserdescription{\the\glslongtok}%
10267       {\the\glslabeltok}},%

```

```

10268 first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
10269 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
10270 {\the\glslabeltok}},%
10271 firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}%
10272 \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
10273 {\the\glslabeltok}},%

```

```

10274 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10275 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10276 \glshasattribute{\the\glslabeltok}{regular}}%
10277 {%
10278 \glssetattribute{\the\glslabeltok}{regular}{false}}%
10279 }%
10280 {}%
10281 }%
10282 }%
10283 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10284 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10285 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10286 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10287 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10288 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10289 \renewcommand*{\glsxtrfullformat}[2]{%
10290 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10291 \ifglsxtrinsertinside\else##2\fi
10292 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
10293 }%
10294 \renewcommand*{\glsxtrfullplformat}[2]{%
10295 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10296 \ifglsxtrinsertinside\else##2\fi
10297 \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
10298 }%
10299 \renewcommand*{\Glsxtrfullformat}[2]{%
10300 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
10301 \ifglsxtrinsertinside\else##2\fi
10302 \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10303 }%
10304 \renewcommand*{\Glsxtrfullplformat}[2]{%
10305 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
10306 \ifglsxtrinsertinside\else##2\fi
10307 \glsxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10308 }%
10309 }

```

-long-user-desc

```

10310 \newabbreviationstyle{short-long-user-desc}%
10311 {%
10312   \renewcommand*{\CustomAbbreviationFields}{%
10313     name={\glxtrshortlonguserdescname},
10314     sort={\glxtrshortlongdescsort},%

10315     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
10316       \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10317     {\the\glslabeltok}},%
10318     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
10319       \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10320     {\the\glslabeltok}},%
10321     text={\protect\glsabbrvfont{\the\glsshorttok}},%
10322     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10323   }%

   Unset the regular attribute if it has been set.
10324   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10325     \glshasattribute{\the\glslabeltok}{regular}%
10326     {%
10327       \glissetattribute{\the\glslabeltok}{regular}{false}%
10328     }%
10329   }%
10330 }%
10331 }%
10332 {%
10333   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10334 }

```

### 1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glxtrlong` set `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

`trifhyphenstart` Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```

10335 \newrobustcmd*{\glxtrifhyphenstart}[3]{%
10336   \ifx\glsinsert#1\relax
10337     \expandafter\@glxtrifhyphenstart#1\relax\relax
10338     \@end@glxtrifhyphenstart{#2}{#3}%
10339   \else
10340     \@glxtrifhyphenstart#1\relax\relax\@end@glxtrifhyphenstart{#2}{#3}%
10341   \fi
10342 }

```

trifhyphenstart

```
10343 \def\@glxtrifhyphenstart#1#2\@end@glxtrifhyphenstart#3#4{%
10344   \ifx-#1\relax#3\else #4\fi
10345 }
```

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.

```
10346 \newcommand*\glxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10347  {%
```

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.

```
10348   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%
10349   \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10350   \ifglxtrininsertinside\else{#4}\fi
10351   \glxtrfullsep{#1}%
10352   \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10353     \ifglxtrininsertinside\else{#4}\fi}%
10354 }%
10355 }
```

abbrvhyphenfont

```
10356 \newcommand*\glsabbrvhyphenfont{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
10357 \newcommand*\glsfirstabbrvhyphenfont{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
10358 \newcommand*\glslonghyphenfont{\glslongdefaultfont}%
```

tlonghyphenfont

```
10359 \newcommand*\glsfirstlonghyphenfont{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10360 \newcommand*\glxtrhyphensuffix{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the `keywords` attribute.

```
10361 \newabbreviationstyle{long-hyphen-short-hyphen}%
10362 {%
```

```

10363 \renewcommand*{\CustomAbbreviationFields}{%
10364     name={\glxtrlongshortname},
10365     sort={\the\glsshorttok},
10366     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10367         \protect\glxtrfullsep{\the\glslabeltok}%
10368         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10369     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10370         \protect\glxtrfullsep{\the\glslabeltok}%
10371         \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10372     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10373     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10374 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10375     \glshasattribute{\the\glslabeltok}{regular}%
10376     {%
10377         \glissetattribute{\the\glslabeltok}{regular}{false}%
10378     }%
10379     {}%
10380 }%
10381}%
10382{%
10383 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10384 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
10385 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10386 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10387 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10388 \renewcommand*{\glxtrfullformat}[2]{%
10389     \glxtrlonghyphenshort{##1}{\glssaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
10390 }%
10391 \renewcommand*{\glxtrfullplformat}[2]{%
10392     \glxtrlonghyphenshort{##1}{\glssaccesslongpl{##1}}%
10393     {\glssaccessshortpl{##1}}{##2}%
10394 }%
10395 \renewcommand*{\Glsxtrfullformat}[2]{%
10396     \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\Glsaccessshort{##1}}{##2}%
10397 }%
10398 \renewcommand*{\Glsxtrfullplformat}[2]{%
10399     \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10400     {\Glsaccessshortpl{##1}}{##2}%
10401 }%
10402}

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

10403 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10404 {%
10405     \renewcommand*{\CustomAbbreviationFields}{%
10406         name={\glxtrlongshortdescname},

```

```

10407   sort={\glxtrlongshortdescsort},
10408   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10409     \protect\glxtrfullsep{\the\glslabeltok}%
10410     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10411   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10412     \protect\glxtrfullsep{\the\glslabeltok}%
10413     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10414   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10415   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10416 }%

```

Unset the regular attribute if it has been set.

```

10417 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10418   \glsasattribute{\the\glslabeltok}{regular}%
10419   {%
10420     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10421   }%
10422   {}%
10423 }%
10424 }%
10425 {%
10426   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10427 }

```

onghyphennoshort `\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}`

```

10428 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10429 {%

```

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.

```

10430   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10431     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10432     \ifglxtrininsertinside\else{#3}\fi
10433   }%
10434 }

```

hort-desc-noreg This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```

10435 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10436 {%
10437   \renewcommand*{\CustomAbbreviationFields}{%

```

```

10438   name={\glxtrlongnoshortdescname},
10439   sort={\expandonce\glxtrorglong},
10440   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10441   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10442   plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10443 }%

```

Unset the regular attribute if it has been set.

```

10444 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10445   \glshasattribute{\the\glslabeltok}{regular}%
10446   {%
10447     \glissetattribute{\the\glslabeltok}{regular}{false}%
10448   }%
10449 }%
10450 }%
10451 }%
10452 {%
10453   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10454 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10455 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
10456 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10457 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10458 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10459 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10460   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10461 }%
10462 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10463   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10464 }%
10465 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10466   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10467 }%
10468 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10469   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10470 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10471 \renewcommand*{\glxtrinlinefullformat}[2]{%
10472   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10473   \glxtrfullsep{##1}%
10474   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10475 }%
10476 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10477   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10478   \glxtrfullsep{##1}%
10479   \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
10480 }%

```



```

10481 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10482   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10483   \glsxtrfullsep{##1}%
10484   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
10485 }%
10486 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10487   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10488   \glsxtrfullsep{##1}%
10489   \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
10490 }%

```

The first use full form only displays the long form.

```

10491 \renewcommand*{\glsxtrfullformat}[2]{%
10492   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10493 }%
10494 \renewcommand*{\glsxtrfullplformat}[2]{%
10495   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10496 }%
10497 \renewcommand*{\Glsxtrfullformat}[2]{%
10498   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10499 }%
10500 \renewcommand*{\Glsxtrfullplformat}[2]{%
10501   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10502 }%
10503 }

```

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.

```

10504 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10505 {%
10506   \renewcommand*{\CustomAbbreviationFields}{%
10507     name={\glsxtrlongnoshortname},
10508     sort={\the\glsshorttok},
10509     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10510     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10511     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10512     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10513     description={\the\glslongtok}%
10514   }%

```

Unset the regular attribute if it has been set.

```

10515 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10516   \glshasattribute{\the\glslabeltok}{regular}%
10517   {%
10518     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10519   }%
10520   {}%
10521 }%
10522 }%

```

```

10523 {%
10524   \GlsXtrUseAbbrStyleFmts{long-desc}%
10525 }

```

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.

```

10526 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10527 {%
10528   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10529   \glsfirstlonghyphenfont{#1}%
10530 }%
10531 }

```

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.

```

10532 \newcommand*{\glsxtrpostthyphenshort}[2]{%
10533 {%
10534   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10535   \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10536   \glsxtrfullsep{#1}%
10537   \glsxtrparen
10538   {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10539   \ifglsxtrininsertinside\else{#2}\fi
10540 }%
10541 }%
10542 }

```

hyphensubsequent `\glsxtrpostthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10543 \newcommand*{\glsxtrpostthyphensubsequent}[2]{%
10544   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%
10545   \ifglsxtrininsertinside \else{#2}\fi
10546 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10547 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10548 {%
10549   \renewcommand*{\CustomAbbreviationFields}{%
10550     name={\glxtrlongshortname},
10551     sort={\the\glsshorttok},
10552     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10553     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10554     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10555     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10556 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10557   \csdef{glxtrpostlink\glscategorylabel}{%
10558     \glxtrifwasfirstuse
10559     {%
10560       \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10561     }%
10562   }%

```

Put the insertion into the post-link:

```

10563     \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10564   }%
10565 }%
10566 \glshasattribute{\the\glslabeltok}{regular}%
10567 {%
10568   \glissetattribute{\the\glslabeltok}{regular}{false}%
10569 }%
10570 {}%
10571 }%
10572 }%
10573 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10574 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10575 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10576 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10577 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10578 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10579 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10580   \glsabbrvfont{\glsaccessshort{##1}}%
10581 }%
10582 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10583   \glsabbrvfont{\glsaccessshortpl{##1}}%
10584 }%
10585 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10586   \glsabbrvfont{\Glsaccessshort{##1}}%
10587 }%
10588 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

```

```

10589 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10590 }%

First use full form:

10591 \renewcommand*{\glsxtrfullformat}[2]{%
10592 \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10593 }%
10594 \renewcommand*{\glsxtrfullplformat}[2]{%
10595 \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10596 }%
10597 \renewcommand*{\Glsxtrfullformat}[2]{%
10598 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10599 }%
10600 \renewcommand*{\Glsxtrfullplformat}[2]{%
10601 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10602 }%

```

In-line format.

```

10603 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10604 \glsfirstlonghyphenfont{\glsaccesslong{##1}}%
10605 \ifglsxtrininsertinside{##2}\fi}%
10606 \ifglsxtrininsertinside \else{##2}\fi
10607 }%
10608 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10609 \glsfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10610 \ifglsxtrininsertinside{##2}\fi}%
10611 \ifglsxtrininsertinside \else{##2}\fi
10612 }%
10613 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10614 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10615 \ifglsxtrininsertinside{##2}\fi}%
10616 \ifglsxtrininsertinside \else{##2}\fi
10617 }%
10618 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10619 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10620 \ifglsxtrininsertinside{##2}\fi}%
10621 \ifglsxtrininsertinside \else{##2}\fi
10622 }%
10623 }

```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10624 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10625 {%
10626 \renewcommand*{\CustomAbbreviationFields}{%
10627 name={\glsxtrlongshortdescname},
10628 sort={\glsxtrlongshortdescsort},%
10629 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10630 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10631 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10632 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%

```

```

10633 }%
10634 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10635   \csdef{glxtrpostlink\glscategorylabel}{%
10636     \glxtrifwasfirstuse
10637     {%
10638       \glxtrposthyphenshort{\glslabel}{\glinsert}%
10639     }%
10640   }%

```

Put the insertion into the post-link:

```

10641     \glxtrposthyphensubsequent{\glslabel}{\glinsert}%
10642   }%
10643 }%
10644 \glshasattribute{\the\glslabeltok}{regular}%
10645 {%
10646   \glissetattribute{\the\glslabeltok}{regular}{false}%
10647 }%
10648 {}%
10649 }%
10650 }%
10651 {%
10652   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10653 }

```

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.

```
10654 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10655 {%
```

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.)

```

10656   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{%
10657   \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10658   \ifglxtrininsertinside\else{#4}\fi
10659   \glxtrfullsep{#1}%
10660   \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10661   \ifglxtrininsertinside\else{#4}\fi}%
10662 }%
10663 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```
10664 \newabbreviationstyle{short-hyphen-long-hyphen}%

```

```

10665 {%
10666   \renewcommand*{\CustomAbbreviationFields}{%
10667     name={\glxtrshortlongname},
10668     sort={\the\glsshorttok},
10669     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10670       \protect\glxtrfullsep{\the\glslabeltok}%
10671       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10672     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10673       \protect\glxtrfullsep{\the\glslabeltok}%
10674       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10675     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
10676     description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10677   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10678     \glshasattribute{\the\glslabeltok}{regular}%
10679     {%
10680       \glissetattribute{\the\glslabeltok}{regular}{false}%
10681     }%
10682   }%
10683 }%
10684 }%
10685 {%
10686   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10687   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10688   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10689   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10690   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10691   \renewcommand*{\glxtrfullformat}[2]{%
10692     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10693   }%
10694   \renewcommand*{\glxtrfullplformat}[2]{%
10695     \glxtrshorthyphenlong{##1}%
10696     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10697   }%
10698   \renewcommand*{\Glsxtrfullformat}[2]{%
10699     \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10700   }%
10701   \renewcommand*{\Glsxtrfullplformat}[2]{%
10702     \glxtrshorthyphenlong{##1}%
10703     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10704   }%
10705 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10706 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10707 {%
10708   \renewcommand*{\CustomAbbreviationFields}{%

```

```

10709   name={\glxtrshortlongdescname},
10710   sort={\glxtrshortlongdescsort},
10711   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10712     \protect\glxtrfullsep{\the\glslabeltok}%
10713     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10714   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10715     \protect\glxtrfullsep{\the\glslabeltok}%
10716     \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10717   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10718   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10719 }%

```

Unset the regular attribute if it has been set.

```

10720 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10721   \glshasattribute{\the\glslabeltok}{regular}%
10722   {%
10723     \glissetattribute{\the\glslabeltok}{regular}{false}%
10724   }%
10725   {}%
10726 }%
10727 }%
10728 {%
10729   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10730 }

```

`\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.

```
10731 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

10732   {%
10733     \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}}%
10734     \glsfirstabbrvhyphenfont{#1}%
10735   }%
10736 }

```

`\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.

```

10737 \newcommand*{\glxtrposthyphenlong}[2]{%
10738   {%

```

```

10739 \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
10740 \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10741 \glxtrfullsep{#1}%
10742 \glxtrparen
10743 {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
10744 \ifglxtrininsertinside\else{#2}\fi
10745 }%
10746 }%
10747 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10748 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10749 {%
10750 \renewcommand*{\CustomAbbreviationFields}{%
10751 name={\glxtrshortlongname},
10752 sort={\the\glsshorttok},
10753 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10754 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10755 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
10756 description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10757 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10758 \csdef{glxtrpostlink\glscategorylabel}{%
10759 \glxtrifwasfirstuse
10760 {%
10761 \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10762 }%
10763 }%

```

Put the insertion into the post-link:

```

10764 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10765 }%
10766 }%
10767 \glshasattribute{\the\glslabeltok}{regular}%
10768 {%
10769 \glissetattribute{\the\glslabeltok}{regular}{false}%
10770 }%
10771 {}%
10772 }%
10773 }%
10774 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10775 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10776 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
10777 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10778 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10779 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:



```

10780 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10781   \glsabbrvfont{\glsaccessshort{##1}}}%
10782 }%
10783 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10784   \glsabbrvfont{\glsaccessshortpl{##1}}}%
10785 }%
10786 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10787   \glsabbrvfont{\Glsaccessshort{##1}}}%
10788 }%
10789 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10790   \glsabbrvfont{\Glsaccessshortpl{##1}}}%
10791 }%

```

First use full form:

```

10792 \renewcommand*{\glxtrfullformat}[2]{%
10793   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10794 }%
10795 \renewcommand*{\glxtrfullplformat}[2]{%
10796   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10797 }%
10798 \renewcommand*{\Glsxtrfullformat}[2]{%
10799   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10800 }%
10801 \renewcommand*{\Glsxtrfullplformat}[2]{%
10802   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10803 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10804 \renewcommand*{\glxtrinlinefullformat}[2]{%
10805   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10806   \ifglxtrininsertinside{##2}\fi}%
10807   \ifglxtrininsertinside \else{##2}\fi
10808 }%
10809 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10810   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10811   \ifglxtrininsertinside{##2}\fi}%
10812   \ifglxtrininsertinside \else{##2}\fi
10813 }%
10814 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10815   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10816   \ifglxtrininsertinside{##2}\fi}%
10817   \ifglxtrininsertinside \else{##2}\fi
10818 }%
10819 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10820   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10821   \ifglxtrininsertinside{##2}\fi}%
10822   \ifglxtrininsertinside \else{##2}\fi
10823 }%
10824 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
10825 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10826 {%
10827   \renewcommand*{\CustomAbbreviationFields}{%
10828     name={\glstrshortlongdescname},
10829     sort={\glstrshortlongdescsort},%
10830     first={\protect\glfirstabbrvhyphenfont{\the\glsshorttok}},%
10831     firstplural={\protect\glfirstabbrvhyphenfont{\the\glsshortpltok}},%
10832     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10833     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10834 }%
10835 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10836   \csdef{glstrpostlink\glscategorylabel}{%
10837     \glstrifwasfirstuse
10838     {%
10839       \glstrposthyphenlong{\glslabel}{\glsinsert}%
10840     }%
10841     {%
```

Put the insertion into the post-link:

```
10842       \glstrposthyphensubsequent{\glslabel}{\glsinsert}%
10843     }%
10844   }%
10845   \glshasattribute{\the\glslabeltok}{regular}%
10846   {%
10847     \glissetattribute{\the\glslabeltok}{regular}{false}%
10848   }%
10849   {}}%
10850 }%
10851 }%
10852 {%
10853   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10854 }
```

### 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

```
10855 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
10856 \newcommand*{\glfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
10857 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
10858 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10859 \newcommand*{\glxtronlysuffix}{\glxtrabbrvpluralsuffix}
```

\glxtronlyname The default name format for this style.

```
10860 \newcommand*{\glxtronlyname}{%
10861   \protect\glxabbrvonlyfont{\the\glsshorttok}%
10862 }
```

only-short-only

```
10863 \newabbreviationstyle{long-only-short-only}%
10864 {%
10865   \renewcommand*{\CustomAbbreviationFields}{%
10866     name={\glxtronlyname},
10867     sort={\the\glsshorttok},
10868     first={\protect\glxfirstlongonlyfont{\the\glslongtok}},%
10869     firstplural={\protect\glxfirstlongonlyfont{\the\glslongpltok}},%
10870     plural={\protect\glxabbrvonlyfont{\the\glsshortpltok}},%
10871     description={\protect\glxlongonlyfont{\the\glslongtok}}}%
10872 }
```

Unset the regular attribute if it has been set.

```
10872 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10873   \glshasattribute{\the\glslabeltok}{regular}%
10874   {%
10875     \glissetattribute{\the\glslabeltok}{regular}{false}%
10876   }%
10877   {}%
10878 }%
10879 }%
10880 {%
10881   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtronlysuffix}%
10882   \renewcommand*{\glxabbrvfont}[1]{\glxabbrvonlyfont{##1}}%
10883   \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvonlyfont{##1}}%
10884   \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlongonlyfont{##1}}%
10885   \renewcommand*{\glxlongfont}[1]{\glxlongonlyfont{##1}}%
10886 }
```

The first use full form doesn't show the short form.

```
10886 \renewcommand*{\glxtrfullformat}[2]{%
10887   \glxfirstlongonlyfont{\glxaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10888   \ifglxtrinsertinside\else##2\fi
10889 }%
10890 \renewcommand*{\glxtrfullplformat}[2]{%
10891   \glxfirstlongonlyfont{\glxaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10892   \ifglxtrinsertinside\else##2\fi
10893 }%
10894 \renewcommand*{\GlsXtrfullformat}[2]{%
10895   \glxfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10896   \ifglxtrinsertinside\else##2\fi
10897 }%
```

```

10898 \renewcommand*{\Glsxtrfullplformat}[2]{%
10899   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10900   \ifglxtrinsertinside\else##2\fi
10901 }%

```

The inline full form does show the short form.

```

10902 \renewcommand*{\glxtrinlinefullformat}[2]{%
10903   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10904   \ifglxtrinsertinside\else##2\fi
10905   \glxtrfullsep{##1}%
10906   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshort{##1}}}%
10907 }%
10908 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10909   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10910   \ifglxtrinsertinside\else##2\fi
10911   \glxtrfullsep{##1}%
10912   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10913 }%
10914 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10915   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10916   \ifglxtrinsertinside\else##2\fi
10917   \glxtrfullsep{##1}%
10918   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10919 }%
10920 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10921   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10922   \ifglxtrinsertinside\else##2\fi
10923   \glxtrfullsep{##1}%
10924   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10925 }%
10926 }

```

xtronlydescsort

```

10927 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

10928 \newcommand*{\glxtronlydescname}{%
10929   \protect\glslongfont{\the\glslongtok}}%
10930 }

```

short-only-desc

```

10931 \newabbreviationstyle{long-only-short-only-desc}%
10932 {%
10933   \renewcommand*{\CustomAbbreviationFields}{%
10934     name={\glxtronlydescname},
10935     sort={\glxtronlydescsort},%
10936     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10937     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10938     text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%

```

```

10939   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
10940 }%

  Unset the regular attribute if it has been set.
10941   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10942     \glshasattribute{\the\glslabeltok}{regular}%
10943     {%
10944       \glssetattribute{\the\glslabeltok}{regular}{false}%
10945     }%
10946   }%
10947 }%
10948 }%
10949 {%
10950   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10951 }

```

## 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:

```

10952 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```

10953 \renewcommand*{\markright}[1]{%
10954   \glsxtrmarkhook

```

```

10955 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
10956 \glsxtrrestoremarkhook
10957 }

```

\markboth Save original definition:

```
10958 \let\@glsxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```

10959 \renewcommand*{\markboth}[2]{%
10960 \glsxtrmarkhook
10961 \@glsxtr@org@markboth
10962 {\@glsxtrinmark#1\@glsxtrnotinmark}%
10963 {\@glsxtrinmark#2\@glsxtrnotinmark}%
10964 \glsxtrrestoremarkhook
10965 }

```

Also do this for \@starttoc

\@starttoc Save original definition:

```
10966 \let\@glsxtr@org@@starttoc\@starttoc
```

Redefine:

```

10967 \renewcommand*{\@starttoc}[1]{%
10968 \glsxtrmarkhook
10969 \@glsxtrinmark
10970 \@glsxtr@org@@starttoc{#1}%
10971 \@glsxtrnotinmark
10972 \glsxtrrestoremarkhook
10973 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

10974 \newcommand*{\glsxtrRevertMarks}{%
10975 \let\markright\@glsxtr@org@markright
10976 \let\markboth\@glsxtr@org@markboth
10977 \let\@starttoc\@glsxtr@org@@starttoc
10978 }

```

rRevertTocMarks Just restores \@starttoc.

```

10979 \newcommand*{\glsxtrRevertTocMarks}{%
10980 \let\@starttoc\@glsxtr@org@@starttoc
10981 }

```

\glsxtrifinmark

```
10982 \newcommand*{\glsxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```

10983 \newrobustcmd*{\@glsxtrinmark}{%
10984 \let\glsxtrifinmark\@firstoftwo
10985 }

```

glsxtrnotinmark

```
10986 \newrobustcmd*{\@glsxtrnotinmark}{%
10987   \let\glsxtrifinmark\@secondoftwo
10988 }
```

eorpdforheading

```
10989 \ifdef\teorpdfstring
10990 {
10991   \newcommand*{\glsxtrtitleorpdforheading}[3]{\teorpdfstring{#1}{#2}}
10992 }
10993 {
10994   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
10995 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
10996 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
10997   \let\@glsxtr@org@MakeUppercase\MakeUppercase
10998   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
10999   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11000   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11001   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11002   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11003   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11004   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11005   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11006   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11007   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11008   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11009   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11010   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11011   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11012   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11013   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11014   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11015   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11016   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11017   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11018   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11019   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11020   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
11021   \let\glsxtrifinmark\@firstoftwo
11022   \let\MakeUppercase\MakeTextUppercase
11023   \let\glsxtrtitleorpdforheading\@thirdofthree
11024   \let\glsxtrtitleshort\glsxtrheadshort
11025   \let\glsxtrtitleshortpl\glsxtrheadshortpl
```

```

11026 \let\Glsxtrtitleshort\Glsxtrheadshort
11027 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11028 \let\glxstrtitlename\glxstrheadname
11029 \let\Glsxtrtitlename\Glsxtrheadname
11030 \let\glxstrtitletext\glxstrheadtext
11031 \let\Glsxtrtitletext\Glsxtrheadtext
11032 \let\glxstrtitleplural\glxstrheadplural
11033 \let\Glsxtrtitleplural\Glsxtrheadplural
11034 \let\glxstrtitlefirst\glxstrheadfirst
11035 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11036 \let\glxstrtitlefirstplural\glxstrheadfirstplural
11037 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11038 \let\glxstrtitlelong\glxstrheadlong
11039 \let\glxstrtitlelongpl\glxstrheadlongpl
11040 \let\Glsxtrtitlelong\Glsxtrheadlong
11041 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11042 \let\glxstrtitlefull\glxstrheadfull
11043 \let\glxstrtitlefullpl\glxstrheadfullpl
11044 \let\Glsxtrtitlefull\Glsxtrheadfull
11045 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11046 }

```

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.)

```

11047 \newcommand*{\glxstrrestoremarkhook}{%
11048 \let\glxstrifinmark\@secondoftwo
11049 \let\MakeUppercase\@glxstr@org@MakeUppercase
11050 \let\glxstrtitleorpdforheading\@glxstr@org@glxstrtitleorpdforheading
11051 \let\glxstrtitleshort\@glxstr@org@glxstrtitleshort
11052 \let\glxstrtitleshortpl\@glxstr@org@glxstrtitleshortpl
11053 \let\Glsxtrtitleshort\@glxstr@org@Glsxtrtitleshort
11054 \let\Glsxtrtitleshortpl\@glxstr@org@Glsxtrtitleshortpl
11055 \let\glxstrtitlename\@glxstr@org@glxstrtitlename
11056 \let\Glsxtrtitlename\@glxstr@org@Glsxtrtitlename
11057 \let\glxstrtitletext\@glxstr@org@glxstrtitletext
11058 \let\Glsxtrtitletext\@glxstr@org@Glsxtrtitletext
11059 \let\glxstrtitleplural\@glxstr@org@glxstrtitleplural
11060 \let\Glsxtrtitleplural\@glxstr@org@Glsxtrtitleplural
11061 \let\glxstrtitlefirst\@glxstr@org@glxstrtitlefirst
11062 \let\Glsxtrtitlefirst\@glxstr@org@Glsxtrtitlefirst
11063 \let\glxstrtitlefirstplural\@glxstr@org@glxstrtitlefirstplural
11064 \let\Glsxtrtitlefirstplural\@glxstr@org@Glsxtrtitlefirstplural
11065 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
11066 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
11067 \let\Glsxtrtitlelong\@glxstr@org@Glsxtrtitlelong
11068 \let\Glsxtrtitlelongpl\@glxstr@org@Glsxtrtitlelongpl
11069 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull

```



```

11070 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
11071 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
11072 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
11073 }

```

Instead of using one document-wide conditional, use `headuc` attribute to determine whether or not to use the all upper case form.

**glsxtrheadshort** Command used to display short form in the page header.

```

11074 \newcommand*{\glsxtrheadshort}[1]{%
11075 \protect\NoCaseChange
11076 {%
11077 \glsifattribute{#1}{headuc}{true}%
11078 {%
11079 \GLSxtrshort[noindex,hyper=false]{#1}[]%
11080 }%
11081 {%
11082 \glsxtrshort[noindex,hyper=false]{#1}[]%
11083 }%
11084 }%
11085 }

```

**lsxtrtitleshort** Command to display short form of abbreviation in section title and table of contents.

```

11086 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11087 \glsxtrshort[noindex,hyper=false]{#1}[]%
11088 }

```

**sxtrheadshortpl** Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrshortpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

11089 \newcommand*{\glsxtrheadshortpl}[1]{%
11090 \protect\NoCaseChange
11091 {%
11092 \glsifattribute{#1}{headuc}{true}%
11093 {%
11094 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11095 }%
11096 {%
11097 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11098 }%
11099 }%
11100 }

```

**lsxtrtitleshortpl** Command to display plural short form of abbreviation in section title and table of contents.

```

11101 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
11102 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
11103 }

```

**Glsxtrheadshort** Command used to display short form in the page header with the first letter converted to upper case.

```

11104 \newcommand*{\Glsxtrheadshort}[1]{%
11105   \protect\NoCaseChange
11106   {%
11107     \glsifattribute{#1}{headuc}{true}%
11108     {%
11109       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11110     }%
11111     {%
11112       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11113     }%
11114   }%
11115 }

```

**ltxtrtitleshort** Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11116 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11117   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11118 }

```

**sxtrheadshortpl** Command used to display plural short form in the page header with the first letter converted to upper case.

```

11119 \newcommand*{\Glsxtrheadshortpl}[1]{%
11120   \protect\NoCaseChange
11121   {%
11122     \glsifattribute{#1}{headuc}{true}%
11123     {%
11124       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11125     }%
11126     {%
11127       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11128     }%
11129   }%
11130 }

```

**xtrtitleshortpl** Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11131 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11132   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11133 }

```

**\glsxtrheadname** As above but for the name value.

```

11134 \newcommand*{\glsxtrheadname}[1]{%
11135   \protect\NoCaseChange
11136   {%
11137     \glsifattribute{#1}{headuc}{true}%
11138     {%

```

```

11139      \GLSname[noindex,hyper=false]{#1}[]%
11140    }%
11141    {%
11142      \glsname[noindex,hyper=false]{#1}[]%
11143    }%
11144  }%
11145 }

```

`\glsxtrtitlename` Command to display name value in section title and table of contents.

```

11146 \newrobustcmd*{\glsxtrtitlename}[1]{%
11147   \glsname[noindex,hyper=false]{#1}[]%
11148 }

```

`\Glsxtrheadname` First letter converted to upper case

```

11149 \newcommand*{\Glsxtrheadname}[1]{%
11150   \protect\NoCaseChange
11151   {%
11152     \glsifattribute{#1}{headuc}{true}%
11153     {%
11154       \GLSname[noindex,hyper=false]{#1}[]%
11155     }%
11156     {%
11157       \Glsname[noindex,hyper=false]{#1}[]%
11158     }%
11159   }%
11160 }

```

`\Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11161 %\changes{1.21}{2017-11-03}{new}
11162 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11163   \Glsname[noindex,hyper=false]{#1}[]%
11164 }

```

`\glsxtrheadtext` As above but for the text value.

```

11165 \newcommand*{\glsxtrheadtext}[1]{%
11166   \protect\NoCaseChange
11167   {%
11168     \glsifattribute{#1}{headuc}{true}%
11169     {%
11170       \GLStext[noindex,hyper=false]{#1}[]%
11171     }%
11172     {%
11173       \glstext[noindex,hyper=false]{#1}[]%
11174     }%
11175   }%
11176 }

```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
11177 \newrobustcmd*{\glsxtrtitletext}[1]{%
11178   \glstext[noindex,hyper=false]{#1}[]%
11179 }
```

`\Glsxtrheadtext` First letter converted to upper case

```
11180 \newcommand*{\Glsxtrheadtext}[1]{%
11181   \protect\NoCaseChange
11182   {%
11183     \glsifattribute{#1}{headuc}{true}%
11184     {%
11185       \GLStext[noindex,hyper=false]{#1}[]%
11186     }%
11187     {%
11188       \Glstext[noindex,hyper=false]{#1}[]%
11189     }%
11190   }%
11191 }
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
11192 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11193   \Glstext[noindex,hyper=false]{#1}[]%
11194 }
```

`lsxtrheadplural` As above but for the plural value.

```
11195 \newcommand*{\glsxtrheadplural}[1]{%
11196   \protect\NoCaseChange
11197   {%
11198     \glsifattribute{#1}{headuc}{true}%
11199     {%
11200       \GLSplural[noindex,hyper=false]{#1}[]%
11201     }%
11202     {%
11203       \glsplural[noindex,hyper=false]{#1}[]%
11204     }%
11205   }%
11206 }
```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```
11207 \newrobustcmd*{\glsxtrtitleplural}[1]{%
11208   \glsplural[noindex,hyper=false]{#1}[]%
11209 }
```

`lsxtrheadplural` Convert first letter to upper case.

```
11210 \newcommand*{\Glsxtrheadplural}[1]{%
11211   \protect\NoCaseChange
11212   {%
```

```

11213 \glsifattribute{#1}{headuc}{true}%
11214 {%
11215     \GLSplural[noindex,hyper=false]{#1}[]%
11216 }%
11217 {%
11218     \Glsplural[noindex,hyper=false]{#1}[]%
11219 }%
11220 }%
11221 }

```

**sxtrtitleplural** Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

11222 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11223     \GLsplural[noindex,hyper=false]{#1}[]%
11224 }

```

**glsxtrheadfirst** As above but for the first value.

```

11225 \newcommand*{\glsxtrheadfirst}[1]{%
11226     \protect\NoCaseChange
11227 {%
11228     \glsifattribute{#1}{headuc}{true}%
11229 {%
11230     \GLSfirst[noindex,hyper=false]{#1}[]%
11231 }%
11232 {%
11233     \glsfirst[noindex,hyper=false]{#1}[]%
11234 }%
11235 }%
11236 }

```

**lsxtrtitlefirst** Command to display first value in section title and table of contents.

```

11237 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
11238     \glsfirst[noindex,hyper=false]{#1}[]%
11239 }

```

**Glsxtrheadfirst** First letter converted to upper case

```

11240 \newcommand*{\Glsxtrheadfirst}[1]{%
11241     \protect\NoCaseChange
11242 {%
11243     \glsifattribute{#1}{headuc}{true}%
11244 {%
11245     \GLSfirst[noindex,hyper=false]{#1}[]%
11246 }%
11247 {%
11248     \Glsfirst[noindex,hyper=false]{#1}[]%
11249 }%
11250 }%
11251 }

```

`\lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11252 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11253   \Glsfirst[noindex,hyper=false]{#1}[]%
11254 }
```

`\headfirstplural` As above but for the firstplural value.

```
11255 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11256   \protect\NoCaseChange
11257   {%
11258     \glsifattribute{#1}{headuc}{true}%
11259     {%
11260       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11261     }%
11262     {%
11263       \glsfirstplural[noindex,hyper=false]{#1}[]%
11264     }%
11265   }%
11266 }
```

`\itlefirstplural` Command to display firstplural value in section title and table of contents.

```
11267 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11268   \glsfirstplural[noindex,hyper=false]{#1}[]%
11269 }
```

`\headfirstplural` First letter converted to upper case

```
11270 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11271   \protect\NoCaseChange
11272   {%
11273     \glsifattribute{#1}{headuc}{true}%
11274     {%
11275       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11276     }%
11277     {%
11278       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11279     }%
11280   }%
11281 }
```

`\itlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
11282 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11283   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11284 }
```

`\glsxtrheadlong` Command used to display long form in the page header.

```
11285 \newcommand*{\glsxtrheadlong}[1]{%
11286   \protect\NoCaseChange
```

```

11287 {%
11288   \glsifattribute{#1}{headuc}{true}%
11289   {%
11290     \GLSxtrlong[noindex,hyper=false]{#1}[]%
11291   }%
11292   {%
11293     \glsxtrlong[noindex,hyper=false]{#1}[]%
11294   }%
11295 }%
11296 }

```

`\glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

11297 \newrobustcmd*{\glsxtrtitlelong}[1]{%
11298   \glsxtrlong[noindex,hyper=false]{#1}[]%
11299 }

```

`\glsxtrheadlongpl` 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.

```

11300 \newcommand*{\glsxtrheadlongpl}[1]{%
11301   \protect\NoCaseChange
11302   {%
11303     \glsifattribute{#1}{headuc}{true}%
11304     {%
11305       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11306     }%
11307     {%
11308       \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11309     }%
11310   }%
11311 }

```

`\glsxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

11312 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%
11313   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
11314 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

11315 \newcommand*{\Glsxtrheadlong}[1]{%
11316   \protect\NoCaseChange
11317   {%
11318     \glsifattribute{#1}{headuc}{true}%
11319     {%
11320       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11321     }%
11322     {%
11323       \Glsxtrlong[noindex,hyper=false]{#1}[]%

```

```

11324 }%
11325 }%
11326 }

```

**Glsxtrtitlelong** Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11327 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11328   \Glsxtrlong[noindex,hyper=false]{#1}[]%
11329 }

```

**lsxtrheadlongpl** Command used to display plural long form in the page header with the first letter converted to upper case.

```

11330 \newcommand*{\Glsxtrheadlongpl}[1]{%
11331   \protect\NoCaseChange
11332   {%
11333     \glsifattribute{#1}{headuc}{true}%
11334     {%
11335       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11336     }%
11337     {%
11338       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11339     }%
11340   }%
11341 }

```

**sxtrtitlelongpl** Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11342 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11343   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11344 }

```

**\glsxtrheadfull** Command used to display full form in the page header.

```

11345 \newcommand*{\glsxtrheadfull}[1]{%
11346   \protect\NoCaseChange
11347   {%
11348     \glsifattribute{#1}{headuc}{true}%
11349     {%
11350       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11351     }%
11352     {%
11353       \glsxtrfull[noindex,hyper=false]{#1}[]%
11354     }%
11355   }%
11356 }

```

**glsxtrtitlefull** Command to display full form of abbreviation in section title and table of contents.

```

11357 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11358   \glsxtrfull[noindex,hyper=false]{#1}[]%
11359 }

```



`\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.

```

11360 \newcommand*{\glxtrheadfullpl}[1]{%
11361   \protect\NoCaseChange
11362   {%
11363     \gl@ifattribute{#1}{headuc}{true}%
11364     {%
11365       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11366     }%
11367     {%
11368       \glxtrfullpl[noindex,hyper=false]{#1}[]%
11369     }%
11370   }%
11371 }
```

`\sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```

11372 \newrobustcmd*{\glxtrtitlefullpl}[1]{%
11373   \glxtrfullpl[noindex,hyper=false]{#1}[]%
11374 }
```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```

11375 \newcommand*{\Glsxtrheadfull}[1]{%
11376   \protect\NoCaseChange
11377   {%
11378     \gl@ifattribute{#1}{headuc}{true}%
11379     {%
11380       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11381     }%
11382     {%
11383       \Glsxtrfull[noindex,hyper=false]{#1}[]%
11384     }%
11385   }%
11386 }
```

`\Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11387 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11388   \Glsxtrfull[noindex,hyper=false]{#1}[]%
11389 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11390 \newcommand*{\Glsxtrheadfullpl}[1]{%
11391   \protect\NoCaseChange
11392   {%
11393     \gl@ifattribute{#1}{headuc}{true}%
11394     {%
11395       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11396     }%
11397     {%
11398       \glxtrfullpl[noindex,hyper=false]{#1}[]%
11399     }%
11400   }%
11401 }
```

```

11394   {%
11395     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11396   }%
11397   {%
11398     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11399   }%
11400 }%
11401 }

```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11402 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11403   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
11404 }

```

`\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.

```

11405 \ifdef\texorpdfstring
11406 {
11407   \newcommand*{\glsfmtshort}[1]{%
11408     \texorpdfstring
11409       {\glsxtrtitleshort{#1}}%
11410       {\glsentryshort{#1}}%
11411   }
11412 }
11413 {
11414   \newcommand*{\glsfmtshort}[1]{%
11415     \glsxtrtitleshort{#1}}
11416 }

```

Similarly for the plural version.

```

\glsfmtshortpl
11417 \ifdef\texorpdfstring
11418 {
11419   \newcommand*{\glsfmtshortpl}[1]{%
11420     \texorpdfstring
11421       {\glsxtrtitleshortpl{#1}}%
11422       {\glsentryshortpl{#1}}%
11423   }
11424 }
11425 {
11426   \newcommand*{\glsfmtshortpl}[1]{%
11427     \glsxtrtitleshortpl{#1}}
11428 }

```

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).

```
11429 \ifdef\teorpdfstring
11430 {
11431   \newcommand*\Glsfmtshort}[1]{%
11432     \teorpdfstring
11433       {\Glsxtrtitleshort{#1}}%
11434       {\glsentryshort{#1}}%
11435   }
11436 }
11437 {
11438   \newcommand*\Glsfmtshort}[1]{%
11439     \Glsxtrtitleshort{#1}}
11440 }
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
11441 \ifdef\teorpdfstring
11442 {
11443   \newcommand*\Glsfmtshortpl}[1]{%
11444     \teorpdfstring
11445       {\Glsxtrtitleshortpl{#1}}%
11446       {\glsentryshortpl{#1}}%
11447   }
11448 }
11449 {
11450   \newcommand*\Glsfmtshortpl}[1]{%
11451     \Glsxtrtitleshortpl{#1}}
11452 }
```

`\glsfmtname` As above but for the name value.

```
11453 \ifdef\teorpdfstring
11454 {
11455   \newcommand*\glsfmtname}[1]{%
11456     \teorpdfstring
11457       {\glsxtrtitlename{#1}}%
11458       {\glsentryname{#1}}%
11459   }
11460 }
11461 {
11462   \newcommand*\glsfmtname}[1]{%
11463     \glsxtrtitlename{#1}}
11464 }
```

`\Glsfmtname` First letter converted to upper case.

```
11465 \ifdef\teorpdfstring
11466 {
11467   \newcommand*\Glsfmtname}[1]{%
11468     \teorpdfstring
11469       {\Glsxtrtitlename{#1}}%
11470       {\glsentryname{#1}}%
11471   }
```

```

11471 }
11472 }
11473 {
11474   \newcommand*{\Glsfmtname}[1]{%
11475     \Glsxtrtitlename{#1}}
11476 }

```

`\glsfmttext` As above but for the text value.

```

11477 \ifdef\texorpdfstring
11478 {
11479   \newcommand*{\glsfmttext}[1]{%
11480     \texorpdfstring
11481       {\Glsxtrtitletext{#1}}%
11482       {\Glsentrytext{#1}}%
11483   }
11484 }
11485 {
11486   \newcommand*{\glsfmttext}[1]{%
11487     \Glsxtrtitletext{#1}}
11488 }

```

`\Glsfmttext` First letter converted to upper case.

```

11489 \ifdef\texorpdfstring
11490 {
11491   \newcommand*{\Glsfmttext}[1]{%
11492     \texorpdfstring
11493       {\Glsxtrtitletext{#1}}%
11494       {\Glsentrytext{#1}}%
11495   }
11496 }
11497 {
11498   \newcommand*{\Glsfmttext}[1]{%
11499     \Glsxtrtitletext{#1}}
11500 }

```

`\glsfmtplural` As above but for the plural value.

```

11501 \ifdef\texorpdfstring
11502 {
11503   \newcommand*{\glsfmtplural}[1]{%
11504     \texorpdfstring
11505       {\Glsxtrtitleplural{#1}}%
11506       {\Glsentryplural{#1}}%
11507   }
11508 }
11509 {
11510   \newcommand*{\glsfmtplural}[1]{%
11511     \Glsxtrtitleplural{#1}}
11512 }

```

`\Glsfmtplural` First letter converted to upper case.

```
11513 \ifdef\teorpdfstring
11514 {
11515   \newcommand*\Glsfmtplural}[1]{%
11516     \teorpdfstring
11517     {\Glsxtrtitleplural{#1}}%
11518     {\glsentryplural{#1}}%
11519   }
11520 }
11521 {
11522   \newcommand*\Glsfmtplural}[1]{%
11523     \Glsxtrtitleplural{#1}}
11524 }
```

`\glsfmtfirst` As above but for the first value.

```
11525 \ifdef\teorpdfstring
11526 {
11527   \newcommand*\glsfmtfirst}[1]{%
11528     \teorpdfstring
11529     {\glsxtrtitlefirst{#1}}%
11530     {\glsentryfirst{#1}}%
11531   }
11532 }
11533 {
11534   \newcommand*\glsfmtfirst}[1]{%
11535     \glsxtrtitlefirst{#1}}
11536 }
```

`\Glsfmtfirst` First letter converted to upper case.

```
11537 \ifdef\teorpdfstring
11538 {
11539   \newcommand*\Glsfmtfirst}[1]{%
11540     \teorpdfstring
11541     {\Glsxtrtitlefirst{#1}}%
11542     {\glsentryfirst{#1}}%
11543   }
11544 }
11545 {
11546   \newcommand*\Glsfmtfirst}[1]{%
11547     \Glsxtrtitlefirst{#1}}
11548 }
```

`\glsfmtfirstpl` As above but for the firstplural value.

```
11549 \ifdef\teorpdfstring
11550 {
11551   \newcommand*\glsfmtfirstpl}[1]{%
11552     \teorpdfstring
11553     {\glsxtrtitlefirstplural{#1}}%
11554     {\glsentryfirstplural{#1}}%
11555   }
```

```

11555 }
11556 }
11557 {
11558   \newcommand*{\glsfmtfirstpl}[1]{%
11559     \glsxtrtitlefirstplural{#1}}
11560 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11561 \ifdef\texorpdfstring
11562 {
11563   \newcommand*{\Glsfmtfirstpl}[1]{%
11564     \texorpdfstring
11565       {\Glsxtrtitlefirstplural{#1}}%
11566       {\glsentryfirstplural{#1}}%
11567   }
11568 }
11569 {
11570   \newcommand*{\Glsfmtfirstpl}[1]{%
11571     \Glsxtrtitlefirstplural{#1}}
11572 }

```

`\glsfmtlong` As above but for the long value.

```

11573 \ifdef\texorpdfstring
11574 {
11575   \newcommand*{\glsfmtlong}[1]{%
11576     \texorpdfstring
11577       {\glsxtrtitlelong{#1}}%
11578       {\glsentrylong{#1}}%
11579   }
11580 }
11581 {
11582   \newcommand*{\glsfmtlong}[1]{%
11583     \glsxtrtitlelong{#1}}
11584 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11585 \ifdef\texorpdfstring
11586 {
11587   \newcommand*{\Glsfmtlong}[1]{%
11588     \texorpdfstring
11589       {\Glsxtrtitlelong{#1}}%
11590       {\glsentrylong{#1}}%
11591   }
11592 }
11593 {
11594   \newcommand*{\Glsfmtlong}[1]{%
11595     \Glsxtrtitlelong{#1}}
11596 }

```

`\glsfmtlongpl` As above but for the longplural value.

```
11597 \ifdef\txorpdfstring
11598 {
11599   \newcommand*\glsfmtlongpl}[1]{%
11600     \txorpdfstring
11601     {\glsxtrtitlelongpl{#1}}%
11602     {\glsentrylongpl{#1}}%
11603   }
11604 }
11605 {
11606   \newcommand*\glsfmtlongpl}[1]{%
11607     \glsxtrtitlelongpl{#1}}
11608 }
```

`\Glsfmtlongpl` First letter converted to upper case.

```
11609 \ifdef\txorpdfstring
11610 {
11611   \newcommand*\Glsfmtlongpl}[1]{%
11612     \txorpdfstring
11613     {\Glsxtrtitlelongpl{#1}}%
11614     {\glsentrylongpl{#1}}%
11615   }
11616 }
11617 {
11618   \newcommand*\Glsfmtlongpl}[1]{%
11619     \Glsxtrtitlelongpl{#1}}
11620 }
```

`\glsfmtfull` In-line full format.

```
11621 \ifdef\txorpdfstring
11622 {
11623   \newcommand*\glsfmtfull}[1]{%
11624     \txorpdfstring
11625     {\glsxtrtitlefull{#1}}%
11626     {\glsxtrinlinefullformat{#1}{}}%
11627   }
11628 }
11629 {
11630   \newcommand*\glsfmtfull}[1]{%
11631     \glsxtrtitlefull{#1}}
11632 }
```

`\Glsfmtfull` First letter converted to upper case.

```
11633 \ifdef\txorpdfstring
11634 {
11635   \newcommand*\Glsfmtfull}[1]{%
11636     \txorpdfstring
11637     {\Glsxtrtitlefull{#1}}%
11638     {\Glsxtrinlinefullformat{#1}{}}%
11639   }
11640 }
```

```

11639 }
11640 }
11641 {
11642   \newcommand*{\Glsfmtfull}[1]{%
11643     \Glsxtrtitlefull{#1}}
11644 }

```

`\glsfmtfullpl` In-line full plural format.

```

11645 \ifdef\texorpdfstring
11646 {
11647   \newcommand*{\glsfmtfullpl}[1]{%
11648     \texorpdfstring
11649     {\Glsxtrtitlefullpl{#1}}%
11650     {\Glsxtrinlinefullplformat{#1}{}}%
11651   }
11652 }
11653 {
11654   \newcommand*{\glsfmtfullpl}[1]{%
11655     \Glsxtrtitlefullpl{#1}}
11656 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11657 \ifdef\texorpdfstring
11658 {
11659   \newcommand*{\Glsfmtfullpl}[1]{%
11660     \texorpdfstring
11661     {\Glsxtrtitlefullpl{#1}}%
11662     {\Glsxtrinlinefullplformat{#1}{}}%
11663   }
11664 }
11665 {
11666   \newcommand*{\Glsfmtfullpl}[1]{%
11667     \Glsxtrtitlefullpl{#1}}
11668 }

```

## 1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11669 \newcommand*{\RequireGlossariesExtraLang}[1]{%
11670   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11671 }

```

`sariesExtraLang`

```

11672 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11673   \ProvidesFile{glossariesxtr-#1.ldf}%

```



11674 }

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.)

`\glstr@loaddialect` The dialect label should be stored in `\this@dialect` before using this command.

```
11675 \newcommand{\glstr@loaddialect}{%
11676   \IfTrackedLanguageFileExists{\this@dialect}%
11677   {glossariesxtr-}% prefix
11678   {.ldf}%
11679   {%
11680     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11681   }%
11682   {}% not found
```

If `glossaries-extra-bib2gls` has been loaded, `\@glstrdialecthook` will check for the associated script, otherwise it will do nothing.

```
11683 \glstrdialecthook
11684 }
```

```
11685 \ifpackageloaded{tracklang}
11686 {%
11687   \AnyTrackedLanguages
11688   {%
11689     \ForEachTrackedDialect{\this@dialect}{\glstr@loaddialect}%
11690   }%
11691   {}%
11692 }
11693 }
```

Load `glossaries-extra-stylemods` if required.

```
11694 \@glstr@redefstyles
```

and set the style:

```
11695 \@glstr@do@style
```

## 1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```
11696 \NeedsTeXFormat{LaTeX2e}
11697 \ProvidesPackage{glossaries-extra-bib2gls}[2018/05/24 v1.32 (NLCT)]
```

These are some convenient macros for use with custom rules.

`\glshex`

```
11698 \newcommand*{\glshex}{\string\u}
```

`\glscapturedgroup`

```
11699 \newcommand*{\glscapturedgroup}{\string\$}
```

`\nZeroChildCount` For use with bib2gls's save-child-count resource option.

```
11700 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11701   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11702 }
```

`\rprovidecommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```
11703 \newcommand*{\glsxtrprovidecommand}{\providecommand}
```

`\lossarylocation` For use with indexcounter and bib2gls.

```
11704 \newcommand*{\glsxtr@wrglossarylocation}[2]{#1}
```

`\IndexCounterLink` `\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```
11705 \ifdef\hyperref
11706 {%
11707   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
11708     \glsxtrifhasfield{indexcounter}{#2}%
11709     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11710     {#1}%
11711   }
11712 }
11713 {
11714   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
11715 }
```

`\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.

```
11716 \newcommand*{\GlsXtrDualField}{dual}
```

`\sXtrDualBackLink` `\GlsXtrDualBackLink{<text>}{<label>}`

Adds a hyperlink to the dual entry.

```
11717 \newcommand*{\GlsXtrDualBackLink}[2]{%}
```

```

11718 \glstrifhasfield{\GlsXtrDualField}{#2}%
11719 {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11720 {#2}%
11721 }

```

**TeXEntryAliases** Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```

11722 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
11723   article=bibtexentry,
11724   book=bibtexentry,
11725   booklet=bibtexentry,
11726   conference=bibtexentry,
11727   inbook=bibtexentry,
11728   incollection=bibtexentry,
11729   inproceedings=bibtexentry,
11730   manual=bibtexentry,
11731   mastersthesis=bibtexentry,
11732   misc=bibtexentry,
11733   phdthesis=bibtexentry,
11734   proceedings=bibtexentry,
11735   techreport=bibtexentry,
11736   unpublished=bibtexentry
11737 }

```

**ideBibTeXFields** Convenient shortcut to define the standard BibTeX fields.

```

11738 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11739   \glsaddstoragekey{address}{\glstrbibaddress}%
11740   \glsaddstoragekey{author}{\glstrbibauthor}%
11741   \glsaddstoragekey{booktitle}{\glstrbibbooktitle}%
11742   \glsaddstoragekey{chapter}{\glstrbibchapter}%
11743   \glsaddstoragekey{edition}{\glstrbibedition}%
11744   \glsaddstoragekey{howpublished}{\glstrbibhowpublished}%
11745   \glsaddstoragekey{institution}{\glstrbibinstitution}%
11746   \glsaddstoragekey{journal}{\glstrbibjournal}%
11747   \glsaddstoragekey{month}{\glstrbibmonth}%
11748   \glsaddstoragekey{note}{\glstrbibnote}%
11749   \glsaddstoragekey{number}{\glstrbibnumber}%
11750   \glsaddstoragekey{organization}{\glstrbiborganization}%
11751   \glsaddstoragekey{pages}{\glstrbibpages}%
11752   \glsaddstoragekey{publisher}{\glstrbibpublisher}%
11753   \glsaddstoragekey{school}{\glstrbibschooll}%
11754   \glsaddstoragekey{series}{\glstrbibseries}%
11755   \glsaddstoragekey{title}{\glstrbibtitle}%
11756   \glsaddstoragekey{bibtextype}{\glstrbibtype}%
11757   \glsaddstoragekey{volume}{\glstrbibvolume}%
11758 }

```

Provide missing Greek letters for use in maths mode. These commands are recognised by bib2gls and will be mapped to the Mathematical Greek Italic letters. This ensures that the

Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The  $\TeX$  version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

$\backslash$ Alpha

```
11759 \providecommand*\Alpha{\mathrm{A}}
```

$\backslash$ Beta

```
11760 \providecommand*\Beta{\mathrm{B}}
```

$\backslash$ Epsilon

```
11761 \providecommand*\Epsilon{\mathrm{E}}
```

$\backslash$ Zeta

```
11762 \providecommand*\Zeta{\mathrm{Z}}
```

$\backslash$ Eta

```
11763 \providecommand*\Eta{\mathrm{H}}
```

$\backslash$ Iota

```
11764 \providecommand*\Iota{\mathrm{I}}
```

$\backslash$ Kappa

```
11765 \providecommand*\Kappa{\mathrm{K}}
```

$\backslash$ Mu

```
11766 \providecommand*\Mu{\mathrm{M}}
```

$\backslash$ Nu

```
11767 \providecommand*\Nu{\mathrm{N}}
```

$\backslash$ Omicron

```
11768 \providecommand*\Omicron{\mathrm{O}}
```

$\backslash$ Rho

```
11769 \providecommand*\Rho{\mathrm{P}}
```

$\backslash$ Tau

```
11770 \providecommand*\Tau{\mathrm{T}}
```

$\backslash$ Chi

```
11771 \providecommand*\Chi{\mathrm{X}}
```

$\backslash$ Digamma

```
11772 \providecommand*\Digamma{\mathrm{F}}
```

```

\omicron
11773 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11774 \@ifpackageloaded{upgreek}%
11775 {

\Upalpha
11776 \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11777 \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11778 \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11779 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11780 \providecommand*\Upeta{\mathrm{H}}

\Upiota
11781 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11782 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11783 \providecommand*\Upmu{\mathrm{M}}

\Upnu
11784 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11785 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11786 \providecommand*\Uprho{\mathrm{P}}

\Uptau
11787 \providecommand*\Uptau{\mathrm{T}}

\Upchi
11788 \providecommand*\Upchi{\mathrm{X}}

```

`\upomicron`

```
11789 \providecommand*\upomicron{\mathrm{o}}
11790 }%
11791 {}% upgreek.sty not loaded
```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-⟨tag⟩.ldf` (where `⟨tag⟩` identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of `⟨tag⟩`. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxtrcontrolrules
; \glxtrspacerules
; \glxtrnonprintablerules
; \glxtrcombiningdiacriticrules
, \glxtrhyphenrules
< \glxtrgeneralpuncrules
< \glxtrdigitrules
< \glxtrfractionrules
< \glxtrGeneralLatinIVrules
< \glxtrMathItalicGreekIrules
}
```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they’ve been made active.

```
11792 \newcommand*\glxtrcontrolrules{%
11793 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
11794 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
11795 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
11796 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
11797 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
11798 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
11799 0010\string'\string=\glshex 0011
11800 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11801 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11802 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11803 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11804 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
```

```

11805 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11806 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11807 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11808 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11809 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11810 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11811 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11812 }

```

**lsxtrspacerules** These are space characters.

```

11813 \newcommand*{\glsxtrspacerules}{%
11814 \string' \string'\string;
11815 \string'\glshex 00A0\string'\string;
11816 \string'\glshex 2000\string'\string;
11817 \string'\glshex 2001\string'\string;
11818 \string'\glshex 2002\string'\string;
11819 \string'\glshex 2003\string'\string;
11820 \string'\glshex 2004\string'\string;
11821 \string'\glshex 2005\string'\string;
11822 \string'\glshex 2006\string'\string;
11823 \string'\glshex 2007\string'\string;
11824 \string'\glshex 2008\string'\string;
11825 \string'\glshex 2009\string'\string;
11826 \string'\glshex 200A\string'\string;
11827 \string'\glshex 3000\string'
11828 }

```

**nonprintablerules** These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11829 \newcommand*{\glsxtrnonprintablerules}{%
11830 \string'\glshex FEFF\string'\string;
11831 \string'\glshex 000A\string'\string;
11832 \string'\glshex 0009\string'\string;
11833 \string'\glshex 000C\string'\string;
11834 \string'\glshex 000B\string'
11835 }

```

**diacriticrules** Combining diacritic marks. This is split into multiple macros.

```

11836 \newcommand*{\glsxtrcombiningdiacriticrules}{%
11837 \glsxtrcombiningdiacriticIrules\string;
11838 \glsxtrcombiningdiacriticIIrules\string;
11839 \glsxtrcombiningdiacriticIIIrules\string;
11840 \glsxtrcombiningdiacriticIVrules
11841 }

```

**diacriticIrules** First set of combining diacritic marks.

```

11842 \newcommand*{\glsxtrcombiningdiacriticIrules}{%
11843 \glshex 0301\string;% combining acute
11844 \glshex 0300\string;% combining grave
11845 \glshex 0306\string;% combining breve

```

```

11846 \glshex 0302\string;% combining circumflex
11847 \glshex 030C\string;% combining caron
11848 \glshex 030A\string;% combining ring
11849 \glshex 030D\string;% combining vertical line above
11850 \glshex 0308\string;% combining diaeresis
11851 \glshex 030B\string;% combining double acute
11852 \glshex 0303\string;% combining tilde
11853 \glshex 0307\string;% combining dot above
11854 \glshex 0304% combining macron
11855 }

```

iacriticIIrules Second set of combining diacritic marks.

```

11856 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
11857 \glshex 0337\string;% combining short solidus overlay
11858 \glshex 0327\string;% combining cedilla
11859 \glshex 0328\string;% combining ogonek
11860 \glshex 0323\string;% combining dot below
11861 \glshex 0332\string;% combining low line
11862 \glshex 0305\string;% combining overline
11863 \glshex 0309\string;% combining hook above
11864 \glshex 030E\string;% combining double vertical line above
11865 \glshex 030F\string;% combining double grave accent
11866 \glshex 0310\string;% combining candrabindu
11867 \glshex 0311\string;% combining inverted breve
11868 \glshex 0312\string;% combining turned comma above
11869 \glshex 0313\string;% combining comma above
11870 \glshex 0314\string;% combining reversed comma above
11871 \glshex 0315\string;% combining comma above right
11872 \glshex 0316\string;% combining grave accent below
11873 \glshex 0317% combining acute accent below
11874 }

```

acriticIIIrules Third set of combining diacritic marks.

```

11875 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
11876 \glshex 0318\string;% combining left tack below
11877 \glshex 0319\string;% combining right tack below
11878 \glshex 031A\string;% combining left angle above
11879 \glshex 031B\string;% combining horn
11880 \glshex 031C\string;% combining left half ring below
11881 \glshex 031D\string;% combining up tack below
11882 \glshex 031E\string;% combining down tack below
11883 \glshex 031F\string;% combining plus sign below
11884 \glshex 0320\string;% combining minus sign below
11885 \glshex 0321\string;% combining palatalized hook below
11886 \glshex 0322\string;% combining retroflex hook below
11887 \glshex 0324\string;% combining diaeresis below
11888 \glshex 0325\string;% combining ring below
11889 \glshex 0326\string;% combining comma below
11890 \glshex 0329\string;% combining vertical line below

```



11891 \glshex 032A\string;% combining bridge below  
 11892 \glshex 032B\string;% combining inverted double arch below  
 11893 \glshex 032C\string;% combining caron below  
 11894 \glshex 032D\string;% combining circumflex accent below  
 11895 \glshex 032E\string;% combining breve below  
 11896 \glshex 032F\string;% combining inverted breve below  
 11897 \glshex 0330\string;% combining tilde below  
 11898 \glshex 0331\string;% combining macron below  
 11899 \glshex 0333\string;% combining double low line  
 11900 \glshex 0334\string;% combining tilde overlay  
 11901 \glshex 0335\string;% combining short stroke overlay  
 11902 \glshex 0336\string;% combining long stroke overlay  
 11903 \glshex 0338\string;% combining long solidus overlay  
 11904 \glshex 0339\string;% combining combining right half ring below  
 11905 \glshex 033A\string;% combining inverted bridge below  
 11906 \glshex 033B\string;% combining square below  
 11907 \glshex 033C\string;% combining seagull below  
 11908 \glshex 033D\string;% combining x above  
 11909 \glshex 033E\string;% combining vertical tilde  
 11910 \glshex 033F\string;% combining double overline  
 11911 \glshex 0342\string;% combining Greek perispomeni  
 11912 \glshex 0344\string;% combining Greek dialytika tonos  
 11913 \glshex 0345\string;% combining Greek ypogegrammeni  
 11914 \glshex 0360\string;% combining double tilde  
 11915 \glshex 0361\string;% combining double inverted breve  
 11916 \glshex 0483\string;% combining Cyrillic titlo  
 11917 \glshex 0484\string;% combining Cyrillic palatalization  
 11918 \glshex 0485\string;% combining Cyrillic dasia pneumata  
 11919 \glshex 0486% combining Cyrillic psili pneumata  
 11920 }

iacriticIVrules Fourth set of combining diacritic marks.

11921 \newcommand\*{\glxstrcombiningdiacriticIVrules}{%  
 11922 \glshex 20D0\string;% combining left harpoon above  
 11923 \glshex 20D1\string;% combining right harpoon above  
 11924 \glshex 20D2\string;% combining long vertical line overlay  
 11925 \glshex 20D3\string;% combining short vertical line overlay  
 11926 \glshex 20D4\string;% combining anticlockwise arrow above  
 11927 \glshex 20D5\string;% combining clockwise arrow above  
 11928 \glshex 20D6\string;% combining left arrow above  
 11929 \glshex 20D7\string;% combining right arrow above  
 11930 \glshex 20D8\string;% combining ring overlay  
 11931 \glshex 20D9\string;% combining clockwise ring overlay  
 11932 \glshex 20DA\string;% combining anticlockwise ring overlay  
 11933 \glshex 20DB\string;% combining three dots above  
 11934 \glshex 20DC\string;% combining four dots above  
 11935 \glshex 20DD\string;% combining enclosing circle  
 11936 \glshex 20DE\string;% combining enclosing square  
 11937 \glshex 20DF\string;% combining enclosing diamond

```

11938 \glshex 20E0\string;% combining enclosing circle backslash
11939 \glshex 20E1% combining left right arrow above
11940 }

```

#### sxtrhyphenrules Hyphens.

```

11941 \newcommand*{\glsxtrhyphenrules}{%
11942 \string'\string-\string'\string;% ASCII hyphen
11943 \glshex 00AD\string;% soft hyphen
11944 \glshex 2010\string;% hyphen
11945 \glshex 2011\string;% non-breaking hyphen
11946 \glshex 2012\string;% figure dash
11947 \glshex 2013\string;% en dash
11948 \glshex 2014\string;% em dash
11949 \glshex 2015\string;% horizontal bar
11950 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11951 }

```

#### eneralpuncrules General punctuation.

```

11952 \newcommand*{\glsxtrgeneralpuncrules}{%
11953 \glsxtrgeneralpuncIrules
11954 \string<\glsxtrcurrencyrules
11955 \string<\glsxtrgeneralpuncIIrules
11956 }

```

#### eneralpuncIrules First set of general punctuation.

```

11957 \newcommand*{\glsxtrgeneralpuncIrules}{%
11958 \string'\glshex 005F\string'% underscore
11959 \string<\glshex 00AF% macron
11960 \string<\string'\glshex 002C\string'% comma
11961 \string<\string'\glshex 003B\string'% semi-colon
11962 \string<\string'\glshex 003A\string'% colon
11963 \string<\string'\glshex 0021\string'% exclamation mark
11964 \string<\glshex 00A1% inverted exclamation mark
11965 \string<\string'\glshex 003F\string'% question mark
11966 \string<\glshex 00BF% inverted question mark
11967 \string<\string'\glshex 002F\string'% solidus
11968 \string<\string'\glshex 002E\string'% full stop
11969 \string<\glshex 00B4% acute accent
11970 \string<\string'\glshex 0060\string'% grave accent
11971 \string<\string'\glshex 005E\string'% circumflex accent
11972 \string<\glshex 00A8% diaeresis
11973 \string<\string'\glshex 007E\string'% tilde
11974 \string<\glshex 00B7% middle dot
11975 \string<\glshex 00B8% cedilla
11976 \string<\string'\glshex 0027\string'% straight apostrophe
11977 \string<\string'\glshex 0022\string'% straight double quote
11978 \string<\glshex 00AB% left guillemet
11979 \string<\glshex 00BB% right guillemet
11980 \string<\string'\glshex 0028\string'% left parenthesis

```

```

11981 \string=\glshex 207D\string=\glshex 208D% super/subscript left parenthesis
11982 \string<\string'\glshex 0029\string'% right parenthesis
11983 \string=\glshex 207E\string=\glshex 208E% super/subscript right parenthesis
11984 \string<\string'\glshex 005B\string'% left square bracket
11985 \string<\string'\glshex 005D\string'% right square bracket
11986 \string<\string'\glshex 007B\string'% left curly bracket
11987 \string<\string'\glshex 007D\string'% right curly bracket
11988 \string<\glshex 00A7% section sign
11989 \string<\glshex 00B6% pilcrow sign
11990 \string<\glshex 00A9% copyright sign
11991 \string<\glshex 00AE% registered sign
11992 \string<\string'\glshex 0040\string'% at sign
11993 }

```

trcurrencyrules General punctuation.

```

11994 \newcommand*{\glxtrcurrencyrules}{%
11995 \glshex 00A4% currency sign
11996 \string<\glshex 0E3F% Thai currency symbol baht
11997 \string<\glshex 00A2% cent sign
11998 \string<\glshex 20A1% colon sign
11999 \string<\glshex 20A2% cruzeiro sign
12000 \string<\string'\glshex 0024\string'% dollar sign
12001 \string<\glshex 20AB% dong sign
12002 \string<\glshex 20AC% euro sign
12003 \string<\glshex 20A3% French franc sign
12004 \string<\glshex 20A4% lira sign
12005 \string<\glshex 20A5% mill sign
12006 \string<\glshex 20A6% naira sign
12007 \string<\glshex 20A7% peseta sign
12008 \string<\glshex 00A3% pound sign
12009 \string<\glshex 20A8% rupee sign
12010 \string<\glshex 20AA% new sheqel sign
12011 \string<\glshex 20A9% won sign
12012 \string<\glshex 00A5% yen sign
12013 }

```

eralpuncIIrules Second set of general punctuation.

```

12014 \newcommand*{\glxtrgeneralpuncIIrules}{%
12015 \string'\glshex 002A\string'% asterisk
12016 \string<\string'\glshex 005C\string'% backslash
12017 \string<\string'\glshex 0026\string'% ampersand
12018 \string<\string'\glshex 0023\string'% hash sign
12019 \string<\string'\glshex 0025\string'% percent sign
12020 \string<\string'\glshex 002B\string'% plus sign
12021 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12022 \string<\glshex 00B1% plus-minus sign
12023 \string<\glshex 00F7% division sign
12024 \string<\glshex 00D7% multiplication sign
12025 \string<\string'\glshex 003C\string'% less-than sign

```

```

12026 \string<\string'\glshex 003D\string'% equals sign
12027 \string<\string'\glshex 003E\string'% greater-than sign
12028 \string<\glshex 00AC% not sign
12029 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12030 \string<\glshex 00A6% broken bar
12031 \string<\glshex 00B0% degree sign
12032 \string<\glshex 00B5% micron sign
12033 }

```

**GeneralLatinIrules** Basic Latin alphabet.

```

12034 \newcommand*{\glxtrGeneralLatinIrules}{%
12035 \glxtrLatinA
12036 \string<b,B%
12037 \string<c,C%
12038 \string<d,D%
12039 \string<\glxtrLatinE
12040 \string<f,F%
12041 \string<g,G%
12042 \string<\glxtrLatinH
12043 \string<\glxtrLatinI
12044 \string<j,J%
12045 \string<\glxtrLatinK
12046 \string<\glxtrLatinL
12047 \string<\glxtrLatinM
12048 \string<\glxtrLatinN
12049 \string<\glxtrLatinO
12050 \string<\glxtrLatinP
12051 \string<q,Q%
12052 \string<r,R%
12053 \string<\glxtrLatinS
12054 \string<\glxtrLatinT
12055 \string<u,U%
12056 \string<v,V%
12057 \string<w,W%
12058 \string<\glxtrLatinX
12059 \string<y,Y%
12060 \string<z,Z
12061 }

```

**GeneralLatinIIrules** General Latin alphabet (eth between D and E, ß treated as SS).

```

12062 \newcommand*{\glxtrGeneralLatinIIrules}{%
12063 \glxtrLatinA
12064 \string<b,B%
12065 \string<c,C%
12066 \string<d,D%
12067 \string<\glxtrLatinEth
12068 \string<\glxtrLatinE
12069 \string<f,F%
12070 \string<g,G%

```

```

12071 \string<\glxtrLatinH
12072 \string<\glxtrLatinI
12073 \string<j,J%
12074 \string<\glxtrLatinK
12075 \string<\glxtrLatinL
12076 \string<\glxtrLatinM
12077 \string<\glxtrLatinN
12078 \string<\glxtrLatinO
12079 \string<\glxtrLatinP
12080 \string<q,Q%
12081 \string<r,R%
12082 \string<\glxtrLatinS
12083 \string& SS \string, \glxtrLatinEszettSs
12084 \string<\glxtrLatinT
12085 \string<u,U%
12086 \string<v,V%
12087 \string<w,W%
12088 \string<\glxtrLatinX
12089 \string<y,Y%
12090 \string<z,Z%
12091 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

12092 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12093 \glxtrLatinA
12094 \string<b,B%
12095 \string<c,C%
12096 \string<d,D%
12097 \string<\glxtrLatinEth
12098 \string<\glxtrLatinE
12099 \string<f,F%
12100 \string<g,G%
12101 \string<\glxtrLatinH
12102 \string<\glxtrLatinI
12103 \string<j,J%
12104 \string<\glxtrLatinK
12105 \string<\glxtrLatinL
12106 \string<\glxtrLatinM
12107 \string<\glxtrLatinN
12108 \string<\glxtrLatinO
12109 \string<\glxtrLatinP
12110 \string<q,Q%
12111 \string<r,R%
12112 \string<\glxtrLatinS
12113 \string& SZ, \glxtrLatinEszettSz
12114 \string<\glxtrLatinT
12115 \string<u,U%
12116 \string<v,V%
12117 \string<w,W%

```

```

12118 \string<\glxtrLatinX
12119 \string<y,Y%
12120 \string<z,Z%
12121 }

```

**GeneralLatinIVrules** General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12122 \newcommand*{\glxtrGeneralLatinIVrules}{%
12123 \glxtrLatinA
12124 \string& AE , \glxtrLatinAELigature
12125 \string<b,B%
12126 \string<c,C%
12127 \string<d,D%
12128 \string<\glxtrLatinEth
12129 \string<\glxtrLatinE
12130 \string<f,F%
12131 \string<g,G%
12132 \string<\glxtrLatinH
12133 \string<\glxtrLatinI
12134 \string<j,J%
12135 \string<\glxtrLatinK
12136 \string<\glxtrLatinL
12137 \string<\glxtrLatinM
12138 \string<\glxtrLatinN
12139 \string<\glxtrLatinO
12140 \string& OE , \glxtrLatinOELigature
12141 \string<\glxtrLatinP
12142 \string<q,Q%
12143 \string<r,R%
12144 \string<\glxtrLatinS
12145 \string& SS , \glxtrLatinEszettSs
12146 \string<\glxtrLatinT
12147 \string& th =\glshex 00DE
12148 \string& TH =\glshex 00FE
12149 \string<u,U%
12150 \string<v,V%
12151 \string<w,W%
12152 \string<\glxtrLatinX
12153 \string<y,Y%
12154 \string<z,Z%
12155 }

```

**GeneralLatinVrules** General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12156 \newcommand*{\glxtrGeneralLatinVrules}{%
12157 \glxtrLatinA
12158 \string<b,B%
12159 \string<c,C%
12160 \string<d,D%
12161 \string<\glxtrLatinEth

```

```

12162 \string<\glxtrLatinE
12163 \string<f,F%
12164 \string<g,G%
12165 \string<\glxtrLatinH
12166 \string<\glxtrLatinI
12167 \string<j,J%
12168 \string<\glxtrLatinK
12169 \string<\glxtrLatinL
12170 \string<\glxtrLatinM
12171 \string<\glxtrLatinN
12172 \string<\glxtrLatinO
12173 \string<\glxtrLatinP
12174 \string<q,Q%
12175 \string<r,R%
12176 \string<\glxtrLatinS
12177 \string& SS , \glxtrLatinEszettSs
12178 \string<\glxtrLatinT
12179 \string& th =\glshex 00DE
12180 \string& TH =\glshex 00FE
12181 \string<u,U%
12182 \string<v,V%
12183 \string<w,W%
12184 \string<\glxtrLatinX
12185 \string<y,Y%
12186 \string<z,Z%
12187 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12188 \newcommand*{\glxtrGeneralLatinVIrules}{%
12189 \glxtrLatinA
12190 \string<b,B%
12191 \string<c,C%
12192 \string<d,D%
12193 \string<\glxtrLatinEth
12194 \string<\glxtrLatinE
12195 \string<f,F%
12196 \string<g,G%
12197 \string<\glxtrLatinH
12198 \string<\glxtrLatinI
12199 \string<j,J%
12200 \string<\glxtrLatinK
12201 \string<\glxtrLatinL
12202 \string<\glxtrLatinM
12203 \string<\glxtrLatinN
12204 \string<\glxtrLatinO
12205 \string<\glxtrLatinP
12206 \string<q,Q%
12207 \string<r,R%
12208 \string<\glxtrLatinS

```

```

12209 \string& SZ , \glxtrLatinEszettSz
12210 \string<\glxtrLatinT
12211 \string& th =\glshex 00DE
12212 \string& TH =\glshex 00FE
12213 \string<u,U%
12214 \string<v,V%
12215 \string<w,W%
12216 \string<\glxtrLatinX
12217 \string<y,Y%
12218 \string<z,Z%
12219 }

```

**allLatinVIIrules** General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, Ć between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

12220 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12221 \glxtrLatinA
12222 \string<\glxtrLatinAELigature
12223 \string<b,B%
12224 \string<c,C%
12225 \string<d,D%
12226 \string<\glxtrLatinEth
12227 \string<\glxtrLatinE
12228 \string<f,F%
12229 \string<\glxtrLatinInsularG
12230 \string<\glxtrLatinH
12231 \string<\glxtrLatinI
12232 \string<j,J%
12233 \string<\glxtrLatinK
12234 \string<\glxtrLatinL
12235 \string<\glxtrLatinM
12236 \string<\glxtrLatinN
12237 \string<\glxtrLatinO
12238 \string<\glxtrLatinOELigature
12239 \string<\glxtrLatinP
12240 \string<q,Q%
12241 \string<r,R%
12242 \string<\glshex 017F=\glxtrLatinS % s and long s
12243 \string<\glxtrLatinT
12244 \string<\glxtrLatinThorn
12245 \string<u,U%
12246 \string<v,V%
12247 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12248 \string<\glxtrLatinX
12249 \string<y,Y%
12250 \string<z,Z%
12251 }

```

**allLatinVIIIrules** General Latin alphabet (Æ treated as AE and Ć treated as OE, Þ treated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).



```

12252 \newcommand*{\glxtrGeneralLatinVIIIRules}{%
12253   \glxtrLatinA
12254   \string& AE , \glxtrLatinAELigature
12255   \string<b,B%
12256   \string<c,C%
12257   \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12258   \string<\glxtrLatinE
12259   \string<f,F%
12260   \string<g,G%
12261   \string<\glxtrLatinH
12262   \string<\glxtrLatinI
12263   \string<j,J%
12264   \string<\glxtrLatinK
12265   \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12266   \string<\glxtrLatinM
12267   \string<\glxtrLatinN
12268   \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12269   \string& OE , \glxtrLatinOELigature
12270   \string<\glxtrLatinP
12271   \string<q,Q%
12272   \string<r,R%
12273   \string<\glxtrLatinS
12274   \string& SS , \glxtrLatinEszettSs
12275   \string<\glxtrLatinT
12276   \string& th =\glshex 00DE
12277   \string& TH =\glshex 00FE
12278   \string<u,U%
12279   \string<v,V%
12280   \string<w,W%
12281   \string<\glxtrLatinX
12282   \string<y,Y%
12283   \string<z,Z%
12284 }

```

#### \glxtrLatinA

```

12285 \newcommand*{\glxtrLatinA}{%
12286   a\string=\glshex 00AA\string=\glshex 2090,A
12287 }

```

#### \glxtrLatinE

```

12288 \newcommand*{\glxtrLatinE}{%
12289   e\string=\glshex 2091,E
12290 }

```

#### \glxtrLatinH

```

12291 \newcommand*{\glxtrLatinH}{%
12292   h\string=\glshex 2095,H
12293 }

```

\glsxtrLatinI

```
12294 \newcommand*{\glsxtrLatinI}{%  
12295   i\string=\glshex 2071,I  
12296 }
```

\glsxtrLatinK

```
12297 \newcommand*{\glsxtrLatinK}{%  
12298   k\string=\glshex 2096,K  
12299 }
```

\glsxtrLatinL

```
12300 \newcommand*{\glsxtrLatinL}{%  
12301   l\string=\glshex 2097,L  
12302 }
```

\glsxtrLatinM

```
12303 \newcommand*{\glsxtrLatinM}{%  
12304   m\string=\glshex 2098,M  
12305 }
```

\glsxtrLatinN

```
12306 \newcommand*{\glsxtrLatinN}{%  
12307   n\string=\glshex 207F\string=\glshex 2099,N  
12308 }
```

\glsxtrLatinO

```
12309 \newcommand*{\glsxtrLatinO}{%  
12310   o\string=\glshex 00BA\string=\glshex 2092,O  
12311 }
```

\glsxtrLatinP

```
12312 \newcommand*{\glsxtrLatinP}{%  
12313   p\string=\glshex 209A,P  
12314 }
```

\glsxtrLatinS

```
12315 \newcommand*{\glsxtrLatinS}{%  
12316   s\string=\glshex 209B,S  
12317 }
```

\glsxtrLatinT

```
12318 \newcommand*{\glsxtrLatinT}{%  
12319   t\string=\glshex 209C,T  
12320 }
```

\glsxtrLatinX

```
12321 \newcommand*{\glsxtrLatinX}{%  
12322   x\string=\glshex 2093,X  
12323 }
```

lsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).  
12324 \newcommand\*{\glxtrLatinSchwa}{%  
12325 \glshex 0259\string=\glshex 2094,\glshex 018F  
12326 }

trLatinEszettSs  
12327 \newcommand\*{\glxtrLatinEszettSs}{%  
12328 \glshex 00DF% eszett  
12329 \string=\glshex 017Fs % long S s  
12330 }

trLatinEszettSz  
12331 \newcommand\*{\glxtrLatinEszettSz}{%  
12332 \glshex 00DF% eszett  
12333 \string= \glshex 017Fz % long S z  
12334 }

\glxtrLatinEth  
12335 \newcommand\*{\glxtrLatinEth}{%  
12336 \glshex 00F0,\glshex 00D0% eth  
12337 }

lsxtrLatinThorn  
12338 \newcommand\*{\glxtrLatinThorn}{%  
12339 \glshex 00FE,\glshex 00DE% thorn  
12340 }

LatinAELigature  
12341 \newcommand\*{\glxtrLatinAELigature}{%  
12342 \glshex 00E6,\glshex 00C6% AE-ligature  
12343 }

LatinOELigature  
12344 \newcommand\*{\glxtrLatinOELigature}{%  
12345 \glshex 0153,\glshex 0152% OE-ligature  
12346 }

\glxtrLatinAA  
12347 \newcommand\*{\glxtrLatinAA}{%  
12348 \glshex 00E5=a\glshex 030A,% \aa  
12349 \glshex 00C5=A\glshex 030A% \AA  
12350 }

glxtrLatinWynn  
12351 \newcommand\*{\glxtrLatinWynn}{%  
12352 \glshex 01BF,\glshex 01F7% wynn  
12353 }

trLatinInsularG

```
12354 \newcommand*{\glxtrLatinInsularG}{%
12355   \glshex 1D79,\glshex A77D% insular G
12356   \string; g, G
12357 }
```

sxtrLatinOslash

```
12358 \newcommand*{\glxtrLatinOslash}{%
12359   \glshex 00F8,\glshex 00D8% \o, \O
12360 }
```

sxtrLatinLslash

```
12361 \newcommand*{\glxtrLatinLslash}{%
12362   \glshex 0142,\glshex 0141% \l, \L
12363 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12364 \newcommand*{\glxtrMathUpGreekIrules}{%
12365   \glxtrUpAlpha
12366   \string<\glxtrUpBeta
12367   \string<\glxtrUpGamma
12368   \string<\glxtrUpDelta
12369   \string<\glxtrUpEpsilon
12370   \string<\glxtrUpDigamma
12371   \string<\glxtrUpZeta
12372   \string<\glxtrUpEta
12373   \string<\glxtrUpTheta
12374   \string<\glxtrUpIota
12375   \string<\glxtrUpKappa
12376   \string<\glxtrUpLambda
12377   \string<\glxtrUpMu
12378   \string<\glxtrUpNu
12379   \string<\glxtrUpXi
12380   \string<\glxtrUpOmicron
12381   \string<\glxtrUpPi
12382   \string<\glxtrUpRho
12383   \string<\glxtrUpSigma
12384   \string<\glxtrUpTau
12385   \string<\glxtrUpUpsilon
12386   \string<\glxtrUpPhi
12387   \string<\glxtrUpChi
12388   \string<\glxtrUpPsi
12389   \string<\glxtrUpOmega
12390 }
```

hUpGreekIIrules Doesn't include digamma.

```
12391 \newcommand*{\glxtrMathUpGreekIIrules}{%
12392   \glxtrUpAlpha
12393   \string<\glxtrUpBeta
```

```

12394 \string<\glxtrUpGamma
12395 \string<\glxtrUpDelta
12396 \string<\glxtrUpEpsilon
12397 \string<\glxtrUpZeta
12398 \string<\glxtrUpEta
12399 \string<\glxtrUpTheta
12400 \string<\glxtrUpIota
12401 \string<\glxtrUpKappa
12402 \string<\glxtrUpLambda
12403 \string<\glxtrUpMu
12404 \string<\glxtrUpNu
12405 \string<\glxtrUpXi
12406 \string<\glxtrUpOmicron
12407 \string<\glxtrUpPi
12408 \string<\glxtrUpRho
12409 \string<\glxtrUpSigma
12410 \string<\glxtrUpTau
12411 \string<\glxtrUpUpsilon
12412 \string<\glxtrUpPhi
12413 \string<\glxtrUpChi
12414 \string<\glxtrUpPsi
12415 \string<\glxtrUpOmega
12416 }

```

**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.

```

12417 \newcommand*{\glxtrMathItalicGreekIrules}{%
12418 \glxtrMathItalicAlpha
12419 \string<\glxtrMathItalicBeta
12420 \string<\glxtrMathItalicGamma
12421 \string<\glxtrMathItalicDelta
12422 \string<\glxtrMathItalicEpsilon
12423 \string<\glxtrUpDigamma
12424 \string<\glxtrMathItalicZeta
12425 \string<\glxtrMathItalicEta
12426 \string<\glxtrMathItalicTheta
12427 \string<\glxtrMathItalicIota
12428 \string<\glxtrMathItalicKappa
12429 \string<\glxtrMathItalicLambda
12430 \string<\glxtrMathItalicMu
12431 \string<\glxtrMathItalicNu
12432 \string<\glxtrMathItalicXi
12433 \string<\glxtrMathItalicOmicron
12434 \string<\glxtrMathItalicPi
12435 \string<\glxtrMathItalicRho
12436 \string<\glxtrMathItalicSigma
12437 \string<\glxtrMathItalicTau
12438 \string<\glxtrMathItalicUpsilon
12439 \string<\glxtrMathItalicPhi

```

```

12440 \string<\glxtrMathItalicChi
12441 \string<\glxtrMathItalicPsi
12442 \string<\glxtrMathItalicOmega
12443 }

```

licGreekIIrules Doesn't include digamma.

```

12444 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12445 \glxtrMathItalicAlpha
12446 \string<\glxtrMathItalicBeta
12447 \string<\glxtrMathItalicGamma
12448 \string<\glxtrMathItalicDelta
12449 \string<\glxtrMathItalicEpsilon
12450 \string<\glxtrMathItalicZeta
12451 \string<\glxtrMathItalicEta
12452 \string<\glxtrMathItalicTheta
12453 \string<\glxtrMathItalicIota
12454 \string<\glxtrMathItalicKappa
12455 \string<\glxtrMathItalicLambda
12456 \string<\glxtrMathItalicMu
12457 \string<\glxtrMathItalicNu
12458 \string<\glxtrMathItalicXi
12459 \string<\glxtrMathItalicOmicron
12460 \string<\glxtrMathItalicPi
12461 \string<\glxtrMathItalicRho
12462 \string<\glxtrMathItalicSigma
12463 \string<\glxtrMathItalicTau
12464 \string<\glxtrMathItalicUpsilon
12465 \string<\glxtrMathItalicPhi
12466 \string<\glxtrMathItalicChi
12467 \string<\glxtrMathItalicPsi
12468 \string<\glxtrMathItalicOmega
12469 }

```

pperGreekIrules Upper case only (includes upright digamma).

```

12470 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12471 \glshex 1D6E2% upper case alpha (maths italic)
12472 \string<\glshex 1D6E3% upper case beta (maths italic)
12473 \string<\glshex 1D6E4% upper case gamma (maths italic)
12474 \string<\glshex 1D6E5% upper case delta (maths italic)
12475 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12476 \string<\glshex 03DC% upper case digamma
12477 \string<\glshex 1D6E7% upper case zeta (maths italic)
12478 \string<\glshex 1D6E8% upper case eta (maths italic)
12479 \string<\glshex 1D6E9% upper case theta (maths italic)
12480 \string=<\glshex 1D6F3% upper case theta variant (maths italic)
12481 \string<\glshex 1D6EA% upper case iota (maths italic)
12482 \string<\glshex 1D6EB% upper case kappa (maths italic)
12483 \string<\glshex 1D6EC% upper case lambda (maths italic)
12484 \string<\glshex 1D6ED% upper case mu (maths italic)

```

```

12485 \string<\glshex 1D6EE% upper case nu (maths italic)
12486 \string<\glshex 1D6EF% upper case xi (maths italic)
12487 \string<\glshex 1D6F0% upper case omicron (maths italic)
12488 \string<\glshex 1D6F1% upper case pi (maths italic)
12489 \string<\glshex 1D6F2% upper case rho (maths italic)
12490 \string<\glshex 1D6F4% upper case sigma (maths italic)
12491 \string<\glshex 1D6F5% upper case tau (maths italic)
12492 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12493 \string<\glshex 1D6F7% upper case phi (maths italic)
12494 \string<\glshex 1D6F8% upper case chi (maths italic)
12495 \string<\glshex 1D6F9% upper case psi (maths italic)
12496 \string<\glshex 1D6FA% upper case omega (maths italic)
12497 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

12498 \newcommand*{\glstrMathItalicUpperGreekIIrules}{%
12499 \glshex 1D6E2% upper case alpha (maths italic)
12500 \string<\glshex 1D6E3% upper case beta (maths italic)
12501 \string<\glshex 1D6E4% upper case gamma (maths italic)
12502 \string<\glshex 1D6E5% upper case delta (maths italic)
12503 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12504 \string<\glshex 1D6E7% upper case zeta (maths italic)
12505 \string<\glshex 1D6E8% upper case eta (maths italic)
12506 \string<\glshex 1D6E9% upper case theta (maths italic)
12507 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12508 \string<\glshex 1D6EA% upper case iota (maths italic)
12509 \string<\glshex 1D6EB% upper case kappa (maths italic)
12510 \string<\glshex 1D6EC% upper case lambda (maths italic)
12511 \string<\glshex 1D6ED% upper case mu (maths italic)
12512 \string<\glshex 1D6EE% upper case nu (maths italic)
12513 \string<\glshex 1D6EF% upper case xi (maths italic)
12514 \string<\glshex 1D6F0% upper case omicron (maths italic)
12515 \string<\glshex 1D6F1% upper case pi (maths italic)
12516 \string<\glshex 1D6F2% upper case rho (maths italic)
12517 \string<\glshex 1D6F4% upper case sigma (maths italic)
12518 \string<\glshex 1D6F5% upper case tau (maths italic)
12519 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12520 \string<\glshex 1D6F7% upper case phi (maths italic)
12521 \string<\glshex 1D6F8% upper case chi (maths italic)
12522 \string<\glshex 1D6F9% upper case psi (maths italic)
12523 \string<\glshex 1D6FA% upper case omega (maths italic)
12524 }

```

owerGreekIrules Lower case only (includes upright digamma).

```

12525 \newcommand*{\glstrMathItalicLowerGreekIrules}{%
12526 \glshex 1D6FC% lower case alpha (maths italic)
12527 \string<\glshex 1D6FD% lower case beta (maths italic)
12528 \string<\glshex 1D6FE% lower case gamma (maths italic)
12529 \string<\glshex 1D6FF% lower case delta (maths italic)

```

```

12530 \string<\glshex 1D700% lower case epsilon (maths italic)
12531 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12532 \string<\glshex 03DD% lower case digamma
12533 \string<\glshex 1D701% lower case zeta (maths italic)
12534 \string<\glshex 1D702% lower case eta (maths italic)
12535 \string<\glshex 1D703% lower case theta (maths italic)
12536 \string=\glshex 1D717% lower case theta variant (maths italic)
12537 \string<\glshex 1D704% lower case iota (maths italic)
12538 \string<\glshex 1D705% lower case kappa (maths italic)
12539 \string=\glshex 1D718% lower case kappa variant (maths italic)
12540 \string<\glshex 1D706% lower case lambda (maths italic)
12541 \string<\glshex 1D707% lower case mu (maths italic)
12542 \string<\glshex 1D708% lower case nu (maths italic)
12543 \string<\glshex 1D709% lower case xi (maths italic)
12544 \string<\glshex 1D70A% lower case omicron (maths italic)
12545 \string<\glshex 1D70B% lower case pi (maths italic)
12546 \string=\glshex 1D71B% lower case pi variant (maths italic)
12547 \string<\glshex 1D70C% lower case rho (maths italic)
12548 \string=\glshex 1D71A% lower case rho variant (maths italic)
12549 \string<\glshex 1D70D% lower case final sigma (maths italic)
12550 \string=\glshex 1D70E% lower case sigma (maths italic)
12551 \string<\glshex 1D70F% lower case tau (maths italic)
12552 \string<\glshex 1D710% lower case upsilon (maths italic)
12553 \string<\glshex 1D711% lower case phi (maths italic)
12554 \string=\glshex 1D719% lower case phi variant (maths italic)
12555 \string<\glshex 1D712% lower case chi (maths italic)
12556 \string<\glshex 1D713% lower case psi (maths italic)
12557 \string<\glshex 1D714% lower case omega (maths italic)
12558 }

```

LowerGreekIIrules Lower case only (doesn't includes upright digamma).

```

12559 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
12560 \glshex 1D6FC% lower case alpha (maths italic)
12561 \string<\glshex 1D6FD% lower case beta (maths italic)
12562 \string<\glshex 1D6FE% lower case gamma (maths italic)
12563 \string<\glshex 1D6FF% lower case delta (maths italic)
12564 \string<\glshex 1D700% lower case epsilon (maths italic)
12565 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12566 \string<\glshex 1D701% lower case zeta (maths italic)
12567 \string<\glshex 1D702% lower case eta (maths italic)
12568 \string<\glshex 1D703% lower case theta (maths italic)
12569 \string=\glshex 1D717% lower case theta variant (maths italic)
12570 \string<\glshex 1D704% lower case iota (maths italic)
12571 \string<\glshex 1D705% lower case kappa (maths italic)
12572 \string=\glshex 1D718% lower case kappa variant (maths italic)
12573 \string<\glshex 1D706% lower case lambda (maths italic)
12574 \string<\glshex 1D707% lower case mu (maths italic)
12575 \string<\glshex 1D708% lower case nu (maths italic)
12576 \string<\glshex 1D709% lower case xi (maths italic)

```



```

12577 \string<\glshex 1D70A% lower case omicron (maths italic)
12578 \string<\glshex 1D70B% lower case pi (maths italic)
12579 \string=<\glshex 1D71B% lower case pi variant (maths italic)
12580 \string<\glshex 1D70C% lower case rho (maths italic)
12581 \string=<\glshex 1D71A% lower case rho variant (maths italic)
12582 \string<\glshex 1D70D% lower case final sigma (maths italic)
12583 \string=<\glshex 1D70E% lower case sigma (maths italic)
12584 \string<\glshex 1D70F% lower case tau (maths italic)
12585 \string<\glshex 1D710% lower case upsilon (maths italic)
12586 \string<\glshex 1D711% lower case phi (maths italic)
12587 \string=<\glshex 1D719% lower case phi variant (maths italic)
12588 \string<\glshex 1D712% lower case chi (maths italic)
12589 \string<\glshex 1D713% lower case psi (maths italic)
12590 \string<\glshex 1D714% lower case omega (maths italic)
12591 }

```

**MathGreekIrules** Includes both upright and italic with digamma between epsilon and zeta.

```

12592 \newcommand*{\glxtrMathGreekIrules}{%
12593   \glxtrMathItalicAlpha
12594   \string;\glxtrUpAlpha
12595   \string<\glxtrMathItalicBeta
12596   \string;\glxtrUpBeta
12597   \string<\glxtrMathItalicGamma
12598   \string;\glxtrUpGamma
12599   \string<\glxtrMathItalicDelta
12600   \string;\glxtrUpDelta
12601   \string<\glxtrMathItalicEpsilon
12602   \string;\glxtrUpEpsilon
12603   \string<\glxtrUpDigamma
12604   \string<\glxtrMathItalicZeta
12605   \string;\glxtrUpZeta
12606   \string<\glxtrMathItalicEta
12607   \string;\glxtrUpEta
12608   \string<\glxtrMathItalicTheta
12609   \string;\glxtrUpTheta
12610   \string<\glxtrMathItalicIota
12611   \string;\glxtrUpIota
12612   \string<\glxtrMathItalicKappa
12613   \string;\glxtrUpKappa
12614   \string<\glxtrMathItalicLambda
12615   \string;\glxtrUpLambda
12616   \string<\glxtrMathItalicMu
12617   \string;\glxtrUpMu
12618   \string<\glxtrMathItalicNu
12619   \string;\glxtrUpNu
12620   \string<\glxtrMathItalicXi
12621   \string;\glxtrUpXi
12622   \string<\glxtrMathItalicOmicron
12623   \string;\glxtrUpOmicron

```

```

12624 \string<\glxtrMathItalicPi
12625 \string;\glxtrUpPi
12626 \string<\glxtrMathItalicRho
12627 \string;\glxtrUpRho
12628 \string<\glxtrMathItalicSigma
12629 \string;\glxtrUpSigma
12630 \string<\glxtrMathItalicTau
12631 \string;\glxtrUpTau
12632 \string<\glxtrMathItalicUpsilon
12633 \string;\glxtrUpUpsilon
12634 \string<\glxtrMathItalicPhi
12635 \string;\glxtrUpPhi
12636 \string<\glxtrMathItalicChi
12637 \string;\glxtrUpChi
12638 \string<\glxtrMathItalicPsi
12639 \string;\glxtrUpPsi
12640 \string<\glxtrMathItalicOmega
12641 \string;\glxtrUpOmega
12642 }

```

**mathGreekIIrules** Includes both upright and italic (digamma not included).

```

12643 \newcommand*{\glxtrMathGreekIIrules}{%
12644 \glxtrMathItalicAlpha
12645 \string;\glxtrUpAlpha
12646 \string<\glxtrMathItalicBeta
12647 \string;\glxtrUpBeta
12648 \string<\glxtrMathItalicGamma
12649 \string;\glxtrUpGamma
12650 \string<\glxtrMathItalicDelta
12651 \string;\glxtrUpDelta
12652 \string<\glxtrMathItalicEpsilon
12653 \string;\glxtrUpEpsilon
12654 \string<\glxtrMathItalicZeta
12655 \string;\glxtrUpZeta
12656 \string<\glxtrMathItalicEta
12657 \string;\glxtrUpEta
12658 \string<\glxtrMathItalicTheta
12659 \string;\glxtrUpTheta
12660 \string<\glxtrMathItalicIota
12661 \string;\glxtrUpIota
12662 \string<\glxtrMathItalicKappa
12663 \string;\glxtrUpKappa
12664 \string<\glxtrMathItalicLambda
12665 \string;\glxtrUpLambda
12666 \string<\glxtrMathItalicMu
12667 \string;\glxtrUpMu
12668 \string<\glxtrMathItalicNu
12669 \string;\glxtrUpNu
12670 \string<\glxtrMathItalicXi

```

```

12671 \string;\glxtrUpXi
12672 \string<\glxtrMathItalicOmicron
12673 \string;\glxtrUpOmicron
12674 \string<\glxtrMathItalicPi
12675 \string;\glxtrUpPi
12676 \string<\glxtrMathItalicRho
12677 \string;\glxtrUpRho
12678 \string<\glxtrMathItalicSigma
12679 \string;\glxtrUpSigma
12680 \string<\glxtrMathItalicTau
12681 \string;\glxtrUpTau
12682 \string<\glxtrMathItalicUpsilon
12683 \string;\glxtrUpUpsilon
12684 \string<\glxtrMathItalicPhi
12685 \string;\glxtrUpPhi
12686 \string<\glxtrMathItalicChi
12687 \string;\glxtrUpChi
12688 \string<\glxtrMathItalicPsi
12689 \string;\glxtrUpPsi
12690 \string<\glxtrMathItalicOmega
12691 \string;\glxtrUpOmega
12692 }

```

\glxtrUpAlpha

```

12693 \newcommand*{\glxtrUpAlpha}{%
12694 \glshex 03B1,% lower case alpha
12695 \glshex 0391% upper case alpha
12696 }

```

\glxtrUpBeta

```

12697 \newcommand*{\glxtrUpBeta}{%
12698 \glshex 03B2,% lower case beta
12699 \glshex 0392% upper case beta
12700 }

```

\glxtrUpGamma

```

12701 \newcommand*{\glxtrUpGamma}{%
12702 \glshex 03B3,% lower case gamma
12703 \glshex 0393% upper case gamma
12704 }

```

\glxtrUpDelta

```

12705 \newcommand*{\glxtrUpDelta}{%
12706 \glshex 03B4,% lower case delta
12707 \glshex 0394% upper case delta
12708 }

```

\glxtrUpEpsilon

```

12709 \newcommand*{\glxtrUpEpsilon}{%
12710 \glshex 03B5% lower case epsilon
12711 \string=\glshex 03F5,% lower case epsilon variant
12712 \glshex 0395% upper case epsilon
12713 }

```

glxtrUpDigamma

```

12714 \newcommand*{\glxtrUpDigamma}{%
12715 \glshex 03DD,% lower case digamma
12716 \glshex 03DC% upper case digamma
12717 }

```

\glxtrUpZeta

```

12718 \newcommand*{\glxtrUpZeta}{%
12719 \glshex 03B6,% lower case zeta
12720 \glshex 0396% upper case zeta
12721 }

```

\glxtrUpEta

```

12722 \newcommand*{\glxtrUpEta}{%
12723 \glshex 03B7,% lower case eta
12724 \glshex 0397% upper case eta
12725 }

```

\glxtrUpTheta

```

12726 \newcommand*{\glxtrUpTheta}{%
12727 \glshex 03B8% lower case theta
12728 \string=\glshex 03D1,% lower case theta variant
12729 \glshex 0398% upper case theta
12730 }

```

\glxtrUpIota

```

12731 \newcommand*{\glxtrUpIota}{%
12732 \glshex 03B9,% lower case iota
12733 \glshex 0399% upper case iota
12734 }

```

\glxtrUpKappa

```

12735 \newcommand*{\glxtrUpKappa}{%
12736 \glshex 03BA% lower case kappa
12737 \string=\glshex 03F0,% lower case kappa variant
12738 \glshex 039A% upper case kappa
12739 }

```

\glxtrUpLambda

```

12740 \newcommand*{\glxtrUpLambda}{%
12741 \glshex 03BB,% lower lambda
12742 \glshex 039B% upper case lambda
12743 }

```

`\glxtrUpMu`

```
12744 \newcommand*{\glxtrUpMu}{%
12745   \glshex 03BC,% lower case mu
12746   \glshex 039C% upper case mu
12747 }
```

`\glxtrUpNu`

```
12748 \newcommand*{\glxtrUpNu}{%
12749   \glshex 03BD,% lower case nu
12750   \glshex 039D% upper case nu
12751 }
```

`\glxtrUpXi`

```
12752 \newcommand*{\glxtrUpXi}{%
12753   \glshex 03BE,% lower case xi
12754   \glshex 039E% upper case xi
12755 }
```

`\glxtrUpOmicron`

```
12756 \newcommand*{\glxtrUpOmicron}{%
12757   \glshex 03BF,% lower case omicron
12758   \glshex 039F% upper case omicron
12759 }
```

`\glxtrUpPi`

```
12760 \newcommand*{\glxtrUpPi}{%
12761   \glshex 03C0% lower case pi
12762   \string=\glshex 03D6,% lower case pi variant
12763   \glshex 03A0% upper case pi
12764 }
```

`\glxtrUpRho`

```
12765 \newcommand*{\glxtrUpRho}{%
12766   \glshex 03C1% lower case rho
12767   \string=\glshex 03F1,% lower case rho variant
12768   \glshex 03A1% upper case rho
12769 }
```

`\glxtrUpSigma`

```
12770 \newcommand*{\glxtrUpSigma}{%
12771   \glshex 03C2% lower case sigma
12772   \string=\glshex 03C3,% lower case sigma
12773   \glshex 03A3% upper case sigma
12774 }
```

`\glxtrUpTau`

```
12775 \newcommand*{\glxtrUpTau}{%
12776   \glshex 03C4,% lower case tau
```

```
12777 \glshex 03A4% upper case tau
12778 }
```

glsxtrUpUpsilon

```
12779 \newcommand*{\glxtrUpUpsilon}{%
12780 \glshex 03C5,% lower case upsilon
12781 \glshex 03A5% upper case upsilon
12782 }
```

\glxtrUpPhi

```
12783 \newcommand*{\glxtrUpPhi}{%
12784 \glshex 03C6% lower case phi
12785 \string=\glshex 03D5,% lower case phi variant
12786 \glshex 03A6% upper case phi
12787 }
```

\glxtrUpChi

```
12788 \newcommand*{\glxtrUpChi}{%
12789 \glshex 03C7,% lower case chi
12790 \glshex 03A7% upper case chi
12791 }
```

\glxtrUpPsi

```
12792 \newcommand*{\glxtrUpPsi}{%
12793 \glshex 03C8,% lower case psi
12794 \glshex 03A8% upper case psi
12795 }
```

\glxtrUpOmega

```
12796 \newcommand*{\glxtrUpOmega}{%
12797 \glshex 03C9,% lower case omega
12798 \glshex 03A9% upper case omega
12799 }
```

MathItalicAlpha

```
12800 \newcommand*{\glxtrMathItalicAlpha}{%
12801 \glshex 1D6FC,% lower case alpha (maths italic)
12802 \glshex 1D6E2% upper case alpha (maths italic)
12803 }
```

rMathItalicBeta

```
12804 \newcommand*{\glxtrMathItalicBeta}{%
12805 \glshex 1D6FD,% lower case beta (maths italic)
12806 \glshex 1D6E3% upper case beta (maths italic)
12807 }
```

MathItalicGamma

```
12808 \newcommand*{\glxtrMathItalicGamma}{%
```

```

12809 \glshex 1D6FE,% lower case gamma (maths italic)
12810 \glshex 1D6E4% upper case gamma (maths italic)
12811 }

```

MathItalicDelta

```

12812 \newcommand*{\glsxtrMathItalicDelta}{%
12813 \glshex 1D6FF,% lower case delta (maths italic)
12814 \glshex 1D6E5% upper case delta (maths italic)
12815 }

```

thItalicEpsilon

```

12816 \newcommand*{\glsxtrMathItalicEpsilon}{%
12817 \glshex 1D700% lower case epsilon (maths italic)
12818 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12819 \glshex 1D6E6% upper case epsilon (maths italic)
12820 }

```

rMathItalicZeta

```

12821 \newcommand*{\glsxtrMathItalicZeta}{%
12822 \glshex 1D701,% lower case zeta (maths italic)
12823 \glshex 1D6E7% upper case zeta (maths italic)
12824 }

```

trMathItalicEta

```

12825 \newcommand*{\glsxtrMathItalicEta}{%
12826 \glshex 1D702,% lower case eta (maths italic)
12827 \glshex 1D6E8% upper case eta (maths italic)
12828 }

```

MathItalicTheta

```

12829 \newcommand*{\glsxtrMathItalicTheta}{%
12830 \glshex 1D703% lower case theta (maths italic)
12831 \string=\glshex 1D717,% lower case theta variant (maths italic)
12832 \glshex 1D6E9% upper case theta (maths italic)
12833 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12834 }

```

rMathItalicIota

```

12835 \newcommand*{\glsxtrMathItalicIota}{%
12836 \glshex 1D704,% lower case iota (maths italic)
12837 \glshex 1D6EA% upper case iota (maths italic)
12838 }

```

MathItalicKappa

```

12839 \newcommand*{\glsxtrMathItalicKappa}{%
12840 \glshex 1D705% lower case kappa (maths italic)
12841 \string=\glshex 1D718,% lower case kappa variant (maths italic)
12842 \glshex 1D6EB% upper case kappa (maths italic)
12843 }

```

athItalicLambda

```
12844 \newcommand*{\glxtrMathItalicLambda}{%
12845   \glshex 1D706,% lower case lambda (maths italic)
12846   \glshex 1D6EC% upper case lambda (maths italic)
12847 }
```

xtrMathItalicMu

```
12848 \newcommand*{\glxtrMathItalicMu}{%
12849   \glshex 1D707,% lower case mu (maths italic)
12850   \glshex 1D6ED% upper case mu (maths italic)
12851 }
```

xtrMathItalicNu

```
12852 \newcommand*{\glxtrMathItalicNu}{%
12853   \glshex 1D708,% lower case nu (maths italic)
12854   \glshex 1D6EE% upper case nu (maths italic)
12855 }
```

xtrMathItalicXi

```
12856 \newcommand*{\glxtrMathItalicXi}{%
12857   \glshex 1D709,% lower case xi (maths italic)
12858   \glshex 1D6EF% upper case xi (maths italic)
12859 }
```

thItalicOmicron

```
12860 \newcommand*{\glxtrMathItalicOmicron}{%
12861   \glshex 1D70A,% lower case omicron (maths italic)
12862   \glshex 1D6F0% upper case omicron (maths italic)
12863 }
```

xtrMathItalicPi

```
12864 \newcommand*{\glxtrMathItalicPi}{%
12865   \glshex 1D70B% lower case pi (maths italic)
12866   \string=\glshex 1D71B,% lower case pi variant (maths italic)
12867   \glshex 1D6F1% upper case pi (maths italic)
12868 }
```

trMathItalicRho

```
12869 \newcommand*{\glxtrMathItalicRho}{%
12870   \glshex 1D70C% lower case rho (maths italic)
12871   \string=\glshex 1D71A,% lower case rho variant (maths italic)
12872   \glshex 1D6F2% upper case rho (maths italic)
12873 }
```

MathItalicSigma

```
12874 \newcommand*{\glxtrMathItalicSigma}{%
12875   \glshex 1D70D% lower case final sigma (maths italic)
12876   \string=\glshex 1D70E,% lower case sigma (maths italic)
```



```

12877 \glshex 1D6F4% upper case sigma (maths italic)
12878 }

```

trMathItalicTau

```

12879 \newcommand*{\glxtrMathItalicTau}{%
12880 \glshex 1D70F,% lower case tau (maths italic)
12881 \glshex 1D6F5% upper case tau (maths italic)
12882 }

```

thItalicUpsilon

```

12883 \newcommand*{\glxtrMathItalicUpsilon}{%
12884 \glshex 1D710,% lower case upsilon (maths italic)
12885 \glshex 1D6F6% upper case upsilon (maths italic)
12886 }

```

trMathItalicPhi

```

12887 \newcommand*{\glxtrMathItalicPhi}{%
12888 \glshex 1D711% lower case phi (maths italic)
12889 \string=\glshex 1D719,% lower case phi variant (maths italic)
12890 \glshex 1D6F7% upper case phi (maths italic)
12891 }

```

trMathItalicChi

```

12892 \newcommand*{\glxtrMathItalicChi}{%
12893 \glshex 1D712,% lower case chi (maths italic)
12894 \glshex 1D6F8% upper case chi (maths italic)
12895 }

```

trMathItalicPsi

```

12896 \newcommand*{\glxtrMathItalicPsi}{%
12897 \glshex 1D713,% lower case psi (maths italic)
12898 \glshex 1D6F9% upper case psi (maths italic)
12899 }

```

MathItalicOmega

```

12900 \newcommand*{\glxtrMathItalicOmega}{%
12901 \glshex 1D714,% lower case omega (maths italic)
12902 \glshex 1D6FA% upper case omega (maths italic)
12903 }

```

thItalicPartial

```

12904 \newcommand*{\glxtrMathItalicPartial}{%
12905 \glshex 1D715% partial differential (maths italic)
12906 }

```

MathItalicNabla

```

12907 \newcommand*{\glxtrMathItalicNabla}{%
12908 \glshex 1D6FB% nabla (maths italic)
12909 }

```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12910 \newcommand*{\glxtrdigitrules}{%
12911 0\string=\glshex 2080\string=\glshex 2070
12912 \string<1\string=\glshex 2081\string=\glshex 00B9
12913 \string<2\string=\glshex 2082\string=\glshex 00B2
12914 \string<3\string=\glshex 2083\string=\glshex 00B3
12915 \string<4\string=\glshex 2084\string=\glshex 2074
12916 \string<5\string=\glshex 2085\string=\glshex 2075
12917 \string<6\string=\glshex 2086\string=\glshex 2076
12918 \string<7\string=\glshex 2087\string=\glshex 2077
12919 \string<8\string=\glshex 2088\string=\glshex 2078
12920 \string<9\string=\glshex 2089\string=\glshex 2079
12921 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12922 \newcommand*{\glxtrBasicDigitrules}{%
12923 0\string<1\string<2\string<3\string<4%
12924 \string<5\string<6\string<7\string<8\string<9%
12925 }
```

criptDigitrules Subscript digits.

```
12926 \newcommand*{\glxtrSubScriptDigitrules}{%
12927 \glshex 2080% subscript 0
12928 \string<\glshex 2081% subscript 1
12929 \string<\glshex 2082% subscript 2
12930 \string<\glshex 2083% subscript 3
12931 \string<\glshex 2084% subscript 4
12932 \string<\glshex 2085% subscript 5
12933 \string<\glshex 2086% subscript 6
12934 \string<\glshex 2087% subscript 7
12935 \string<\glshex 2088% subscript 8
12936 \string<\glshex 2089% subscript 9
12937 }
```

criptDigitrules Superscript digits.

```
12938 \newcommand*{\glxtrSuperScriptDigitrules}{%
12939 \glshex 2070% superscript 0
12940 \string<\glshex 00B9% superscript 1
12941 \string<\glshex 00B2% superscript 2
12942 \string<\glshex 00B3% superscript 3
12943 \string<\glshex 2074% superscript 4
12944 \string<\glshex 2075% superscript 5
12945 \string<\glshex 2076% superscript 6
12946 \string<\glshex 2077% superscript 7
12947 \string<\glshex 2078% superscript 8
12948 \string<\glshex 2079% superscript 9
12949 }
```

trfractionrules   Vulgar fractions.

```
12950 \newcommand*{\glxtrfractionrules}{%
12951   \glshex 215F% fraction numerator one (1/)
12952   \string<\glshex 2189% zero thirds (0/3 = 0)
12953   \string<\glshex 2152% one tenth (1/10 = 0.1)
12954   \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12955   \string<\glshex 215B% one eighth (1/8 = 0.125)
12956   \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12957   \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12958   \string<\glshex 2155% one fifth (1/5 = 0.2)
12959   \string<\glshex 00BC% one quarter (1/4 = 0.25)
12960   \string<\glshex 2153% one third (1/3 ~ 0.333)
12961   \string<\glshex 215C% three eighths (3/8 = 0.375)
12962   \string<\glshex 2156% two fifths (2/5 = 0.4)
12963   \string<\glshex 00BD% one half (1/2 = 0.5)
12964   \string<\glshex 2157% three fifths (3/5 = 0.6)
12965   \string<\glshex 215D% five eighths (5/8 = 0.625)
12966   \string<\glshex 2154% two thirds (2/3 ~ 0.667)
12967   \string<\glshex 00BE% three quarters (3/4 = 0.75)
12968   \string<\glshex 2158% four fifths (4/5 = 0.8)
12969   \string<\glshex 215A% five sixths (5/6 ~ 0.833)
12970   \string<\glshex 215E% seven eighths (7/8 = 0.875)
12971 }
```

sxtrdialecthook   Check for scripts associated with the document dialects.

```
12972 \renewcommand{\@glxtrdialecthook}{%
12973   \ifundef\CurrentTrackedScript
12974   {%
12975     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
12976     {%
12977       \edef\CurrentTrackedScript{%
12978         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
12979       }%
12980     }%
12981   }%
12982   {}%
12983   \ifdef\CurrentTrackedScript
12984   {%
12985     \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
12986     \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
12987     \let\CurrentTrackedTag\CurrentTrackedScript
12988     \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
12989     {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
12990     {}%
12991     \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
12992   }%
12993   {}%
12994 }
```

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.

```
12995 \ifdef\glsxtr@loaddialect
12996 {%
12997   \@ifpackageloaded{tracklang}
12998   {%
12999     \AnyTrackedLanguages
13000     {%
13001       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13002     }%
13003   }%
13004 }
13005 {}
13006 }
13007 {}
```

## 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:

```
13008 \NeedsTeXFormat{LaTeX2e}
```

```
13009 \ProvidesPackage{glossaries-extra-stylemods}[2018/05/24 v1.32 (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
```

```
13010 \newcommand*{\@glxtr@loadstyles}{}
```

all Provide all known styles.

```
13011 \DeclareOption{all}{%
```

```
13012   \appto\@glxtr@loadstyles{%
```

```
13013     \RequirePackage{glossary-inline}%
```

```
13014     \RequirePackage{glossary-list}%
```

```
13015     \RequirePackage{glossary-tree}%
```

```
13016     \RequirePackage{glossary-mcols}%
```

```
13017     \RequirePackage{glossary-long}%
```

```
13018     \RequirePackage{glossary-longragged}%
```

```
13019     \RequirePackage{glossary-longbooktabs}%
```

```
13020     \RequirePackage{glossary-super}%
```

```
13021     \RequirePackage{glossary-superragged}%
```

```
13022     \RequirePackage{glossary-bookindex}%
```

```
13023   }
```

```
13024 }
```

```
13025 \DeclareOption*{%
```

```
13026   \IfFileExists{glossary-\CurrentOption.sty}
```

```
13027   {\eappto\@glxtr@loadstyles{%
```

```
13028     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
```

```
13029   }%
```

```
13030   {%
```

```
13031     \PackageError{glossaries-extra-styles}%
```

```

13032     {Unknown option ‘\CurrentOption’}{}%
13033   }%
13034 }

```

Process the package options:

```
13035 \ProcessOptions
```

Load the required packages:

```
13036 \@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`.

```
13037 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

13038 \providecommand{\renewglossarystyle}[2]{%
13039   \ifcsundef{@glsstyle@#1}%
13040   {%
13041     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13042   }%
13043   {%
13044     \csdef{@glsstyle@#1}{#2}%
13045   }%
13046 }

```

## 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.

```

13047 \ifdef{@glsstyle@listdotted}
13048 {%
13049   \renewglossarystyle{listdotted}{%
13050     \setglossarystyle{list}%
13051     \renewcommand*{\glossentry}[2]{%
13052       \item[]\makebox[\glslistdottedwidth][l]{%
13053         \glsentryitem{##1}%
13054         \glstarget{##1}{\glossentryname{##1}}%
13055         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13056         \glossentrydesc{##1}\glspostdescription}%
13057     \renewcommand*{\subglossentry}[3]{%
13058       \item[]\makebox[\glslistdottedwidth][l]{%
13059         \glssubentryitem{##2}%
13060         \glstarget{##2}{\glossentryname{##2}}%
13061         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13062         \glossentrydesc{##2}\glspostdescription}%
13063   }

```

```
13064 }
13065 {%
```

Assume the style isn't required if it hasn't already been defined.

```
13066 }
```

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.

```
13067 \ifdef{\@glsstyle@list}
13068 {%
```

listprelocation Space before number list for top-level entries.

```
13069 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13070 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13071 \newcommand{\glslistchildpostlocation}{.}
```

```
\glslistdesc
```

```
13072 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
13073 \renewglossarystyle{list}{%
13074   \renewenvironment{theglossary}%
13075     {\begin{description}}{\end{description}}%
13076   \renewcommand*\{glossaryheader}[1]{%
13077     \renewcommand*\{glsgroupheading}[1]{%
13078       \renewcommand*\{glossentry}[2]{%
13079         \item[\glssentryitem{##1}%
13080           \glstarget{##1}{\glossentryname{##1}}]
13081         \glslistdesc{##1}\glslistprelocation ##2}%
13082       \renewcommand*\{subglossentry}[3]{%
13083         \glssubentryitem{##2}%
13084         \glstarget{##2}{\strut}\space
13085         \glslistdesc{##2}%
13086         \glslistchildprelocation ##3\glslistchildpostlocation}%
13087       \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13088     }
13089 }
13090 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13091 \ifdef{\@glsstyle@altlist}
13092 {%
13093   \renewglossarystyle{altlist}{%
```

```

13094 \setglossarystyle{list}%
13095 \renewcommand*{\glossentry}[2]{%
13096   \item[\glentryitem{##1}%
13097     \glstarget{##1}{\glossentryname{##1}}]%
13098   \mbox{}\par\nobreak\@afterheading
13099   \glslistdesc{##1}\glslistprelocation ##2}%
13100 \renewcommand{\subglossentry}[3]{%
13101   \par
13102   \glssubentryitem{##2}%
13103   \glstarget{##2}{\strut}\glslistdesc{##2}%
13104   \glslistchildprelocation ##3}%
13105 }
13106 }
13107 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13108 \ifdef{\@glsstyle@listgroup}
13109 {%
13110   \renewglossarystyle{listgroup}{%
13111     \setglossarystyle{list}%
13112     \renewcommand*{\glsgroupheading}[1]{%
13113       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13114       \mbox{}\par\nobreak\@afterheading
13115     }%
13116   }
13117 }
13118 {}

```

Similarly for listhypergroup.

```

13119 \ifdef{\@glsstyle@listhypergroup}
13120 {%
13121   \renewglossarystyle{listhypergroup}{%
13122     \setglossarystyle{list}%
13123     \renewcommand*{\glossaryheader}{%
13124       \glslistnavigationitem{\glsnavigation}}%
13125     \renewcommand*{\glsgroupheading}[1]{%
13126       \item[\glslistgroupheaderfmt
13127         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13128       \mbox{}\par\nobreak\@afterheading
13129     }%
13130   }
13131 }
13132 {}

```

Similarly for altlistgroup.

```

13133 \ifdef{\@glsstyle@altlistgroup}
13134 {%
13135   \renewglossarystyle{altlistgroup}{%
13136     \setglossarystyle{altlist}%
13137     \renewcommand*{\glsgroupheading}[1]{%
13138       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```



```

13139      \mbox{}\par\nobreak\@afterheading
13140    }%
13141  }
13142 }
13143 {}

```

Similarly for altlisthypergroup.

```

13144 \ifdef{\@glsstyle@altlisthypergroup}
13145 {%
13146   \renewglossarystyle{altlisthypergroup}{%
13147     \setglossarystyle{altlist}%
13148     \renewcommand*\glossaryheader{%
13149       \glslistnavigationitem{\glsnavigation}}%
13150     \renewcommand*\glsgroupheading}[1]{%
13151       \item[\glslistgroupheaderfmt
13152         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13153       \mbox{}\par\nobreak\@afterheading
13154     }%
13155   }
13156 }
13157 {}

```

## 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`.

```

13158 \ifcsdef{@glsstyle@long}
13159 {%
13160   \renewglossarystyle{long}{%
13161     \renewenvironment{theglossary}%
13162       {\begin{longtable}[lp{\glsdescwidth}}%
13163       {\end{longtable}}%
13164     \renewcommand*\glossaryheader{}%
13165     \renewcommand*\glsgroupheading}[1]{}%
13166     \renewcommand{\glossentry}[2]{%
13167       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13168       \glossentrydesc{##1}\glspostdescription
13169       \glxtrprelocation ##2\tabularnewline
13170     }%
13171     \renewcommand{\subglossentry}[3]{%
13172       &
13173       \glssubentryitem{##2}%
13174       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13175       \glxtrprelocation ##3\tabularnewline
13176     }%
13177     \ifglsnogroupskip
13178       \renewcommand*\glsgroupskip{}%
13179     \else

```

```

13180      \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13181      \fi
13182    }
13183  }
13184 {}

```

Three column style:

```

13185 \ifcsdef{@glsstyle@long3col}
13186 {%
13187   \renewglossarystyle{long3col}{%
13188     \renewenvironment{theglossary}%
13189       {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13190     {\end{longtable}}}%
13191   \renewcommand*{\glossaryheader}{}%
13192   \renewcommand*{\glsgroupheading}[1]{}%
13193   \renewcommand{\glossentry}[2]{%
13194     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13195     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13196   }%
13197   \renewcommand{\subglossentry}[3]{%
13198     &
13199     \glssubentryitem{##2}%
13200     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13201     ##3\tabularnewline
13202   }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

13203   \ifglsnogroupskip
13204     \renewcommand*{\glsgroupskip}{}%
13205   \else
13206     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
13207   \fi
13208 }
13209 }
13210 {}

```

Four column style:

```

13211 \ifcsdef{@glsstyle@long4col}
13212 {%
13213   \renewglossarystyle{long4col}{%
13214     \renewenvironment{theglossary}%
13215       {\begin{longtable}{llll}}}%
13216     {\end{longtable}}}%
13217   \renewcommand*{\glossaryheader}{}%
13218   \renewcommand*{\glsgroupheading}[1]{}%
13219   \renewcommand{\glossentry}[2]{%
13220     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13221     \glossentrydesc{##1}\glspostdescription &
13222     \glossentrysymbol{##1} &
13223     ##2\tabularnewline

```

```

13224 }%
13225 \renewcommand{\subglossentry}[3]{%
13226     &
13227     \glssubentryitem{##2}%
13228     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13229     \glossentrysymbol{##2} & ##3\tabularnewline
13230 }%

13231 \ifglsgroupskip
13232     \renewcommand*\{glsgroupskip}{}%
13233 \else
13234     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13235 \fi
13236 }
13237 }
13238 {}

```

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`.

```

13239 \ifcsdef{@glstyle@longragged}
13240 {%
13241     \renewglossarystyle{longragged}{%
13242         \renewenvironment{theglossary}%
13243             {\begin{longtable}[l>\raggedright]p{\glstdescwidth}}}%
13244             {\end{longtable}}}%
13245     \renewcommand*\{glossaryheader}{}%
13246     \renewcommand*\{glsgroupheading}[1]{}%
13247     \renewcommand{\glossentry}[2]{%
13248         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13249         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13250         \tabularnewline
13251     }%
13252     \renewcommand{\subglossentry}[3]{%
13253         &
13254         \glssubentryitem{##2}%
13255         \glstarget{##2}{\strut}\glossentrydesc{##2}%
13256         \glspostdescription\glxtrprelocation ##3%
13257         \tabularnewline
13258     }%
13259     \ifglsgroupskip
13260         \renewcommand*\{glsgroupskip}{}%
13261     \else
13262         \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13263 \fi
13264 }
13265 }
13266 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13267 \ifcsdef{@glsstyle@longragged3col}
13268 {%
13269 \renewglossarystyle{longragged3col}{%
13270 \renewenvironment{theglossary}%
13271 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13272 >{\raggedright}p{\glspagelistwidth}}}%
13273 {\end{longtable}}}%
13274 \renewcommand*{\glossaryheader}{}%
13275 \renewcommand*{\glsgroupheading}[1]{}%
13276 \renewcommand{\glossentry}[2]{%
13277 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13278 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13279 }%
13280 \renewcommand{\subglossentry}[3]{%
13281 &
13282 \glssubentryitem{##2}%
13283 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13284 ##3\tabularnewline
13285 }%
13286 \ifglsgroupskip
13287 \renewcommand*{\glsgroupskip}{}%
13288 \else
13289 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13290 \fi
13291 }
13292 }
13293 {}

```

Four column style:

```

13294 \ifcsdef{@glsstyle@altlongragged4col}
13295 {%
13296 \renewglossarystyle{altlongragged4col}{%
13297 \renewenvironment{theglossary}%
13298 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13299 >{\raggedright}p{\glspagelistwidth}}}%
13300 {\end{longtable}}}%
13301 \renewcommand*{\glossaryheader}{}%
13302 \renewcommand*{\glsgroupheading}[1]{}%
13303 \renewcommand{\glossentry}[2]{%
13304 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13305 \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13306 ##2\tabularnewline

```

```

13307 }%
13308 \renewcommand{\subglossentry}[3]{%
13309     &
13310     \glssubentryitem{##2}%
13311     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13312     \glossentrysymbol{##2} & ##3\tabularnewline
13313 }%
13314 \ifglsgnogroupskip
13315     \renewcommand*{\glsgroupskip}{}%
13316 \else
13317     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13318 \fi
13319 }
13320 }
13321 {}

```

## 2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glstrprelocation`.

```

13322 \ifcsdef{@glstyle@super}
13323 {%
13324     \renewglossarystyle{super}{%
13325         \renewenvironment{theglossary}%
13326             {\tablehead{}}\tabletail{}}%
13327         \begin{supertabular}{lp{\glsgdescwidth}}%
13328             {\end{supertabular}}%
13329         \renewcommand*{\glossaryheader}{}%
13330         \renewcommand*{\glsgroupheading}[1]{}%
13331         \renewcommand{\glossentry}[2]{%
13332             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13333             \glossentrydesc{##1}\glspostdescription
13334             \glstrprelocation ##2\tabularnewline
13335         }%
13336         \renewcommand{\subglossentry}[3]{%
13337             &
13338             \glssubentryitem{##2}%
13339             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13340             \glstrprelocation ##3\tabularnewline
13341         }%
13342         \ifglsgnogroupskip
13343             \renewcommand*{\glsgroupskip}{}%
13344         \else
13345             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13346         \fi
13347     }
13348 }
13349 {}

```

Three column style:

```

13350 \ifcsdef{@glsstyle@super3col}
13351 {%
13352   \renewglossarystyle{super3col}{%
13353     \renewenvironment{theglossary}%
13354       {\tablehead{}\tabletail{}}%
13355       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13356       {\end{supertabular}}}%
13357   \renewcommand*{\glossaryheader}{}%
13358   \renewcommand*{\glsgroupheading}[1]{}%
13359   \renewcommand{\glossentry}[2]{%
13360     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13361     \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
13362   }%
13363   \renewcommand{\subglossentry}[3]{%
13364     &
13365     \glssubentryitem{##2}%
13366     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13367     ##3\tabularnewline
13368   }%

13369   \ifglsgroupskip
13370     \renewcommand*{\glsgroupskip}{}%
13371   \else
13372     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13373   \fi
13374 }
13375 }
13376 {}

```

Four column styles:

```

13377 \ifcsdef{@glsstyle@super4col}
13378 {%
13379   \renewglossarystyle{super4col}{%
13380     \renewenvironment{theglossary}%
13381       {\tablehead{}\tabletail{}}%
13382       \begin{supertabular}{llll}}}%
13383       \end{supertabular}}}%
13384   \renewcommand*{\glossaryheader}{}%
13385   \renewcommand*{\glsgroupheading}[1]{}%
13386   \renewcommand{\glossentry}[2]{%
13387     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13388     \glsentrydesc{##1}\glspostdescription &
13389     \glsentrysymbol{##1} & ##2\tabularnewline
13390   }%
13391   \renewcommand{\subglossentry}[3]{%
13392     &
13393     \glssubentryitem{##2}%
13394     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13395     \glsentrysymbol{##2} & ##3\tabularnewline

```

```

13396 }%
13397 \ifglsgroupskip
13398 \renewcommand*{\glsgroupskip}{}%
13399 \else
13400 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13401 \fi
13402 }
13403 }
13404 {}

```

## 2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glxstrprelocation`.

```

13405 \ifcsdef{@glstyle@superragged}
13406 {%
13407 \renewglossarystyle{superragged}{%
13408 \renewenvironment{theglossary}%
13409 {\tablehead{}\tabletail}%
13410 \begin{supertabular}{l>\raggedright}p{\glsdscwidth}}}%
13411 {\end{supertabular}}%
13412 \renewcommand*{\glossaryheader}{}%
13413 \renewcommand*{\glsgroupheading}[1]{}%
13414 \renewcommand{\glossentry}[2]{%
13415 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13416 \glossentrydesc{##1}\glspostdescription\glxstrprelocation ##2%
13417 \tabularnewline
13418 }%
13419 \renewcommand{\subglossentry}[3]{%
13420 &
13421 \glssubentryitem{##2}%
13422 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13423 \glxstrprelocation ##3%
13424 \tabularnewline
13425 }%
13426 \ifglsgroupskip
13427 \renewcommand*{\glsgroupskip}{}%
13428 \else
13429 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13430 \fi
13431 }
13432 }
13433 {}

```

Three column style:

```

13434 \ifcsdef{@glstyle@superragged3col}
13435 {%

```

```

13436 \renewglossarystyle{superragged3col}{%
13437   \renewenvironment{theglossary}%
13438     {\tablehead{}\tabletail{}}%
13439     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
13440       >{\raggedright}p{\glspagelistwidth}}}%
13441     {\end{supertabular}}%
13442   \renewcommand*{\glossaryheader}{}%
13443   \renewcommand*{\glsgroupheading}[1]{}%
13444   \renewcommand{\glossentry}[2]{%
13445     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13446     \glossentrydesc{##1}\glspostdescription &
13447     ##2\tabularnewline
13448   }%
13449   \renewcommand{\subglossentry}[3]{%
13450     &
13451     \glssubentryitem{##2}%
13452     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13453     ##3\tabularnewline
13454   }%

13455   \ifglsgroupskip
13456     \renewcommand*{\glsgroupskip}{}%
13457   \else
13458     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13459   \fi
13460 }
13461 }
13462 {}

```

Four columns:

```

13463 \ifcsdef{@glsstyle@altsuperragged4col}
13464 {%
13465   \renewglossarystyle{altsuperragged4col}{%
13466     \renewenvironment{theglossary}%
13467       {\tablehead{}\tabletail{}}%
13468       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
13469         >{\raggedright}p{\glspagelistwidth}}}%
13470       {\end{supertabular}}%
13471     \renewcommand*{\glossaryheader}{}%
13472     \renewcommand{\glossentry}[2]{%
13473       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13474       \glossentrydesc{##1}\glspostdescription &
13475       \glossentrysymbol{##1} & ##2\tabularnewline
13476     }%
13477     \renewcommand{\subglossentry}[3]{%
13478       &
13479       \glssubentryitem{##2}%
13480       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13481       \glossentrysymbol{##2} & ##3\tabularnewline
13482     }%

```



```

13483 \ifglsnogroupskip
13484 \renewcommand*{\glsgroupskip}{}%
13485 \else
13486 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13487 \fi
13488 }
13489 }
13490 {}

```

## 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`.

```

13491 \ifdef{\@glsstyle@inline}
13492 {%
13493 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13494 \renewcommand*{\glsinlinedescformat}[3]{%
13495 \space#1\glxtrpostdescription}
13496 \renewcommand*{\glsinlinesubdescformat}[3]{%
13497 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13498 }
13499 {}

```

## 2.8 Tree Styles

Redefine both `\glstreenamfmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamfmt` to make it easier to change both at the same time or only change one without affecting the other.

```

13500 \ifdef\glstreenamfmt
13501 {

```

```

\glstreedefaultnamfmt

```

```

13502 \newcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}

```

```

\glstreenamfmt

```

```

13503 \renewcommand{\glstreenamfmt}[1]{\glstreedefaultnamfmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13504 \def\glstreegroupheaderfmt#1{\glstreedefaultnamfmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13505 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13506 }
13507 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13508 \ifdef{\@glsstyle@index}
13509 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13510 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13511 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13512 \renewglossarystyle{index}{%
13513   \renewenvironment{theglossary}%
13514     {\setlength{\parindent}{0pt}%
13515      \setlength{\parskip}{0pt plus 0.3pt}%
13516      \let\item\glstreeitem
13517      \let\subitem\glstreesubitem
13518      \let\subsubitem\glstreesubsubitem
13519    }%
13520   {\par}%
13521   \renewcommand*{\glossaryheader}{}%
13522   \renewcommand*{\glsgroupheading}[1]{}%
13523   \renewcommand*{\glossentry}[2]{%
13524     \item\glssentryitem{##1}%
13525     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13526     \glstreesymbol{##1}%
13527     \glstreedesc{##1}%
13528     \glstreeprelocation ##2%
13529   }%
13530   \renewcommand{\subglossentry}[3]{%
13531     \ifcase##1\relax
13532       \item
13533     \or
13534       \subitem
13535       \glssubentryitem{##2}%
13536     \else
13537       \subsubitem
13538     \fi
13539     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13540     \glstreechildsymbol{##2}%
13541     \glstreechilddesc{##2}%
13542     \glstreechildprelocation ##3%
13543   }%
```

```

13544 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}%
13545 }
13546 }
13547 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13548 \ifdef{\@glsstyle@indexgroup}
13549 {%
13550 \renewglossarystyle{indexgroup}{%
13551 \setglossarystyle{index}%
13552 \renewcommand*{\glsgroupheading}[1]{%
13553 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13554 \nopagebreak\indexspace
13555 \nobreak\@afterheading
13556 }%
13557 }
13558 }
13559 {}

```

Similarly for indexhypergroup.

```

13560 \ifdef{\@glsstyle@indexhypergroup}
13561 {%
13562 \renewglossarystyle{indexhypergroup}{%
13563 \setglossarystyle{index}%
13564 \renewcommand*{\glossaryheader}{%
13565 \item\glstreenavigationfmt{\glsnavigation}%
13566 \nobreak\@afterheading\indexspace}%
13567 \renewcommand*{\glsgroupheading}[1]{%
13568 \item\glstreegroupheaderfmt
13569 {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13570 \nopagebreak\indexspace
13571 \nobreak\@afterheading}%
13572 }%
13573 }
13574 {}

```

Adjust tree style to remove hard coded space before number list.

```

13575 \ifdef{\@glsstyle@tree}
13576 {%
13577 %Provide a command for use with the \glostyle{tree} styles that displays
13578 %the pre-description separator, the
13579 %description and post-description hook.
13580 %\begin{macro}{\glstreedesc}
13581 %\changes{1.31}{2018-05-09}{new}
13582 % \begin{macrocode}
13583 \newcommand{\glstreedesc}[1]{%
13584 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13585 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13586 \newcommand{\glstreesymbol}[1]{%
13587   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13588 }%
```

And for the child entries:

`lstreechilddesc`

```
13589 \newcommand{\glstreechilddesc}[1]{%
13590   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13591 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13592 \newcommand{\glstreechildsymbol}[1]{%
13593   \glstreesymbol{#1}%
13594 }%

13595 \renewglossarystyle{tree}{%
13596   \renewenvironment{theglossary}%
13597     {\setlength{\parindent}{0pt}%
13598     \setlength{\parskip}{0pt plus 0.3pt}}%
13599   {%
13600     \renewcommand*{\glossaryheader}{}%
13601     \renewcommand*{\glsgroupheading}[1]{}%
13602     \renewcommand{\glossentry}[2]{%
13603       \hangindent0pt\relax
13604       \parindent0pt\relax
13605       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13606       \glstreesymbol{##1}%
13607       \glstreedesc{##1}%
13608       \glstreeprelocation##2\par
13609     }%
13610     \renewcommand{\subglossentry}[3]{%
13611       \hangindent##1\glstreeindent\relax
13612       \parindent##1\glstreeindent\relax
13613       \ifnum##1=1\relax
13614         \glssubentryitem{##2}%
13615       \fi
13616       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13617       \glstreechildsymbol{##2}%
13618       \glstreechilddesc{##2}%
13619       \glstreechildprelocation ##3\par
13620     }%
13621     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13622   }%
13623 }
13624 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13625 \ifdef{\@glstyle@treegroup}
```

```

13626 {%
13627   \renewglossarystyle{treegroup}{%
13628     \setglossarystyle{tree}%
13629     \renewcommand{\glsgroupheading}[1]{\par
13630       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13631       \nopagebreak\indexspace\nobreak\@afterheading}%
13632   }
13633 }
13634 {}

```

Similarly for treehypergroup

```

13635 \ifdef{\@glsstyle@treehypergroup}
13636 {%
13637   \renewglossarystyle{treehypergroup}{%
13638     \setglossarystyle{tree}%
13639     \renewcommand*\{\glossaryheader}{%
13640       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13641       \nobreak\@afterheading\indexspace}%
13642     \renewcommand*\{\glsgroupheading}[1]{%
13643       \par\noindent
13644       \glstreegroupheaderfmt
13645       {\glstravigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
13646       \nopagebreak\indexspace\nobreak\@afterheading}%
13647   }
13648 }
13649 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13650 \ifdef{\@glsstyle@treenoname}
13651 {%
13652 %Provide a command for use with the \glsstyle{treenoname} styles that displays
13653 %the pre-description separator, the
13654 %description and post-description hook.
13655 %\begin{macro}{\glstreenonamedesc}
13656 %\changes{1.31}{2018-05-09}{new}
13657 %   \begin{macrocode}
13658   \newcommand{\glstreenonamedesc}[1]{%
13659     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13660   }%

```

Similarly for the symbol.

treenonamesymbol

```

13661   \newcommand{\glstreenonamesymbol}[1]{%
13662     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13663   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13664   \newcommand{\glstreenonamechilddesc}[1]{%
13665     \glossentrydesc{#1}\glspostdescription
13666   }%

```

```

13667 \renewglossarystyle{treenoname}{%
13668   \renewenvironment{theglossary}%
13669     {\setlength{\parindent}{0pt}%
13670      \setlength{\parskip}{0pt plus 0.3pt}}%
13671   {}%
13672   \renewcommand*{\glossaryheader}{}%
13673   \renewcommand*{\glsgroupheading}[1]{}%
13674   \renewcommand{\glossentry}[2]{%
13675     \hangindent0pt\relax
13676     \parindent0pt\relax
13677     \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13678     \glstreenonamesymbol{##1}%
13679     \glstreenonamedesc{##1}%
13680     \glstreeprelocation##2\par
13681   }%
13682   \renewcommand{\subglossentry}[3]{%
13683     \hangindent##1\glstreeindent\relax
13684     \parindent##1\glstreeindent\relax
13685     \ifnum##1=1\relax
13686       \glssubentryitem{##2}%
13687     \fi
13688     \glstarget{##2}{\strut}%
13689     \glstreenonamechilddesc{##2}%
13690     \glstreechildprelocation##3\par
13691   }%
13692   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13693 }
13694 }
13695 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13696 \ifdef{\@glstyle@treenonamegroup}
13697 {%
13698   \renewglossarystyle{treenonamegroup}{%
13699     \setglossarystyle{treenoname}%
13700     \renewcommand{\glsgroupheading}[1]{\par
13701       \noindent\glstreegroupheaderfmt
13702       {\glsggetgrouptitle{##1}}}%
13703     \nopagebreak\indexspace\nobreak\@afterheading
13704   }%
13705 }
13706 }
13707 {}

```

Similarly for `treenonamehypergroup`

```

13708 \ifdef{\@glstyle@treenonamehypergroup}
13709 {%
13710   \renewglossarystyle{treenonamehypergroup}{%
13711     \setglossarystyle{treenoname}%
13712     \renewcommand*{\glossaryheader}{%

```

```

13713      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13714      \nobreak\@afterheading\indexspace}%
13715      \renewcommand*{\glsgroupheading}[1]{%
13716      \par\noindent
13717      \glstreegroupheaderfmt
13718      {\glsnavhypertarget{##1}{\glsgroupgettitle{##1}}}%
13719      \nopagebreak\indexspace\nobreak\@afterheading}%
13720  }
13721 }
13722 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13723 \ifdef{\glstyle@almtree}
13724 {}%

```

Only redefine this if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

13725 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13726   {%
13727     \let\par\glxtrAltTreePar
13728     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13729     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13730   }%
13731 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13732 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13733 \newcommand{\glxtrAltTreePar}{%
13734   \@@par
13735   \glxtrAltTreeSetHangIndent
13736   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13737 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13738 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13739   \glxtralttreeSymbolDescLocation{#2}{#3}%
13740 }

```

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.

```
13741 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13742 \newcommand*\glxstralttreeInit{%
13743   \settowidth{\glxstrtreetopindent}{\glstreenamfmt{\glsggetwidestname\space}}%
13744   \glxstrAltTreeIndent=\parindent
13745 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13746 \newcommand*\gglsetwidest}[2][0]{%
13747   \csgdef{@glswidestname\romannumeral#1}{#2}%
13748 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13749 \newcommand*\eglssetwidest}[2][0]{%
13750   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13751 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13752 \newcommand*\xglsetwidest}[2][0]{%
13753   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13754 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13755 \newcommand*\glupdatewidest}[2][0]{%
13756   \ifcsundef{@glswidestname\romannumeral#1}%
13757   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13758   {%
13759     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13760     \settowidth{\dimen@ii}{#2}%
13761     \ifdim\dimen@ii>\dimen@
13762     \csdef{@glswidestname\romannumeral#1}{#2}%
13763     \fi
13764   }%
13765 }
```

glupdatewidest As above but global definition.

```
13766 \newcommand*\gglupdatewidest}[2][0]{%
13767   \ifcsundef{@glswidestname\romannumeral#1}%
13768   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13769   {%
13770     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13771     \settowidth{\dimen@ii}{#2}%
13772     \ifdim\dimen@ii>\dimen@
13773     \csgdef{@glswidestname\romannumeral#1}{#2}%
13774     \fi
```



```

13775 }%
13776 }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13777 \newcommand*{\eglsupdatewidest}[2][0]{%
13778 \ifcsundef{@glswidestname\romannumeral#1}%
13779 {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13780 {%
13781 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13782 \settowidth{\dimen@ii}{#2}%
13783 \ifdim\dimen@ii>\dimen0
13784 \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13785 \fi
13786 }%
13787 }

```

`glsupdatewidest` As above but global.

```

13788 \newcommand*{\xglsupdatewidest}[2][0]{%
13789 \ifcsundef{@glswidestname\romannumeral#1}%
13790 {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13791 {%
13792 \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13793 \settowidth{\dimen@ii}{#2}%
13794 \ifdim\dimen@ii>\dimen0
13795 \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13796 \fi
13797 }%
13798 }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13799 \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glswidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13800 \newcommand*{\glsgetwidestsubname}[1]{%
13801 \ifcsundef{@glswidestname\romannumeral#1}%
13802 {\@glswidestname}%
13803 {\csuse{@glswidestname\romannumeral#1}}%
13804 }

```

`glsFindWidestTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```

13805 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname

```

`glsFindWidestUsedTopLevelName` Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```

13806 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
13807 \dimen@=0pt\relax

```

```

13808 \gls@tmplen=Opt\relax
13809 \foralllglossaries[#1]{\@gls@type}%
13810 {%
13811 \forglsentries[\@gls@type]{\@glo@label}%
13812 {%
13813 \ifglsused{\@glo@label}%
13814 {%
13815 \ifglshasparent{\@glo@label}%
13816 {}%
13817 {%
13818 \settowidth{\dimen@}%
13819 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13820 \ifdim\dimen@>\gls@tmplen
13821 \gls@tmplen=\dimen@
13822 \eglssetwidest{\glsentryname{\@glo@label}}%
13823 \fi
13824 }%
13825 }%
13826 {}%
13827 }%
13828 }%
13829 }

```

**destUsedAnyName** Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13830 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13831 \dimen@=Opt\relax
13832 \gls@tmplen=Opt\relax
13833 \foralllglossaries[#1]{\@gls@type}%
13834 {%
13835 \forglsentries[\@gls@type]{\@glo@label}%
13836 {%
13837 \ifglsused{\@glo@label}%
13838 {%
13839 \settowidth{\dimen@}%
13840 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13841 \ifdim\dimen@>\gls@tmplen
13842 \gls@tmplen=\dimen@
13843 \eglssetwidest{\glsentryname{\@glo@label}}%
13844 \fi
13845 }%
13846 {}%
13847 }%
13848 }%
13849 }

```

**ndWidestAnyName** Like the above but doesn't check if the entry has been used.

```

13850 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13851 \dimen@=Opt\relax

```

```

13852 \gls@tmplen=0pt\relax
13853 \foralllglossaries[#1]{\@gls@type}%
13854 {%
13855   \forglsentries[\@gls@type]{\@glo@label}%
13856   {%
13857     \settowidth{\dimen@}%
13858     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13859     \ifdim\dimen@>\gls@tmplen
13860       \gls@tmplen=\dimen@
13861       \eglssetwidest{\glsentryname{\@glo@label}}%
13862     \fi
13863   }%
13864 }%
13865 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.  
Any entry that has a great-grandparent is ignored.

```

13866 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13867   \dimen@=0pt\relax
13868   \dimen@i=0pt\relax
13869   \dimen@ii=0pt\relax
13870   \foralllglossaries[#1]{\@gls@type}%
13871   {%
13872     \forglsentries[\@gls@type]{\@glo@label}%
13873     {%
13874       \ifglused{\@glo@label}%
13875       {%
13876         \ifglshasparent{\@glo@label}%
13877         {%
13878           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
13879           \ifglshasparent{\@glo@parent}%
13880           {%
13881             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
13882             \ifglshasparent{\@glo@parent}%
13883             {}%
13884           {%
13885             \settowidth{\gls@tmplen}%
13886             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13887             \ifdim\gls@tmplen>\dimen@ii
13888               \dimen@ii=\gls@tmplen
13889               \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13890             \fi
13891           }%
13892         }%
13893       {%
13894         \settowidth{\gls@tmplen}%
13895         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13896         \ifdim\gls@tmplen>\dimen@i
13897           \dimen@i=\gls@tmplen

```

```

13898         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13899     \fi
13900 }%
13901 }%
13902 {%
13903     \settowidth{\gls@tmplen}%
13904         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13905     \ifdim\gls@tmplen>\dimen@
13906         \dimen@=\gls@tmplen
13907         \eglssetwidest{\glsentryname{\@glo@label}}%
13908     \fi
13909 }%
13910 }%
13911 {}%
13912 }%
13913 }%
13914 }

```

**dWidestLevelTwo** This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13915 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13916     \dimen@=0pt\relax
13917     \dimen@i=0pt\relax
13918     \dimen@ii=0pt\relax
13919     \foralllglossaries[#1]{\@gls@type}%
13920     {%
13921         \forglsentries[\@gls@type]{\@glo@label}%
13922         {%
13923             \ifglshasparent{\@glo@label}%
13924             {%
13925                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13926                 \ifglshasparent{\@glo@parent}%
13927                 {%
13928                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13929                     \ifglshasparent{\@glo@parent}%
13930                     {}%
13931                 }%
13932                 \settowidth{\gls@tmplen}%
13933                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13934                 \ifdim\gls@tmplen>\dimen@ii
13935                     \dimen@ii=\gls@tmplen
13936                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13937                 \fi
13938             }%
13939         }%
13940     }%
13941     \settowidth{\gls@tmplen}%
13942         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13943     \ifdim\gls@tmplen>\dimen@i
13944         \dimen@i=\gls@tmplen

```

```

13945         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13946     \fi
13947 }%
13948 }%
13949 {%
13950     \settowidth{\gls@tmplen}%
13951         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13952     \ifdim\gls@tmplen>\dimen@
13953         \dimen@=\gls@tmplen
13954         \eglssetwidest{\glsentryname{\@glo@label}}%
13955     \fi
13956 }%
13957 }%
13958 }%
13959 }

```

**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.

```

13960 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13961     \dimen@=0pt\relax
13962     \gls@tmplen=0pt\relax
13963     #2=0pt\relax
13964     \foralllglossaries[#1]{\@gls@type}%
13965     {%
13966         \forglsentries[\@gls@type]{\@glo@label}%
13967         {%
13968             \ifglsused{\@glo@label}%
13969             {%
13970                 \settowidth{\dimen@}%
13971                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13972                 \ifdim\dimen@>\gls@tmplen
13973                     \gls@tmplen=\dimen@
13974                     \eglssetwidest{\glsentryname{\@glo@label}}%
13975                 \fi
13976                 \settowidth{\dimen@}%
13977                     {\glsentrysymbol{\@glo@label}}%
13978                 \ifdim\dimen@>#2\relax
13979                     #2=\dimen@
13980                 \fi
13981             }%
13982         }%
13983     }%
13984 }%
13985 }

```

**stAnyNameSymbol** Like the above but doesn't check if the entry has been used.

```

13986 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
13987     \dimen@=0pt\relax
13988     \gls@tmplen=0pt\relax

```

```

13989      #2=Opt\relax
13990      \foralllglossaries[#1]{\@gls@type}%
13991      {%
13992        \forglsentries[\@gls@type]{\@glo@label}%
13993        {%
13994          \settowidth{\dimen@}%
13995            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13996          \ifdim\dimen@>\gls@tmplen
13997            \gls@tmplen=\dimen@
13998            \eglssetwidest{\glsentryname{\@glo@label}}%
13999          \fi
14000          \settowidth{\dimen@}%
14001            {\glsentrysymbol{\@glo@label}}%
14002          \ifdim\dimen@>#2\relax
14003            #2=\dimen@
14004          \fi
14005        }%
14006      }%
14007    }

```

**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.

```

14008    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14009      \dimen@=Opt\relax
14010      \gls@tmplen=Opt\relax
14011      #2=Opt\relax
14012      #3=Opt\relax
14013      \foralllglossaries[#1]{\@gls@type}%
14014      {%
14015        \forglsentries[\@gls@type]{\@glo@label}%
14016        {%
14017          \ifglsused{\@glo@label}%
14018          {%
14019            \settowidth{\dimen@}%
14020              {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14021            \ifdim\dimen@>\gls@tmplen
14022              \gls@tmplen=\dimen@
14023              \eglssetwidest{\glsentryname{\@glo@label}}%
14024            \fi
14025            \settowidth{\dimen@}%
14026              {\glsentrysymbol{\@glo@label}}%
14027            \ifdim\dimen@>#2\relax
14028              #2=\dimen@
14029            \fi
14030            \settowidth{\dimen@}%
14031              {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14032            \ifdim\dimen@>#3\relax

```

```

14033         #3=\dimen@
14034     \fi
14035 }%
14036 {}%
14037 }%
14038 }%
14039 }

```

**eSymbolLocation** Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14040 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14041     \dimen@=0pt\relax
14042     \gls@tmplen=0pt\relax
14043     #2=0pt\relax
14044     #3=0pt\relax
14045     \forallglossaries[#1]{\@gls@type}%
14046     {%
14047         \forglsentries[\@gls@type]{\@glo@label}%
14048         {%
14049             \settowidth{\dimen@}%
14050             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14051             \ifdim\dimen@>\gls@tmplen
14052                 \gls@tmplen=\dimen@
14053                 \eglssetwidest{\glsentryname{\@glo@label}}%
14054             \fi
14055             \settowidth{\dimen@}%
14056             {\glsentrysymbol{\@glo@label}}%
14057             \ifdim\dimen@>#2\relax
14058                 #2=\dimen@
14059             \fi
14060             \settowidth{\dimen@}%
14061             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14062             \ifdim\dimen@>#3\relax
14063                 #3=\dimen@
14064             \fi
14065         }%
14066     }%
14067 }

```

**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.

```

14068 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14069     \dimen@=0pt\relax
14070     \gls@tmplen=0pt\relax
14071     #2=0pt\relax
14072     \forallglossaries[#1]{\@gls@type}%
14073     {%
14074         \forglsentries[\@gls@type]{\@glo@label}%
14075         {%

```

```

14076     \ifglsused{\@glo@label}%
14077     {%
14078         \settowidth{\dimen@}%
14079         {\glstreenamfmt{\glentryname{\@glo@label}}}%
14080         \ifdim\dimen@>\gls@tmplen
14081             \gls@tmplen=\dimen@
14082             \eglssetwidest{\glentryname{\@glo@label}}%
14083         \fi
14084         \settowidth{\dimen@}%
14085         {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14086         \ifdim\dimen@>#2\relax
14087             #2=\dimen@
14088         \fi
14089     }%
14090     {}%
14091 }%
14092 }%
14093 }

```

**AnyNameLocation** Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14094 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14095     \dimen@=0pt\relax
14096     \gls@tmplen=0pt\relax
14097     #2=0pt\relax
14098     \forallglossaries[#1]{\@gls@type}%
14099     {%
14100         \forglsentries[\@gls@type]{\@glo@label}%
14101         {%
14102             \settowidth{\dimen@}%
14103             {\glstreenamfmt{\glentryname{\@glo@label}}}%
14104             \ifdim\dimen@>\gls@tmplen
14105                 \gls@tmplen=\dimen@
14106                 \eglssetwidest{\glentryname{\@glo@label}}%
14107             \fi
14108             \settowidth{\dimen@}%
14109             {\GlsXtrFormatLocationList{\glentrynumberlist{\@glo@label}}}%
14110             \ifdim\dimen@>#2\relax
14111                 #2=\dimen@
14112             \fi
14113         }%
14114     }%
14115 }

```

**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`.

```

14116 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
14117     \glstreeindent=\glsxtrtreetopindent\relax
14118 }

```



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.

```
14119 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14120 \ifcsundef{@glswidestname\romannumeral#1}%
14121 {%
14122 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14123 }%
14124 {%
14125 \settowidth{#3}{\glstreenamefmt{
14126 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14127 }%
14128 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14129 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14130 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14131 \renewglossarystyle{alttree}{%
14132 \renewenvironment{theglossary}%
14133 {%
14134 \glxstralttreeInit
14135 \def\@gls@prevlevel{-1}%
14136 \mbox{}\par}%
14137 {\par}%
14138 \renewcommand*{\glossaryheader}{}%
14139 \renewcommand*{\glsgroupheading}[1]{}%
14140 \renewcommand{\glossentry}[2]{%
14141 \ifnum\@gls@prevlevel=0\relax
14142 \else
14143 \glxstrComputeTreeIndent{##1}%
14144 \fi
14145 \parindent\glstreeindent
14146 \glxstrAltTreeSetHangIndent
14147 \makebox[Opt][r]%
14148 {%
14149 \glstreenamebox{\glstreeindent}%
14150 {%
14151 \glsentryitem{##1}%
14152 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14153 }%
14154 }%
```

```

14155 \glxtralttreeSymbolDescLocation{##1}{##2}%
14156 \def\@gls@prevlevel{0}%
14157 }
14158 \renewcommand{\subglossentry}[3]{%
14159 \ifnum##1=1\relax
14160 \glssubentryitem{##2}%
14161 \fi
14162 \ifnum\@gls@prevlevel=##1\relax
14163 \else
14164 \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14165 \ifnum\@gls@prevlevel<##1\relax
14166 \setlength\glstreeindent\gls@tmplen
14167 \addtolength\glstreeindent\parindent
14168 \parindent\glstreeindent
14169 \else
14170 \ifnum\@gls@prevlevel=0\relax
14171 \glxtrComputeTreeIndent{##2}%
14172 \else
14173 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14174 \fi
14175 \addtolength\parindent{-\glstreeindent}%
14176 \setlength\glstreeindent\parindent
14177 \fi
14178 \fi
14179 \glxtrAltTreeSetSubHangIndent{##1}%
14180 \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14181 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14182 \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14183 \def\@gls@prevlevel{##1}%
14184 }%
14185 \renewcommand*\@gls@groupskip{\ifglsnogroupskip\else\indexspace\fi}%
14186 }
14187 }%
14188 {%
14189 }

```

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.

```

14190 \ifdef{\@glsstyle@alttreegroup}
14191 {%
14192 \renewglossarystyle{alttreegroup}{%
14193 \setglossarystyle{alttree}%
14194 \renewcommand{\gls@groupheading}[1]{\par
14195 \def\@gls@prevlevel{-1}%
14196 \hangindent0pt\relax
14197 \parindent0pt\relax
14198 \glstreegroupheaderfmt{\gls@getgrouptitle{##1}}}%
14199 \nopagebreak\indexspace\nopagebreak
14200 }%
14201 }%

```

```

14202 }%
14203 {%
14204 }

```

Similarly for `alttreehypergroup`.

```

14205 \ifdef{\@glsstyle@alttreehypergroup}
14206 {%
14207   \renewglossarystyle{alttreehypergroup}{%
14208     \setglossarystyle{alttree}%
14209     \renewcommand*{\glossaryheader}{%
14210       \par
14211       \def\@gls@prevlevel{-1}%
14212       \hangindent0pt\relax
14213       \parindent0pt\relax
14214       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14215     }%
14216     \renewcommand*{\glsgroupheading}[1]{%
14217       \par
14218       \def\@gls@prevlevel{-1}%
14219       \hangindent0pt\relax
14220       \parindent0pt\relax
14221       \glstreegroupheaderfmt
14222       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14223       \nopagebreak\indexspace\nopagebreak
14224     }%
14225   }
14226 }%
14227 {%
14228 }

```

## 2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14229 \ifdef{\@glsstyle@mcolindexgroup}
14230 {%
14231   \renewglossarystyle{mcolindexgroup}{%
14232     \setglossarystyle{mcolindex}%
14233     \renewcommand*{\glsgroupheading}[1]{%
14234       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14235       \nopagebreak\indexspace\nobreak\@afterheading
14236     }%
14237   }
14238 }%
14239 {%
14240 }

```

Similarly for `mcolindexhypergroup`.

```

14241 \ifdef{\@glsstyle@mcolindexhypergroup}
14242 {%

```

```

14243 \renewglossarystyle{mcolindexhypergroup}{%
14244   \setglossarystyle{mcolindex}%
14245   \renewcommand*{\glossaryheader}{%
14246     \item\glstreenavigationfmt{\glsnavigation}%
14247     \indexspace
14248   }%
14249   \renewcommand*{\glsgroupheading}[1]{%
14250     \item\glstreegroupheaderfmt
14251       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14252     \nopagebreak\indexspace\nobreak\@afterheading
14253   }%
14254 }
14255 }%
14256 {%
14257 }

```

Similarly for mcolindexspannav.

```

14258 \ifdef{\@glsstyle@mcolindexspannav}
14259 {%
14260   \renewglossarystyle{mcolindexspannav}{%
14261     \setglossarystyle{index}%
14262     \renewenvironment{theglossary}%
14263     {%
14264       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14265       \setlength{\parindent}{0pt}%
14266       \setlength{\parskip}{0pt plus 0.3pt}%
14267       \let\item\glstreeitem}%
14268     {\end{multicols}}}%
14269   \renewcommand*{\glsgroupheading}[1]{%
14270     \item\glstreegroupheaderfmt
14271       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14272     \nopagebreak\indexspace\nobreak\@afterheading
14273   }%
14274 }
14275 }%
14276 {%
14277 }

```

Similarly for mcoltreegroup.

```

14278 \ifdef{\@glsstyle@mcoltreegroup}
14279 {%
14280   \renewglossarystyle{mcoltreegroup}{%
14281     \setglossarystyle{mcoltree}%
14282     \renewcommand{\glsgroupheading}[1]{\par
14283       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14284     \nopagebreak\indexspace\nobreak\@afterheading
14285   }%
14286 }
14287 }%
14288 {%

```

14289 }

Similarly for mcoltreehypergroup.

```
14290 \ifdef{\@glsstyle@mcoltreehypergroup}
14291 {%
14292   \renewglossarystyle{mcoltreehypergroup}{%
14293     \setglossarystyle{mcoltree}%
14294     \renewcommand*{\glossaryheader}{%
14295       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14296     }%
14297     \renewcommand*{\glsgroupheading}[1]{%
14298       \par\noindent
14299       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14300       \nopagebreak\indexspace\nobreak\@afterheading
14301     }%
14302   }
14303 }%
14304 {%
14305 }
```

Similarly for mcoltreespannav.

```
14306 \ifdef{\@glsstyle@mcoltreespannav}
14307 {%
14308   \renewglossarystyle{mcoltreespannav}{%
14309     \setglossarystyle{tree}%
14310     \renewenvironment{theglossary}%
14311     {%
14312       \begin{multicols}{\glsmcols}%
14313       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14314       \setlength{\parindent}{0pt}%
14315       \setlength{\parskip}{0pt plus 0.3pt}%
14316     }%
14317     {\end{multicols}}%
14318     \renewcommand*{\glsgroupheading}[1]{%
14319       \par\noindent
14320       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14321       \nopagebreak\indexspace\nobreak\@afterheading
14322     }%
14323   }
14324 }%
14325 {%
14326 }
```

Similarly for mcoltreenonamegroup.

```
14327 \ifdef{\@glsstyle@mcoltreenonamegroup}
14328 {%
14329   \renewglossarystyle{mcoltreenonamegroup}{%
14330     \setglossarystyle{mcoltreenoname}%
14331     \renewcommand{\glsgroupheading}[1]{\par
14332       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14333     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14334 }%
14335 }
14336 }%
14337 {%
14338 }

```

Similarly for mcoltreenonamehypergroup.

```

14339 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14340 {%
14341   \renewglossarystyle{mcoltreenonamehypergroup}{%
14342     \setglossarystyle{mcoltreenoname}%
14343     \renewcommand*{\glossaryheader}{%
14344       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14345     \renewcommand*{\glsgroupheading}[1]{%
14346       \par\noindent
14347       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14348       \nopagebreak\indexspace\nobreak\@afterheading}%
14349   }
14350 }%
14351 {%
14352 }

```

Similarly for mcoltreenonamespannav.

```

14353 \ifdef{\@glsstyle@mcoltreenonamespannav}
14354 {%
14355   \renewglossarystyle{mcoltreenonamespannav}{%
14356     \setglossarystyle{treenoname}%
14357     \renewenvironment{theglossary}%
14358     {%
14359       \begin{multicols}{\glsmcols}%
14360       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14361       \setlength{\parindent}{0pt}%
14362       \setlength{\parskip}{0pt plus 0.3pt}%
14363     }%
14364     {\end{multicols}}}%
14365   \renewcommand*{\glsgroupheading}[1]{%
14366     \par\noindent
14367     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14368     \nopagebreak\indexspace\nobreak\@afterheading}%
14369   }
14370 }%
14371 {%
14372 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

14373 \ifdef{\@glsstyle@mcolalttree}
14374 {%
14375   \renewglossarystyle{mcolalttree}{%
14376     \setglossarystyle{alttree}%
14377     \renewenvironment{theglossary}%

```

```

14378   {%
14379       \glstralttreeInit
14380       \def\@gls@prevlevel{-1}%
14381       \begin{multicols}{\glsmcols}%
14382   }%
14383   {\par\end{multicols}}%
14384 }
14385 }%
14386 {%
14387 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

14388 \ifdef{\@glsstyle@mcolalmtreegroup}
14389 {%
14390   \renewglossarystyle{mcolalmtreegroup}{%
14391       \setglossarystyle{mcolalmtree}%
14392       \renewcommand{\glsgroupheading}[1]{\par
14393           \def\@gls@prevlevel{-1}%
14394           \hangindent0pt\relax
14395           \parindent0pt\relax
14396           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14397       \nopagebreak\indexspace\nopagebreak
14398   }%
14399   }
14400 }%
14401 {%
14402 }

```

Similarly for mcolalmtreehypergroup.

```

14403 \ifdef{\@glsstyle@mcolalmtreehypergroup}
14404 {%
14405   \renewglossarystyle{mcolalmtreehypergroup}{%
14406       \setglossarystyle{mcolalmtree}%
14407       \renewcommand*\glossaryheader{%
14408           \par
14409           \def\@gls@prevlevel{-1}%
14410           \hangindent0pt\relax
14411           \parindent0pt\relax
14412           \glstreenavigationfmt{\glsnavigation}%
14413           \par\indexspace
14414       }%
14415       \renewcommand*\glsgroupheading[1]{%
14416           \par
14417           \def\@gls@prevlevel{-1}%
14418           \hangindent0pt\relax
14419           \parindent0pt\relax
14420           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14421           \nopagebreak\indexspace\nopagebreak
14422       }%
14423   }

```

```

14424 }%
14425 {%
14426 }

```

Similarly for mcolalttreespannav.

```

14427 \ifdef{\@glsstyle@mcolalttreespannav}
14428 {%
14429   \renewglossarystyle{mcolalttreespannav}{%
14430     \setglossarystyle{almtree}%
14431     \renewenvironment{theglossary}%
14432     {%
14433       \glsextralttreeInit
14434       \def\@gls@prevlevel{-1}%
14435       \begin{multicols}{\glsmcols}%
14436         [\noindent\glstreenavigationfmt{\glsnavigation}]%
14437     }%
14438     {\par\end{multicols}}%
14439     \renewcommand*{\glsgroupheading}[1]{%
14440       \par
14441       \def\@gls@prevlevel{-1}%
14442       \hangindent0pt\relax
14443       \parindent0pt\relax
14444       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14445       \nopagebreak\indexspace\nopagebreak
14446     }%
14447   }
14448 }%
14449 {%
14450 }

```

Reset the default style

```

14451 \ifx\@glossary@default@style\relax
14452 \else
14453   \setglossarystyle{\@glsextr@current@style}
14454 \fi

```



## 3 bookindex style (glossary-bookindex.sty)

### 3.1 Package Initialisation and Options

```
14455 \NeedsTeXFormat{LaTeX2e}
14456 \ProvidesPackage{glossary-bookindex}[2018/05/24 v1.32 (NLCT)]
```

Load required packages.

```
14457 \RequirePackage{multicol}
14458 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14459 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14460 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14461 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14462 \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.

```
14463 \newcommand*{\glstrbookindexprelocation}[1]{%
14464   \glstrifhasfield{location}{#1}%
14465   {,\glstrprelocation}%
14466   {\glstrprelocation}%
14467 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14468 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14469   \glstrbookindexprelocation{#1}%
14470 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14471 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14472 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
14473 \newcommand{\glstrbookindexbetween}[2]{}

indexsubbetween Between two level 1 entries identified by the labels in the arguments.
14474 \newcommand{\glstrbookindexsubbetween}[2]{}

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.
14475 \newcommand{\glstrbookindexsubsubbetween}[2]{}

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.
14476 \newcommand{\glstrbookindexatendgroup}[1]{}

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.
14477 \newcommand{\glstrbookindexsubatendgroup}[1]{}

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.
14478 \newcommand{\glstrbookindexsubsubatendgroup}[1]{}

kindexgroupskip Group separator.
14479 \newcommand{\glstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

Format group title.

dexformatheader Group separator.
14480 \newcommand*{\glstrbookindexformatheader}[1]{%
14481 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14482 }

okindexbookmark Book mark group heading if supported.
14483 \ifdef\pdfbookmark
14484 {%
14485 \newcommand*{\glstrbookindexbookmark}[2]{%
14486 \ifdefstring{\@@glossarysec}{chapter}%
14487 {\pdfbookmark[1]{#1}{#2}}%
14488 {\pdfbookmark[2]{#1}{#2}}%
14489 }
14490 }
14491 {%
14492 \newcommand*{\glstrbookindexbookmark}[2]{}
14493 }

kindexcolspread
14494 \newcommand*{\glstrbookindexcolspread}{}

dexmulticolenv
14495 \newcommand*{\glstrbookindexmulticolenv}{multicols}
```

Define the style.

```

14496 \newglossarystyle{bookindex}{%
14497   \setglossarystyle{index}%
14498   \renewenvironment{theglossary}%
14499   {%
14500     \ifdefempty{glxstrbookindexcolspread}
14501     {%
14502       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14503       {\glxstrbookindexcols}%
14504     }%
14505     {%
14506       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14507       {\glxstrbookindexcols}[\glxstrbookindexcolspread]%
14508     }%
14509     \setlength{\parindent}{0pt}%
14510     \setlength{\parskip}{0pt plus 0.3pt}%
14511     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14512     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14513     \let\@glxstr@bookindex@between\@gobble
14514     \let\@glxstr@bookindex@subbetween\@gobble
14515     \let\@glxstr@bookindex@subsubbetween\@gobble
14516     \let\@glxstr@bookindex@atendgroup\relax
14517     \let\@glxstr@bookindex@subatendgroup\relax
14518     \let\@glxstr@bookindex@subsubatendgroup\relax
14519     \let\@glxstr@bookindex@groupskip\relax
14520   }%
14521   {%

```

Do end group hooks.

```

14522     \@glxstr@bookindex@subsubatendgroup
14523     \@glxstr@bookindex@subatendgroup
14524     \@glxstr@bookindex@atendgroup

```

End multicol environment.

```

14525     \expandafter\end\expandafter{\glxstrbookindexmulticolseenv}%
14526   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

14527   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

14528   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

14529     \@glxstr@bookindex@between{##1}%

```

Update separators.

```

14530     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14531     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14532     \let\@glxstr@bookindex@subbetween\@gobble
14533     \let\@glxstr@bookindex@subsubbetween\@gobble
14534     \edef\@glxstr@bookindex@between{%

```

```

14535     \noexpand\glxstrbookindexbetween{##1}%
14536 }%
14537 \edef\@glxstr@bookindex@atendgroup{%
14538     \noexpand\glxstrbookindexatendgroup{##1}%
14539 }%
14540 \let\@glxstr@bookindex@subatendgroup\relax
14541 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14542 \glstreeitem
14543 \glstryitem{##1}%
14544 \glstarget{##1}{\glxstrbookindexname{##1}}%
14545 \glxstrbookindexprelocation{##1}##2%
14546 }%
14547 \renewcommand{\subglossentry}[3]{%
14548     \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14549     \glstreeitem
14550     \or

```

Level 1.

```

14551     \@glxstr@bookindex@sep
14552     \@glxstr@bookindex@subbetween{##2}%
14553     \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14554     \let\@glxstr@bookindex@subsubbetween\@gobble
14555     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14556     \edef\@glxstr@bookindex@subbetween{%
14557         \noexpand\glxstrbookindexsubbetween{##2}%
14558     }%
14559     \edef\@glxstr@bookindex@atsubendgroup{%
14560         \noexpand\glxstrbookindexatsubendgroup{##1}%
14561     }%

```

Start sub-item.

```

14562     \glstreesubitem
14563     \glssubentryitem{##2}%
14564     \else

```

All other levels.

```

14565     \@glxstr@bookindex@subsep
14566     \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14567     \let\@glxstr@bookindex@subsep\relax
14568     \edef\@glxstr@bookindex@subsubbetween{%
14569         \noexpand\glxstrbookindexsubsubbetween{##2}%
14570     }%
14571     \edef\@glxstr@bookindex@atsubsubendgroup{%
14572         \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14573     }%

```

Start sub-sub-item.

```
14574 \glstreesubsubitem
14575 \fi
```

Format entry.

```
14576 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14577 \glxtrbookindexsubprelocation{##2}##3%
14578 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14579 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14580 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14581 \@glxtr@bookindex@subsubatendgroup
14582 \@glxtr@bookindex@subatendgroup
14583 \@glxtr@bookindex@atendgroup
14584 \@glxtr@bookindex@groupskip
```

Update separators.

```
14585 \let\@glxtr@bookindex@groupskip\glxtrbookindexgroupskip
14586 \let\@glxtr@bookindex@between\@gobble
14587 \let\@glxtr@bookindex@atendgroup\relax
14588 \let\@glxtr@bookindex@subatendgroup\relax
14589 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14590 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14591 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14592 \glxtrbookindexformatheader{\thisgrptitle}%
14593 \nopagebreak\indexspace\nopagebreak\@afterheading
14594 }%
14595 }
```

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.

```
14596 \newcommand{\glxtrbookindexthepage}{%
14597 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14598 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14599 \newcommand*\glxtrbookindexmarkentry}[1]{%
14600   \protected@write\@auxout
14601   {\let\glxtrbookindexthepage\relax}%
14602   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14603 }
```

etbookindexmark

```
14604 \newcommand*\glxtr@setbookindexmark}[2]{%
14605   \ifcsundef\glxtr@idxfirstmark@#1}%
14606   {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
14607   {}%
14608   \csgdef\glxtr@idxlastmark@#1}{#2}%
14609 }
```

indexfirstmarkfmt

```
14610 \newcommand*\glxtrbookindexfirstmarkfmt}[1]{%
14611   \glxtr@entryname{#1}%
14612 }
```

bookindexfirstmark

```
14613 \newcommand*\glxtrbookindexfirstmark}{%
14614   \letcs\glxtr@label\glxtr@idxfirstmark@\glxtrbookindexthepage}%
14615   \ifdef\glxtr@label
14616   {\glxtrbookindexfirstmarkfmt\glxtr@label}}%
14617   {}%
14618 }
```

indexlastmarkfmt

```
14619 \newcommand*\glxtrbookindexlastmarkfmt}[1]{%
14620   \glxtr@entryname{#1}%
14621 }
```

bookindexlastmark

```
14622 \newcommand*\glxtrbookindexlastmark}{%
14623   \letcs\glxtr@label\glxtr@idxlastmark@\glxtrbookindexthepage}%
14624   \ifdef\glxtr@label
14625   {\glxtrbookindexlastmarkfmt\glxtr@label}}%
14626   {}%
14627 }
```

# Glossary

**First use** The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

**First use flag** A conditional that determines whether or not the entry has been used according to the rules of **first use**.

**First use text** The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.

# Change History

0.1 (2015-11-22)	
General: Initial experimental release . . . . .	5
0.2 (2015-11-30)	
\Glsfmtshort: new . . . . .	323
\glsfmtshort: new . . . . .	322
\Glsfmtshortpl: new . . . . .	323
\glsfmtshortpl: new . . . . .	322
short: switched inline full form to short (long) . . . . .	225
0.3 (2015-12-02)	
\@ACRlong: added redefinition . . . . .	78
\@ACRlongpl: added redefinition . . . . .	79
\@ACRshort: added redefinition . . . . .	76
\@ACRshortpl: added redefinition . . . . .	77
\@Acrlong: added redefinition . . . . .	77
\@Acrlongpl: added redefinition . . . . .	78
\@Acrshort: added redefinition . . . . .	75
\@Acrshortpl: added redefinition . . . . .	76
\@GLSdesc: added redefinition . . . . .	71
\@GLSdescplural@: added redefinition . . . . .	72
\@GLSfirst@: added redefinition . . . . .	69
\@GLSfirstplural@: added redefinition . . . . .	70
\@GLSname@: added redefinition . . . . .	71
\@GLSplural@: added redefinition . . . . .	70
\@GLSSymbol@: added redefinition . . . . .	72
\@GLSSymbolplural@: added redefinition . . . . .	73
\@GLStext@: added redefinition . . . . .	68
\@GLSuseri@: added redefinition . . . . .	73
\@GLSuserii@: added redefinition . . . . .	74
\@GLSuseriii@: added redefinition . . . . .	74
\@GLSuseriv@: added redefinition . . . . .	74
\@GLSuserv@: added redefinition . . . . .	75
\@GLSuservi@: added redefinition . . . . .	75
\@Glsdesc@: added redefinition . . . . .	71
\@Glsdescplural@: added redefinition . . . . .	72
\@Glsfirst@: added redefinition . . . . .	69
\@Glsfirstplural@: added redefinition . . . . .	70
\@Glsname@: added redefinition . . . . .	71
\@Glsplural@: added redefinition . . . . .	69
\@Glsymbol@: added redefinition . . . . .	72
\@Glsymbolplural@: added redefinition . . . . .	73
\@Glsxtr@defaultnoglossarywarning: new . . . . .	131
\@Glsxtr@field@linkdefs: new . . . . .	67
\@Glsxtr@insertdots: new . . . . .	193
\@print@glossary: added redefinition . . . . .	128
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont . . . . .	199
\glsaccessdesc: new . . . . .	157
\glsaccessdescplural: new . . . . .	157
\glsaccessfirst: new . . . . .	154
\glsaccessfirstplural: new . . . . .	155
\glsaccesslong: new . . . . .	159
\glsaccesslong: new . . . . .	159
\glsaccessname: new . . . . .	152
\glsaccessplural: new . . . . .	154
\glsaccessshort: new . . . . .	158
\glsaccessshort: new . . . . .	158
\glsaccessshortpl: new . . . . .	158



\glsaccesssshortpl: new .....	158	\cGLSpl: new .....	105
\glsaccessssymbol: new .....	155	\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 .....	104
\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 .....	207	\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 .....	167
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	167
\glsxtrshortpl .....	322	\glsxtremsuffix: new .....	261
\glsxtrifemptyglossary: new .....	27	\GlsXtrEnableEntryCounting: new ..	99
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new .....	102
argument .....	171	\glsxtrscfont: new .....	233
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new .....	233
argument .....	170	\glsxtrsmfont: new .....	247
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new .....	247
default type to \acronymtype ....	113	short-em: new .....	268
\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 .....	104	short-em-long-desc: new .....	266

short-em-postfootnote: new .....	281	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new .....	243	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 .....	279
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 .....	153	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 .....	157	postfootnote: switch off regular	
\Glsaccessfirst: new .....	154	attribute if set .....	223
\Glsaccessfirstplural: new .....	155	0.5.2 (2015-12-08)	
\Glsaccessname: new .....	153	\@GLSdesc@: added accessibility support	71
\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 .....	70
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 .....	72
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute .....	316	accessibility support .....	73
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	68
attribute .....	316	\@GLSdesc@: added accessibility support	71
\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	152
\@Glsplural@: added accessibility support .....	69	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here .....	60
\@Glsymbol@: added accessibility support .....	72	\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	59
\@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 .....	69	\@glsxtr@autoindex@at: new .....	182
\@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 .....	180
\Glossentrysymbol: added .....	185	\glsxtrpostnamehook: new .....	177
\glossentrysymbol: added .....	185	\if@glsxtr@format@override: new .	180
\GLSaccessdesc: new .....	157, 165	\ProvidesGlossariesExtraLang: new	328
\GLSaccessdescplural: new ...	157, 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 .....	159	\@GLSxtr@p@acrshort@: new .....	90
\glsaccesslongpl: new .....	159	\@GLSxtr@p@acrshortpl@: new .....	91
\GLSaccessname: new .....	153, 163	\@GLSxtr@p@long@: new .....	90
\GLSaccessplural: new .....	154, 163	\@GLSxtr@p@longpl@: new .....	90
\GLSaccessshort: new .....	158, 165	\@GLSxtr@p@plural@: new .....	89
\GLSaccessshortpl: new .....	159, 165	\@GLSxtr@p@short@: new .....	89
\GLSaccesssymbol: new .....	156, 164	\@GLSxtr@p@shortpl@: new .....	90
\GLSaccesssymbolplural: new .	156, 164	\@GLSxtr@p@text@: new .....	89
\GLSaccessstext: new .....	153, 163	\@GlsXtrEnableOnTheFly: new .....	53
		\@Glsxtr: new .....	54
		\@Glsxtr@p@acrlong@: new .....	91

\@Glsxtr@p@acrlongpl@: new	91	\glstdonohyperlink: added	87
\@Glsxtr@p@acrshort@: new	90	\glsenableentryunitcount: new	108
\@Glsxtr@p@acrshortpl@: new	91	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	90	entry's existence	167
\@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	53
\@alt@glshyp@opt: new	85	\glxtrcat: new	53
\@glscalt@hyp@opt: new	84	\glxtrdowrglossaryhook: new	84
\@glscalt@hyp@opt@char: new	85	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	85	new	111
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	52
new	106	\Glsxtrpl: new	54
\@glsc@local@increment@currunitcount:		\glxtrpl: new	54
new	107	\glxtrpostlocalreset: new	99
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	98
new	82	\glxtrpostreset: new	98
\@glxtr: new	53	\glxtrpostunset: new	97
\@glxtr@addunitcounter: new	106	\glxtrprotectlinks: new	88
\@glxtr@currunitcount: new	107	\GlsXtrSetAltModifier: new	85
\@glxtr@ifunitcounter: new	106	\GlsXtrSetDefaultGlsOpts: new	83
\@glxtr@p@acrlong@: new	91	\glxtrstarflywarn: new	53
\@glxtr@p@acrlongpl@: new	91	\GlsXtrWarning: new	55
\@glxtr@p@acrshort@: new	90	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new	90	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	121
\@glxtr@p@shortpl@: new	89	\@glxtr@idx@entrynumberlist: new	123
\@glxtr@p@text@: new	88	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new	107	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	114
'see' to field key map	44	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	105	\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

\Glsfmtfull: new .....	327	\@GLSdesc@: set abbreviation and regular format .....	71
\glsfmtfull: new .....	327	\@GLSdescplural@: set abbreviation and regular format .....	72
\Glsfmtfullpl: new .....	328	\@GLSfirst@: set abbreviation format ..	69
\glsfmtfullpl: new .....	328	\@GLSfirstplural@: set abbreviation and regular format .....	70
\Glsfmtlong: new .....	326	\@GLSname@: set abbreviation and regular format .....	71
\glsfmtlong: new .....	326	\@GLSplural@: set abbreviation and regular format .....	70
\Glsfmtlongpl: new .....	327	\@GLSsymbol@: set regular format .....	72
\glsfmtlongpl: new .....	327	\@GLSsymbolplural@: set regular format	73
\Glsxtrheadfull: new .....	321	\@GLStext@: set abbreviation and regular format .....	68
\glsxtrheadfull: new .....	320	\@GLSuseri@: set regular format .....	73
\Glsxtrheadfullpl: new .....	321	\@GLSuserii@: set regular format .....	74
\glsxtrheadfullpl: new .....	321	\@GLSuseriii@: set regular format .....	74
\Glsxtrheadlong: new .....	319	\@GLSuseriv@: set regular format .....	74
\glsxtrheadlong: new .....	318	\@GLSuseriv@: set regular format .....	75
\Glsxtrheadlongpl: new .....	320	\@GLSdesc@: set abbreviation and regular format .....	71
\glsxtrheadlongpl: new .....	319	\@GLSdescplural@: set abbreviation and regular format .....	72
\Glsxtrtitlefull: new .....	321	\@GLSfirst@: set abbreviation and regular format .....	69
\glsxtrtitlefull: new .....	320	\@GLSfirstplural@: set abbreviation and regular format .....	70
\Glsxtrtitlefullpl: new .....	322	\@GLSname@: set abbreviation and regular format .....	71
\glsxtrtitlefullpl: new .....	321	\@GLSplural@: set abbreviation and regular format .....	69
\Glsxtrtitlelong: new .....	320	\@GLSsymbol@: set regular format .....	72
\glsxtrtitlelong: new .....	319	\@GLSsymbolplural@: set regular format	73
\Glsxtrtitlelongpl: new .....	320	\@GLStext@: set abbreviation and regular format .....	68
\glsxtrtitlelongpl: new .....	319	\@GLSuseri@: set regular format .....	73
\ifglsxtrinsertinside: new .....	216	\@GLSuserii@: set regular format .....	73
postfootnote: added redef of \glsxtrsetupfulldefs .....	223	\@GLSuseriii@: set regular format .....	74
stylemods: new .....	21	\@GLSuseriv@: set regular format .....	74
1.03 (2016-04-27)		\@GLSuseriv@: set regular format .....	75
\@GLSfirstplural@: bug fix: misspelt cs name .....	70	\@GLSdesc@: set abbreviation and regular format .....	71
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	70	\@GLSdescplural@: set abbreviation and regular format .....	72
\@GLSfirstplural@: bug fix: misspelt cs name .....	70	\@GLSfirst@: set abbreviation and regular format .....	69
\@GLsplural@: fixed bug \@GLsplural@ should be redefined not \@GLsplural	69	\@GLSfirstplural@: set abbreviation and regular format .....	70
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	69	\@GLSname@: set abbreviation and regular format .....	71
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl ..	319	\@GLSplural@: set abbreviation and regular format .....	69
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl	313	\@GLSsymbol@: set regular format .....	72
1.04 (2015-04-30)		\@GLSsymbolplural@: set regular format	73
short-em-footnote: renamed from “footnote-em” .....	279	\@GLStext@: set abbreviation and regular format .....	68
1.04 (2016-05-02)		\@GLSuseri@: set regular format .....	73
\@@glsxtrpostloctag: new .....	59	\@GLSuserii@: set regular format .....	73
		\@GLSuseriii@: set regular format .....	74
		\@GLSuseriv@: set regular format .....	74
		\@GLSuseriv@: set regular format .....	74
		\@GLSuseriv@: set regular format .....	75
		\@gls@preglossaryhook: added check for entry's existence .....	187
		\@glsdesc@: set abbreviation and regular format .....	71
		\@glsdescplural@: set abbreviation and regular format .....	72

\@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 .....	69	\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 .....	59	short-em-postfootnote: renamed from “postfootnote-em” .....	281
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand .....	123	short-footnote: new .....	223
\@glsxtr@pagetag: new .....	59	short-long-user: new .....	290
\@glsxtr@pagetag: new .....	59	short-long-user-desc: new .....	291
\@glsxtr@preloctag: new .....	59	short-nolong: new .....	226
\@glsxtr@postloctag: new .....	59	short-nolong-desc: new .....	228
\@glsxtr@preloctag: new .....	58, 59	short-postfootnote: new .....	225
\glossentrydesc: added glossdescfont attribute check .....	172	short-sc-footnote: renamed from “footnote-sc” .....	243
\Glossentryname: added glossnamefont attribute check .....	176	short-sc-nolong: new .....	238
\glossentryname: added glossnamefont attribute check .....	174	short-sc-nolong-desc: new .....	239
moved post name hook inside condition .....	176	short-sc-postfootnote: renamed from “postfootnote-sc” .....	245
\glsabbrvmfont: new .....	261	short-sm-footnote: renamed from “footnote-sm” .....	258
\glsabbrvuserfont: new .....	283	short-sm-nolong: new .....	252
\glsfirstabbrvmfont: new .....	261	short-sm-nolong-desc: new .....	254
\glsfirstabbrvuserfont: new .....	283	short-sm-postfootnote: renamed from “postfootnote-sm” .....	259
\glsfirstlongemfont: new .....	261	\letabbreviationstyle: new .....	215
\glsfirstlonguserfont: new .....	284	\newabbreviationstyle: bug fix: corrected test for existence .....	214
\glsifnotregularcategory: new ...	169	long-em-noshort-em: new .....	274
\glslongdefaultfont: new .....	199	long-em-noshort-em-desc: new ....	277
\glslongemfont: new .....	262	long-em-short-em: new .....	263
\glslongfont: new .....	199	long-em-short-em-desc: new .....	264
\glslonguserfont: new .....	283	long-noshort: new .....	232
\glsxtrassignfieldfont: new .....	68	long-noshort-desc: new .....	231
\GlsXtrEnablePreLocationTag: new ..	58	long-noshort-em: renamed from “long-em” .....	272
\glsxtrfirstscfont: new .....	233	long-noshort-em-desc: renamed from “long-desc-em” .....	276
\glsxtrfirstsmfont: new .....	247	long-noshort-sc: renamed from “long-sc” .....	240
\glsxtrlongshortdescsort: new ...	218	long-noshort-sc-desc: renamed from “long-desc-sc” .....	242
\glsxtrpostnamehook: added category check .....	178	long-noshort-sm: renamed from “long-sm” .....	254

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 .... 214	\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)
\glsFindWidestAnyName: new ..... 386	@@glstrp: new ..... 91
\glsFindWidestAnyNameLocation:	\GLSfirst@: added check for
new ..... 392	nohyperfirst attribute ..... 69
\glsFindWidestAnyNameSymbol: new 389	\GLSfirstplural@: added check for
\glsFindWidestAnyNameSymbolLocation:	nohyperfirst attribute ..... 70
new ..... 391	\GLSxtrp: new ..... 92
\glsFindWidestLevelTwo: new ..... 388	\GLsfirst@: added check for
\glsFindWidestUsedAnyName: new .. 386	nohyperfirst attribute ..... 69
\glsFindWidestUsedAnyNameLocation:	\GLsfirstplural@: added check for
new ..... 391	nohyperfirst attribute ..... 70
\glsFindWidestUsedAnyNameSymbol:	\GLsxtrp: new ..... 92
new ..... 389	\gls@preglossaryhook: added
\glsFindWidestUsedAnyNameSymbolLocation:	\glossxtrsetpopts ..... 187
new ..... 390	\GLsfirst@: added check for
\glsFindWidestUsedLevelTwo: new . 387	nohyperfirst attribute ..... 69
\glsFindWidestUsedTopLevelName:	\GLsfirstplural@: added check for
new ..... 385	nohyperfirst attribute ..... 70
\glsfirstlongfootnotefont: new .. 221	\GLsxtrinmark: new ..... 310
\glsgetwidestname: new ..... 385	\GLsxtrnotinmark: new ..... 311
\glsgetwidestsubname: new ..... 385	\GLsxtrp: new ..... 92
\glslongfootnotefont: new ..... 221	\GLsxtrp@opt: new ..... 91
\glstrAltTreeIndent: new ..... 383	\glossxtrsetpopts: new ..... 91
\glstrAltTreeInit: new ..... 384	\glsps: new ..... 94
\glstrAltTreePar: new ..... 383	\glspt: new ..... 94
\glstrAltTreeSetHangIndent: new 393	\glstr@entry@p: new ..... 93
\glstrAltTreeSetSubHangIndent:	\glstrabbrvfootnote: new ..... 221
new ..... 393	\glstrchecknohyperfirst: new .... 68
\glstrAltTreeSubSymbolDescLocation:	\glstrfieldtitlecasecs: new .... 172
new ..... 383	\glstrifinmark: new ..... 310
\glstrAltTreeSymbolDescLocation:	\GLSxtrp: new ..... 95
new ..... 383	\GLsxtrp: new ..... 94
\glstrComputeTreeIndent: new ... 392	\GLsxtrp: new ..... 93
\glstrComputeTreeSubIndent: new 393	\glstrsetpopts: new ..... 91
\glstrtreetopindent: new ..... 384	short-long-desc: added text key .... 220
short-em-long: fixed incorrect font used	fixed misspelling of \glsabbrvfont in
by long form ..... 266	plural key ..... 220
\xglsetwidest: new ..... 384	long-short-desc: added missing text
1.06 (2016-06-18)	key ..... 218
\GLsdoifexistsorwarn: new ..... 15	fixed misspelling of \glsabbrvfont . 218
\GLsxtr@docdefval: new ..... 15	footnote: changed first forms to use
\GLsxtr@usesee: new ..... 45	\glsfirstlongfootnotefont ... 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 ...	61	\@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	40
\@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 .....	138	\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 .....	135	definition .....	187
\glsxtr@resource: new .....	133	\@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 .....	125	\glxtrresourcefile: added catcode change for @ .....	133
\glissettoctitle: added patch .....	41	\glxtrsetaliasnoindex: new .....	82
\glxtr@counterrecord: new .....	136	\GlsXtrSetField: new .....	34
\glxtr@langtag: new .....	134	\glxtrsetfieldifexists: new .....	33
\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 .....	41	short-postlong-user: new .....	287
\glxtr@s@newignoredglossary: Added check for existence .....	40	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 .....	48	long-postshort-user-desc: new ...	287
\glxtrcopytoglossary: new .....	42	postdot: new .....	16
\glxtrdeffield: new .....	33	\pretoglossarypreamble: new .....	38
\glxtrdisplayendloc: new .....	126	\print@noop@unsrtglossaryunit: new .....	141
\glxtrdisplayendlochook: new ...	126	\print@op@unsrtglossaryunit: new	140
\glxtrdisplaysingleloc: new ....	126	\printunsrtglossary: added starred form .....	138
\glxtrdisplaystartloc: new ....	126	\printunsrtglossaryhandler: new .	140
\glxtrredefield: new .....	33	\printunsrtglossaryunit: new ....	12
\glxtrentryfmt: new .....	30	\printunsrtglossaryunitsetup: new	140
\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 .....	31	\glxtrsetaliasnoindex: switched to \providecommand .....	82
\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 .....	66	\@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 .....	124	\@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 ....	127	\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	\glxtrseealso: new .....	49
short-em-long-em: fixed spelling of		\glxtrseelist: new .....	46
\glsabbrvfont .....	267	\glxtruseseealso: new .....	45
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 .....	287	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 .....	122
footnote: fixed spelling of		\@glxtrifhyphenstart: new .....	292
\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> .....	306	<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> .....	283	<code>new</code> .....	300
<code>\glxtr@newabbreviation:added</code>		<code>long-hyphen-short-hyphen:new</code> ...	293
<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@recordlocref instead of</code>	
<code>\glxtrposthyphenlong:new</code> .....	303	<code>\@glxtr@locref</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@locref</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> ...	301	<code>format test</code> .....	181
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink:new</code> .....	86
<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 .....	149
1.21 (2017-11-03)		\@rGLSpl@: new .....	150
\@@glxtr@record: added check for default options .....	9	\@rGls: new .....	148
\@@glxtrwrglossmark: new .....	24	\@rGls@: new .....	148
\@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 .....	147
redefined altlistgroup to discourage breaks after group headings .....	368	\@rgls@: new .....	147
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 .....	141	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	146	redefined mcoltreehypergroup to discourage breaks after group headings .....	397
\@glxtrglossentry: new .....	136		
\@glxtrnewgls: new .....	143		

redefined mcoltreenonamegroup to discourage breaks after group headings .....	397	\glxtr@org@dohyperlink: new .....	85
redefined mcoltreenonamehypergroup to discourage breaks after group headings .....	398	\glxtr@setbookindexmark: new ...	406
redefined mcoltreenonamespannav to discourage breaks after group headings .....	398	\glxtrbookindexatendgroup: new ..	402
redefined mcoltreespannav to discourage breaks after group headings .....	397	\glxtrbookindexbetween: new ....	402
redefined treenonamegroup to discourage breaks after group headings .....	382	\glxtrbookindexbookmark: new ...	402
redefined treenonamehypergroup to discourage breaks after group headings .....	382	\glxtrbookindexcols: new .....	401
debug: new .....	24	\glxtrbookindexcolspread: new ..	402
\gglsetwidest: new .....	384	\glxtrbookindexfirstmark: new ..	406
\gl:disablehyper: added check for existence .....	87	\glxtrbookindexfirstmarkfmt: new	406
changed to use \def rather than \let ..	87	\glxtrbookindexformatheader: new	402
\gl:dohypertarget: redefined treegroup to discourage breaks after group headings .....	380	\glxtrbookindexgroupskip: new ..	402
redefined treehypergroup to discourage breaks after group headings .....	381	\glxtrbookindexlastmark: new ...	406
\gl:enablehyper: changed to use \def rather than \let .....	87	\glxtrbookindexlastmarkfmt: new	406
\Glsfmtname: new .....	323	\glxtrbookindexmarkentry: new ..	406
\gl:fmtname: new .....	323	\glxtrbookindexname: new .....	401
\gl:shex: new .....	329	\glxtrbookindexparentchildsep: new .....	401
\gl:slstchildpostlocation: new ..	367	\glxtrbookindexparentschildsep: new .....	401
\gl:slstchildprelocation: new ...	367	\glxtrbookindexprelocation: new	401
\gl:slstprelocation: new .....	367	\glxtrbookindexsubatendgroup: new .....	402
\gl:snahyperlink: patched .....	85	\glxtrbookindexsubbetween: new ..	402
\gl:seeitemformat: new .....	45	\glxtrbookindexsubname: new ....	401
\gl:sshowtarget: new .....	25	\glxtrbookindexsubprelocation: new .....	401
\gl:streechildprelocation: new ...	378	\glxtrbookindexsubsubatendgroup: new .....	402
\gl:streeprelocation: new .....	378	\glxtrbookindexsubsubbetween: new .....	402
\gl:triggerrecordformat: new ....	147	\glxtrbookindexthepage: new ....	405
\gl:useabbrvfont: new .....	210	\glxtrdetoklocation: new .....	146
\gl:uselongfont: new .....	210	\glxtr:enablerecordcount: new ...	146
\glxtr@do@alsoindex@wrglossary: new .....	8	\glxtrglossentry: new .....	136
\glxtr@org@do@wrglossary: new ..	27	\glxtrgroupfield: new .....	141
		\Glsxrheadname: new .....	315
		\glxtrheadname: new .....	314
		\GlsXtrIfFieldEqStr: new .....	34
		\glxtriflabelinlist: new .....	140
		\glxtrifrecordtrigger: new .....	146
		\glxtrindexseealso: added check that the entry exists .....	46
		\glxtrinithyperoutside: new .....	64
		\GlsXtrLocationRecordCount: new ..	145
		\glxtrnewgls: new .....	143, 144
		\glxtrnewGLSlike: new .....	144
		\glxtrnewglslike: new .....	144
		\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 ....	137
\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 .....	119
uppercasing .....	237	1.23 (2017-11-12)	
short-sm: corrected first letter		\@@glstrfmt: added check for indexing	30
uppercasing .....	252	added grouping .....	29
\ifglstr@hyperoutside: new .....	63	new .....	29
all: new .....	365	\@glstr@nopostpunc@postdesc: new	120
nolong-short: new .....	228	\@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 .....	239	\@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 .....	139, 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 .....	147	\glstrifcustomdiscardperiod: new	188
\rGLSformat: new .....	150	\GlsXtrIfFieldUndef: new .....	33
\rGlsformat: new .....	150	\glstrrestorepostpunc: new ....	120
\rglsformat: new .....	150	\s@glstrfmt: new .....	29
\rGLSpl: new .....	149	\s@glstrfmt: new .....	29
\rGlspl: new .....	149	\xglupdatewidest: new .....	385
\rglspl: new .....	148	1.24 (2017-11-14)	
\rGLSplformat: new .....	150	\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	\glstrbookindexmulticolseenv: new	402
1.22 (2017-11-08)		1.25 (2017-11-24)	
\@glstr@nopostpunc: new .....	119	\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	286	\glxtrGeneralLatinVIIrules: new	344
\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 . . . . .	218	\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 . . . . .	151	\glxtrLatinH: new . . . . .	345
\GlsXtrLinkCounterName: new . . . .	152	\glxtrLatinI: new . . . . .	346
\GlsXtrLinkCounterValue: new . . . .	151	\glxtrLatinInsularG: new . . . . .	348
\GlsXtrTheLinkCounter: new . . . . .	152	\glxtrLatinK: new . . . . .	346
1.27 (2018-02-26)		\glxtrLatinL: new . . . . .	346
\@glxtr@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 . . . . .	332	\glxtrLatinOslash: new . . . . .	348
\Digamma: new . . . . .	332	\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 . . . . .	347
new . . . . .	336	\glxtrLatinX: new . . . . .	346
\glxtrcombiningdiacriticIrules:		\glxtrMathGreekIIrules: new . . . .	354
new . . . . .	336	\glxtrMathGreekIrules: new . . . .	353
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new . . . .	358
new . . . . .	335	\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 . . . .	358
\glxtrcurrencyrules: new . . . . .	339	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new . . . . .	362	new . . . . .	350
\glxtrfractionrules: new . . . . .	362	\glxtrMathItalicGreekIrules: new	349
\glxtrGeneralLatinIIrules: new	341	\glxtrMathItalicIota: new . . . . .	359

<code>\glxtrMathItalicKappa:new</code> . . . . .	359	<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> . . . . .	357
<code>new</code> . . . . .	351	<code>\glxtrUpTheta:new</code> . . . . .	356
<code>\glxtrMathItalicMu:new</code> . . . . .	360	<code>\glxtrUpUpsilon:new</code> . . . . .	358
<code>\glxtrMathItalicNabla:new</code> . . . . .	361	<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> . . . . .	361	<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> . . . . .	360	<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> . . . . .	333
<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> . . . . .	348	<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> . . . . .	333
<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> . . . . .	355	. . . . .	363
<code>\glxtrUpEta:new</code> . . . . .	356	<code>\glxtrredeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code> . . . . .	355	<code>\protected@csedef</code> . . . . .	33
<code>\glxtrUpIota:new</code> . . . . .	356	<code>\glxtrlocalsetgrouptitle</code> : changed	
<code>\glxtrUpKappa:new</code> . . . . .	356	<code>\csedef\protected@csedef</code> . . . . .	124
<code>\glxtrUpLambda:new</code> . . . . .	356	<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>\@glx@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>\@glxrecordloc</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 .....	206
\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	160
\@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 .....	132
\@glxstr@glossadd@postkeys: new ..	10	\@glxstrbuffer@nodup@unset: new ..	97
\@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 .....	120
\@glxstr@local@textformat: new ...	63	\glsaddeach: new .....	67
\@glxstr@unset: new .....	97	\glsapturedgroup: new .....	330
\@glxstrbuffer@unset: new .....	97	\glsdefpostdesc: new .....	188
\glsadd: added \glsaddpostsetkeys	67	\glsdefpostlink: new .....	189
added \glsaddpresetkeys .....	67	\glsdefpostname: new .....	178
\glsaddpostsetkeys: new .....	66	\glsdohypertarget: bug fix: ensure that new version is picked up .....	121
\glsaddpresetkeys: new .....	66	\glslistdesc: new .....	367
\glsuserdescription: new .....	284	\glslocalreseteach: new .....	99
\glxtrabbreviationfont: new .....	60	\glslocalunseteach: new .....	99
\GlsXtrDualBackLink: new .....	330	\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		\GlsXtrIfXpFieldEqXpStr:new .....	35
redefinition .....	378	\glxtrpostlinkAddSymbolDescOnFirstUse:	
\glstreenonamechilddesc:new ....	381	new .....	190
\glstreenonamesymbol:new .....	381	\GlsXtrRecordWarning:new .....	131
\glstreesymbol:new .....	380	\glxtrRevertTocMarks:new .....	310
\glxtr@newabbreviation:added		\GlsXtrStandaloneGlossaryType:	
\ExtraCustomAbbreviationFields		new .....	137
.....	195	\GlsXtrStandaloneSubEntryItem:	
\GlsXtrForUnsetBufferedList:new .	98	new .....	137
\GlsXtrIfFieldCmpNum:new .....	33	\s@GlsXtrStartUnsetBuffering:new	97
\GlsXtrIfFieldEqNum:new .....	32	1.32 (2018-05-24)	
\GlsXtrIfFieldEqXpStr:new .....	35	\GlsXtrForeignText:new .....	35
\GlsXtrIfFieldNonZero:new .....	32	\GlsXtrForeignTextField:new .....	37
\GlsXtrIfHasNonZeroChildCount:		\GlsXtrUnknownDialectWarning:new	37
new .....	330		

# 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, 109
\@cGLSpl@	101, 109
\@ccGLs@	101, 109
\@ccGLspl@	101, 109
\@ccgls@	101, 109
\@ccglspl@	101, 109
\@do@wrglossary	8, 10, 116
\@do@wrglossary	12–14, 27, 67, 83
\@glo@assign@sortkey	121
\@glo@list	6
\@glo@type	138
\@glossarysec	402
\@gls@expand@field	29
\@glslocalreset	99
\@glslocalunset	98
\@glsreset	98
\@glsunset	97
\@glsxtr@autoindex@escspch	183, 184
\@glsxtr@checkspch	181, 182, 184
\@glsxtr@disabledflycommand	55, 56
\@glsxtr@org@postdescription	119, 120
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 136
\@glsxtrfmt	29
\@glsxtrp	92, 93
\@glsxtrp@postloctag	58
\@glsxtrp@preloctag	58
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 47, 51, 116
\@newglossaryentry@defcounters	100
\@newglossaryentry@defunitcounters	108
\@par	383
\@ACRlong	88
\@ACRlongpl	88
\@ACRshort	88
\@ACRshortpl	88
\@Acrlong	88
\@Acrlongpl	88
\@Acrshort	88
\@Acrshortpl	88
\@GLS@	88, 104, 149
\@GLSdesc@	72
\@GLSpl@	88, 104, 105, 150
\@GLSplural@	89
\@GLSSymbol@	73
\@GLSxtr@	89
\@GLSxtr@full	200
\@GLSxtr@fullpl	202
\@GLSxtr@p@acrlong@	88
\@GLSxtr@p@acrlongpl@	88
\@GLSxtr@p@acrshort@	88
\@GLSxtr@p@acrshortpl@	88
\@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
\@GLSxtrp	96
\@GLSxtrshort	88, 204
\@GLSxtrshortpl	88, 208
\@Gls@	88, 103, 104, 148
\@Gls@acentryname	112
\@Gls@entry@field	80, 94, 95, 179
\@Gls@entryname	112
\@GlsXtrEnableOnTheFly	52, 53
\@GlsXtrStartUnsetBuffering	97
\@GlsXtrStopUnsetBuffering	98

\@Glspl@	88, 103, 104, 149	\@end@glstr@addunused	50
\@Glsplural@	89	\@end@glstr@gettype	118, 121
\@Glstext@	89	\@end@glstr@usesee	45
\@Glsxtr	54, 55	\@end@glstrifhyphenstart	292, 293
\@Glsxtr@full	200	\@endfortrue	32, 178, 213
\@Glsxtr@fullpl	201	\@firstofone	68, 139, 172, 173, 180, 186
\@Glsxtr@p@acrlong@	88	\@firstofthree	62, 67, 75–78, 84, 85, 199, 201, 203, 205, 207, 208
\@Glsxtr@p@acrlongpl@	88	\@firstoftwo	69, 70, 72, 73, 76–79, 81, 85, 114, 178, 191, 192, 200–202, 207–210, 310, 311
\@Glsxtr@p@acrshort@	88	\@for	6, 22, 32, 50, 67, 99, 100, 111, 115, 118, 125, 139, 146, 152, 171, 178, 185
\@Glsxtr@p@acrshortpl@	88	\@glo@alias	47, 48
\@Glsxtr@p@long@	88	\@glo@assign@sortkey	118
\@Glsxtr@p@longpl@	88	\@glo@autosee	25
\@Glsxtr@p@plural@	88	\@glo@autoseehook	48
\@Glsxtr@p@short@	88	\@glo@category	106
\@Glsxtr@p@shortpl@	88	\@glo@check@sortallowed	118
\@Glsxtr@p@text@	88	\@glo@counterprefix	10, 11, 127
\@Glsxtrlong	88, 205	\@glo@countunit	106
\@Glsxtrlongpl	88, 209	\@glo@default@sorttype	118
\@Glsxtrp	95	\@glo@desc	39
\@Glsxtrpl	55	\@glo@descplural	39
\@Glsxtrshort	88, 203	\@glo@group	13
\@Glsxtrshortpl	88, 207	\@glo@label	12, 13, 28, 44, 47–49, 79, 87, 386–392
\@acrlong	88	\@glo@location	12
\@acrlongpl	88	\@glo@loclist	12
\@acrshort	88	\@glo@name	181
\@acrshortpl	88	\@glo@no@assign@sortkey	121
\@addtoreset	151	\@glo@parent	387, 388
\@afterheading	368, 369, 379, 381–383, 395–398, 405	\@glo@see	44–46, 48–50
\@alt@glshyp@opt	85	\@glo@seealso	47, 48
\@auxout	11, 12, 51, 59, 101, 102, 110, 115, 116, 128, 133–136, 406	\@glo@sort	181
\@bibgls@restoreat	133	\@glo@sorttype	118, 125
\@cGLS	104	\@glo@text	62
\@cGLS@	101, 104, 109	\@glo@thislettergrp	142
\@cGLSpl	105	\@glo@thisvalue	283
\@cGLSpl@	101, 105, 109	\@glo@tmp	28, 46, 79
\@cGls@	101, 109	\@glo@type	49, 85, 112, 115, 118, 119, 121, 124, 125, 128, 131, 132, 138, 139
\@cGlspl@	101, 109	\@glo@types	169, 170, 385–392
\@cgl@	101, 109	\@glossary@default@style	56, 118, 400
\@cglspl@	101, 109	\@glossarystyle	118, 119
\@disable@onlypremakeg	115	\@gls@	88, 103, 104, 148
\@do@auxoutstuff	128, 129	\@gls@@link	62
\@do@gl@getcounterprefix	10, 11	\@gls@ReturnAfterFi	127
\@do@gl@see	48, 49	\@gls@actualchar	182
\@do@newglossaryentry	112, 196, 197		
\@do@seeglossary	13, 14, 25, 51, 115		
\@do@wrglossary	66, 147		
\@empty	67, 75–79, 120, 127, 182, 199–210		

\@gls@adjustmode .....	67	\@gls@long .....	195
\@gls@alt@hyp@opt .....	85	\@gls@longpl .....	193, 195, 196
\@gls@alt@hyp@opt@char .....	85	\@gls@map .....	178
\@gls@alt@hyp@opt@keys .....	85	\@gls@nameaccess .....	160, 162
\@gls@automake .....	118	\@gls@nohyperlist .....	40, 42
\@gls@between .....	124, 125	\@gls@noidx@do .....	125
\@gls@checkedmidx .....	181, 182, 184	\@gls@noidx@getgrouptitle .....	139
\@gls@checkmidxchars .....	47, 181	\@gls@noidx@nosanitizesort .....	117
\@gls@codepage .....	128	\@gls@noidx@sanitizesort .....	117
\@gls@counter .....	9, 11, 23, 64, 67, 83, 147	\@gls@noidx@loclist@finalsep .....	122
\@gls@currentlettergroup ...	125, 139, 142	\@gls@noidx@loclist@prev .....	122
\@gls@declareoption .....	5	\@gls@noidx@loclist@sep .....	122
\@gls@default@longpl .....	195, 196	\@gls@noref@warn .....	116, 125
\@gls@doautomake .....	118, 135	\@gls@org@glsnoidx@displayloc ..	122, 123
\@gls@doautomake@err .....	135	\@gls@org@gls@seeformat .....	122, 123
\@gls@enablesavenonumberlist .....	50	\@gls@preglossaryhook .....	119, 186
\@gls@encapchar .....	182	\@gls@prevlevel .....	393–395, 399, 400
\@gls@entry@count .....	101, 102	\@gls@quotechar .....	181
\@gls@entry@field .....		\@gls@reference .....	51, 115, 116
... 28, 33, 48, 79, 93, 95, 96, 100, 137, 138		\@gls@restreat .....	51
\@gls@entry@unitcount .....	109, 110	\@gls@saveentrycounter 13, 14, 27, 65, 67, 147	
\@gls@field@font .....	68–75	\@gls@see@noindex .....	26, 133
\@gls@field@link .....	68–75, 80, 81	\@gls@setdefault@glslink@opts .....	
\@gls@firstaccess .....	160, 162	... 9, 30, 65, 83	
\@gls@getcounterprefix .....	10, 11	\@gls@setsort .....	65, 67
\@gls@getgrouptitle .....	124, 139	\@gls@setupsort@none .....	14
\@gls@grptitle .....	85, 125	\@gls@short .....	195, 196
\@gls@hyp@opt .....		\@gls@shortaccess .....	160–162
80, 81, 85, 104, 105, 144, 147–149, 199–209		\@gls@shortaccesspl .....	160, 161
\@gls@hyp@opt@cs .....	84, 85	\@gls@shortpl .....	193, 196
\@gls@ifaccessattribute@set .....	161	\@gls@sort .....	142
\@gls@ifinlist .....	140	\@gls@textaccess .....	160, 162
\@gls@increment@currcount .....	101	\@gls@thislabel .....	67, 99
\@gls@increment@currunitcount .....	109	\@gls@thisval .....	178
\@gls@initaccesskeys .....	195	\@gls@tmp .....	35, 125
\@gls@keymap .. 12, 13, 28, 45, 47, 79, 134, 178		\@gls@tmpb .....	184
\@gls@label .... 8, 9, 11, 51, 84, 116, 136, 213		\@gls@type .....	116, 118, 213, 386–392
\@gls@levelchar .....	182	\@gls@write@entrycounts .....	101
\@gls@link .....	30, 61, 62, 75–79, 200–210	\@gls@write@entryunitcounts .....	109
\@gls@link@checkfirsthyper .....	62, 114	\@gls@write@entryunitcounts@do ...	110
\@gls@link@label .....	64, 147	\@gls@xref .....	12, 47
\@gls@link@nocheckfirsthyper .....		\@gls@abbrv@current@abbreviation 195, 210	
61, 75–79, 199–210		\@gls@acronymlists .....	112
\@gls@link@opts .....	64	\@gls@doifexistsorwarn ... 15, 174–177, 179	
\@gls@list .....	124, 125	\@gls@entry .....	102, 110
\@gls@local@increment@currcount ...	101	\@gls@link .....	66, 85, 87
\@gls@local@increment@currunitcount 109		\@gls@localreset .....	99
\@gls@location .....	142, 143	\@gls@localunset .....	98, 99
\@gls@loclist .....	122, 123, 142, 143	\@gls@nextpages .....	119

<code>\@glsnonextpages</code> .....	119	<code>\@glxtr@accessdisplay</code> .....	178–180
<code>\@glsnnumberformat</code> .....		<code>\@glxtr@activate@initialtagging</code> ..	
.....	9, 11, 64, 67, 83, 147, 178, 181	.....	186, 187
<code>\@glspartner</code> .....	115	<code>\@glxtr@addunitcounter</code> .....	106
<code>\@glsp1@</code> .....	88, 103, 104, 148	<code>\@glxtr@addunused</code> .....	50
<code>\@glsplural@</code> .....	89	<code>\@glxtr@addunusedxrefs</code> .....	49, 50
<code>\@glspunc@token</code> .....	191	<code>\@glxtr@attrval</code> ....	65, 172–177, 179, 181
<code>\@glsrecordlocref</code> .....	10, 11	<code>\@glxtr@autoindex@cat</code> .....	181–183
<code>\@glsshowtarget</code> .....	87	<code>\@glxtr@autoindex@doextra@esc</code> ....	181
<code>\@glssstyle@altlist</code> .....	367	<code>\@glxtr@autoindex@encap</code> .....	181–183
<code>\@glssstyle@altlistgroup</code> .....	368	<code>\@glxtr@autoindex@esc</code> .....	181–184
<code>\@glssstyle@altlisthypergroup</code> .....	369	<code>\@glxtr@autoindex@escat</code> .....	182, 183
<code>\@glssstyle@alttree</code> .....	383	<code>\@glxtr@autoindex@escencap</code> ...	182, 183
<code>\@glssstyle@alttreegroup</code> .....	394	<code>\@glxtr@autoindex@esclevel</code> ...	182, 183
<code>\@glssstyle@alttreehypergroup</code> .....	395	<code>\@glxtr@autoindex@escquote</code> ...	182, 183
<code>\@glssstyle@index</code> .....	378	<code>\@glxtr@autoindex@level</code> .....	182, 183
<code>\@glssstyle@indexgroup</code> .....	379	<code>\@glxtr@autoindex@setname</code> .....	181
<code>\@glssstyle@indexhypergroup</code> .....	379	<code>\@glxtr@autoindexcrossrefs</code> 14, 15, 44, 48	
<code>\@glssstyle@inline</code> .....	377	<code>\@glxtr@autoseeindexfalse</code> .....	14
<code>\@glssstyle@list</code> .....	367	<code>\@glxtr@autoseeindextrue</code> .....	16
<code>\@glssstyle@listdotted</code> .....	366	<code>\@glxtr@bookindex@atendgroup</code> .	403–405
<code>\@glssstyle@listgroup</code> .....	368	<code>\@glxtr@bookindex@atsubendgroup</code> ..	404
<code>\@glssstyle@listhypergroup</code> .....	368	<code>\@glxtr@bookindex@atsubsubendgroup</code>	404
<code>\@glssstyle@mcolalttree</code> .....	398	<code>\@glxtr@bookindex@between</code> ....	403, 405
<code>\@glssstyle@mcolalttreegroup</code> .....	399	<code>\@glxtr@bookindex@sep</code> .....	403, 404
<code>\@glssstyle@mcolalttreehypergroup</code> ..	399	<code>\@glxtr@bookindex@subatendgroup</code> ..	
<code>\@glssstyle@mcolalttreesspannav</code> ....	400	.....	403–405
<code>\@glssstyle@mcolindexgroup</code> .....	395	<code>\@glxtr@bookindex@subbetween</code> .	403, 404
<code>\@glssstyle@mcolindexhypergroup</code> ....	395	<code>\@glxtr@bookindex@subsep</code> ....	403, 404
<code>\@glssstyle@mcolindexspannav</code> .....	396	<code>\@glxtr@bookindex@subsubatendgroup</code>	
<code>\@glssstyle@mcoltreegroup</code> .....	396	.....	403–405
<code>\@glssstyle@mcoltreehypergroup</code> .....	397	<code>\@glxtr@bookindex@subsubbetween</code> ..	
<code>\@glssstyle@mcoltreenamegroup</code> ....	397	.....	403, 404
<code>\@glssstyle@mcoltreenamehypergroup</code>	398	<code>\@glxtr@bookindexgroupskip</code> ...	403, 405
<code>\@glssstyle@mcoltreenamepannav</code> ..	398	<code>\@glxtr@cat</code> .....	100, 111, 146, 185, 186
<code>\@glssstyle@mcoltreesspannav</code> .....	397	<code>\@glxtr@checkgroup</code> .....	139
<code>\@glssstyle@tree</code> .....	379	<code>\@glxtr@counterrecordhook</code> .....	11
<code>\@glssstyle@treegroup</code> .....	380	<code>\@glxtr@csname</code> .....	107, 109
<code>\@glssstyle@treehypergroup</code> .....	381	<code>\@glxtr@current@style</code> .....	56, 400
<code>\@glssstyle@treename</code> .....	381	<code>\@glxtr@currentunitcount</code> ....	107, 109
<code>\@glssstyle@treenamegroup</code> .....	382	<code>\@glxtr@currunitcount</code> .....	108, 110
<code>\@glssstyle@treenamehypergroup</code> ...	382	<code>\@glxtr@debugnr</code> .....	24
<code>\@glstarget</code> .....	87, 120, 121	<code>\@glxtr@debugval</code> .....	24
<code>\@glstext@</code> .....	89	<code>\@glxtr@declareoption</code> ...	5, 16, 18, 21, 23
<code>\@glsunset</code> .....	97, 98	<code>\@glxtr@defaultnoglossarywarning</code> ..	21
<code>\@glswidestname</code> .....	385, 393	<code>\@glxtr@defaultnumberformat</code> .....	
<code>\@glxtr</code> .....	53, 55	.....	7, 9, 64, 67, 83, 178, 181
<code>\@glxtr@@do@wrglossary</code> .....	116	<code>\@glxtr@defpostpunc</code> .....	16, 17, 25
<code>\@glxtr@abbreviationsdef</code> .....	18, 26		

<code>\@glsxtr@deprecated@abbrstyle</code> .....	<code>\@glsxtr@glossnamefont</code> . 174–177, 179, 180
..... 242, 243, 245,	<code>\@glsxtr@gobbleto@endescpch</code> ..... 184
247, 256, 258, 259, 261, 274, 277, 281, 283	<code>\@glsxtr@groupheading</code> ..... 139, 142
<code>\@glsxtr@dialect</code> ..... 35–37	<code>\@glsxtr@idx@displaynumberlist</code> .... 117
<code>\@glsxtr@disabledflycommand</code> ..... 55	<code>\@glsxtr@idx@entrynumberlist</code> ..... 117
<code>\@glsxtr@display@loc</code> ..... 125	<code>\@glsxtr@ifcsstart</code> ..... 53
<code>\@glsxtr@do@wrindex</code> ..... 84	<code>\@glsxtr@ifpunctoken</code> ..... 191
<code>\@glsxtr@do@gl:disablehyperinlist</code> .. 82	<code>\@glsxtr@ifunitcounter</code> ..... 106
<code>\@glsxtr@do@inc@linkcount</code> ..... 152	<code>\@glsxtr@insert@dots</code> ..... 193
<code>\@glsxtr@do@record@wrglossary</code> .... 8, 14	<code>\@glsxtr@insert@dots@next</code> ..... 193, 194
<code>\@glsxtr@do@redef@for@gl:sentries</code> .... 7	<code>\@glsxtr@insertdots</code> ..... 161, 195
<code>\@glsxtr@do@style</code> ..... 22, 329	<code>\@glsxtr@label</code> ..... 32, 50, 152, 171
<code>\@glsxtr@do@titlecaps@warn</code> .....	<code>\@glsxtr@loadstyles</code> ..... 365, 366
..... 173, 174, 176, 179, 186	<code>\@glsxtr@local@textformat</code> ..... 64–66
<code>\@glsxtr@doabbreviationsdef</code> ..... 18	<code>\@glsxtr@locale</code> ..... 36, 37
<code>\@glsxtr@doaccsupp</code> ..... 21, 24	<code>\@glsxtr@longnewglossaryentry</code> ..... 39
<code>\@glsxtr@docdefsetting</code> ..... 15	<code>\@glsxtr@mark@wordseps</code> ..... 194
<code>\@glsxtr@docdefval</code> ..... 15, 50, 52	<code>\@glsxtr@mark@wordseps@next</code> ..... 194
<code>\@glsxtr@docounterrecord</code> ..... 11	<code>\@glsxtr@markwordseps</code> ..... 195, 196
<code>\@glsxtr@doglossary</code> ..... 139	<code>\@glsxtr@mixed@assign@sortkey</code> .... 118
<code>\@glsxtr@doiflabelinlist</code> ..... 140	<code>\@glsxtr@noidx@displaynumberlist</code> .. 117
<code>\@glsxtr@dolocktag</code> ..... 58, 59	<code>\@glsxtr@noidx@do</code> ..... 141
<code>\@glsxtr@dorecord</code> ..... 8, 10	<code>\@glsxtr@noidx@entrynumberlist</code> .... 117
<code>\@glsxtr@dorecordnodefer</code> ..... 8, 10	<code>\@glsxtr@noidx@numberlistloop</code> .... 117
<code>\@glsxtr@dosee@alsoindex@glossary</code> .. 14	<code>\@glsxtr@nomissingglstextnr</code> ..... 21
<code>\@glsxtr@doseeglossary</code> ..... 13, 25	<code>\@glsxtr@nomissingglstextval</code> ..... 21
<code>\@glsxtr@dstylewarn</code> ..... 213	<code>\@glsxtr@noop@recordcounter</code> ..... 11, 13
<code>\@glsxtr@enabletagging</code> ..... 185	<code>\@glsxtr@nopostpunc</code> ..... 119
<code>\@glsxtr@end@</code> ..... 53	<code>\@glsxtr@nopostpunc@postdesc</code> ..... 120
<code>\@glsxtr@endescspch</code> ..... 182–184	<code>\@glsxtr@notfoundinlist</code> ..... 192
<code>\@glsxtr@entrycount@org@localreset</code> 101	<code>\@glsxtr@op@recordcounter</code> ..... 14
<code>\@glsxtr@entrycount@org@localunset</code> 101	<code>\@glsxtr@optlist</code> ..... 55
<code>\@glsxtr@entrycount@org@reset</code> ..... 101	<code>\@glsxtr@org@@starttoc</code> ..... 310
<code>\@glsxtr@entrycount@org@unset</code> ..... 101	<code>\@glsxtr@org@GLS@</code> ..... 62
<code>\@glsxtr@entryunitcount@org@localreset</code>	<code>\@glsxtr@org@GLSpl@</code> ..... 62
..... 109	<code>\@glsxtr@org@Gls@</code> ..... 61
<code>\@glsxtr@entryunitcount@org@localunset</code>	<code>\@glsxtr@org@Glspl@</code> ..... 61, 62
..... 109	<code>\@glsxtr@org@Glsxtrtitlefirst</code> . 311, 312
<code>\@glsxtr@entryunitcount@org@reset</code> . 109	<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>
<code>\@glsxtr@entryunitcount@org@unset</code> . 109	..... 311, 312
<code>\@glsxtr@err@undefaction</code> ..... 7, 13	<code>\@glsxtr@org@Glsxtrtitlefull</code> .. 311, 313
<code>\@glsxtr@field@linkdefs</code> ..... 61	<code>\@glsxtr@org@Glsxtrtitlefullpl</code> 311, 313
<code>\@glsxtr@format@overridefalse</code> ..... 180	<code>\@glsxtr@org@Glsxtrtitlelong</code> .. 311, 312
<code>\@glsxtr@format@overridetrue</code> ..... 180	<code>\@glsxtr@org@Glsxtrtitlelongpl</code> 311, 312
<code>\@glsxtr@foundinlist</code> ..... 192	<code>\@glsxtr@org@Glsxtrtitlename</code> .. 311, 312
<code>\@glsxtr@full</code> ..... 199	<code>\@glsxtr@org@Glsxtrtitleplural</code> 311, 312
<code>\@glsxtr@fullpl</code> ..... 201	<code>\@glsxtr@org@Glsxtrtitleshort</code> . 311, 312
<code>\@glsxtr@gettype</code> ..... 118	<code>\@glsxtr@org@Glsxtrtitleshortpl</code> 311, 312
<code>\@glsxtr@glossdescfont</code> ..... 172–174	<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> ..	35, 36	<code>\@glsxtr@printglossnr</code> .....	120
<code>\@glsxtr@org@delimN</code> .....	58, 59	<code>\@glsxtr@printglossopts</code> .....	55, 118, 120
<code>\@glsxtr@org@delimR</code> .....	58, 59	<code>\@glsxtr@printglossval</code> .....	120
<code>\@glsxtr@org@doseeglossary</code> .....	25, 116	<code>\@glsxtr@printunsrtglossaryskipentry</code> .....	139, 140
<code>\@glsxtr@org@gloautosee</code> .....	26	<code>\@glsxtr@provide@addstoragekey</code> .....	29
<code>\@glsxtr@org@glolinkprefix</code> .....	64, 66	<code>\@glsxtr@provide@storagekey</code> .....	28
<code>\@glsxtr@org@gls@</code> .....	61	<code>\@glsxtr@record</code> .....	13, 14, 61, 62, 67, 199, 201–210
<code>\@glsxtr@org@glsdohypertarget</code> .....	121	<code>\@glsxtr@record@noglossarywarning</code> ..	14
<code>\@glsxtr@org@glsignore</code> .....	58, 59	<code>\@glsxtr@record@setting</code> .....	8, 10, 13, 46, 51, 52, 114, 115
<code>\@glsxtr@org@glspl@</code> .....	61	<code>\@glsxtr@record@setting@alsoindex</code> .	8, 10, 46, 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> .....	114
<code>\@glsxtr@org@glstrtrtitlefull</code> ..	311, 312	<code>\@glsxtr@recordsee</code> .....	14, 25, 46
<code>\@glsxtr@org@glstrtrtitlefullpl</code> ..	311, 313	<code>\@glsxtr@redef@forglsentries</code> .....	7, 26
<code>\@glsxtr@org@glstrtrtitlelong</code> ..	311, 312	<code>\@glsxtr@redefstyles</code> .....	21, 22, 329
<code>\@glsxtr@org@glstrtrtitlelongpl</code> ..	311, 312	<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> ...	114, 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> .....	65, 66
<code>\@glsxtr@org@theHvalue</code> .....	8, 9	<code>\@glsxtr@theHvalue</code> .....	8–10, 63–67, 147
<code>\@glsxtr@org@unset@buffer</code> .....	97, 98	<code>\@glsxtr@thevalue</code> .....	8–10, 63–67, 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
<code>\@glsxtr@p@acrlong@</code> .....	88	<code>\@glsxtr@type</code> .....	171
<code>\@glsxtr@p@acrlongpl@</code> .....	88	<code>\@glsxtr@unitcountlist</code> .....	106
<code>\@glsxtr@p@acrshort@</code> .....	88	<code>\@glsxtr@unset</code> .....	97, 98
<code>\@glsxtr@p@acrshortpl@</code> .....	88	<code>\@glsxtr@unset@buffer</code> .....	97, 98
<code>\@glsxtr@p@long@</code> .....	88	<code>\@glsxtr@unsrt@getgrouptitle</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		



\@glxstr@wrglossary@locationhyperlink	\@one	102, 111, 144
..... 24	\@newglossaryentry@defcounters	100, 108
\@glxstr@wrglossnr	\@newglossaryentryposthook	.....
\@glxstr@wrglossval	..... 12, 13, 28, 47, 79	
\@glxstrbuffer@nodup@unset	\@newglossaryentryprehook	.....
\@glxstrbuffer@unset	..... 12, 13, 28, 39, 47, 79	
\@glxstrdialecthook	\@nil	127, 142
\@glxstrdocdeffalse	\@nnil	182, 184, 191–194
\@glxstrentryfmt	\@no@glxstrindexaliased	83
\@glxstrfmt	\@no@makeglossaries	132
\@glxstrglossentry	\@nocounterr	152
\@glxstrglossentryother	\@nopostdesc	119
\@glxstrhypernameprefix	\@onelevel@sanitize	12, 47, 55, 124, 141
\@glxstrifhasfield	\@onlypreamble	.....
\@glxstrifhyphenstart	.. 56, 58, 110, 133, 136, 152, 180, 183–185	
\@glxstrindexaliased	\@org@glossaryentrynumbers	119
\@glxstrindexcrossrefsfalse	\@org@newglossaryentryprehook	39
\@glxstrindexcrossrefstrue	\@print@unsrt@glossary	138
\@glxstrinmark	\@printgloss@setsort	118, 119
\@glxstrlong	\@printglossary	55, 138
\@glxstrlongpl	\@printunsrt@glossary@handler	139
\@glxstrnewgls	\@printunsrtglossary	138
\@glxstrnewgls@inner	\@rGLS	149
\@glxstrnewgls@innercsname	\@rGLS@	149
\@glxstrnotinmark	\@rGLSpl	149
\@glxstrp	\@rGLSpl@	149
\@glxstrp@opt	\@rGls	148
\@glxstrpl	\@rGls@	148
\@glxstrpostloctag	\@rGlspl	149
\@glxstrpreloctag	\@rGlspl@	149
\@glxstrsetaliasnoindex	\@rgls	147
\@glxstrshort	\@rgls@	147
\@glxstrshortpl	\@rglspl	148
\@glxstrundeftag	\@rglspl@	148
\@glxstrwrglossmark	\@sGlsXtrEnableOnTheFly	52
\@gobble .. 7, 13, 16, 68, 140, 141, 193, 403–405	\@secondofthree	.....
\@gobbletwo	68–70, 76–80, 200, 202, 203, 205, 207, 209	
\@ifnextchar	\@secondoftwo	62, 67,
\@ifpackageloaded	71–79, 81, 87, 114, 120, 178, 192, 199–	
152, 172, 174, 176, 178, 180, 329, 333, 364	201, 203–210, 223, 246, 260, 282, 311, 312	
\@ifstar 28, 29, 32, 39–41, 52, 84, 97, 98, 138, 185	\@sglsxtr@provide@storagekey	28
\@ifundefined	\@starttoc	310
\@ignored@glossaries	\@thirdofthree	68–71,
\@input	76–79, 81, 201, 202, 204, 206, 208, 210, 311	
\@input@	\@thirdoftwo	71–75
\@istfilename	\@this@key	178
\@makeglossary	\@tracklang@lang	36, 37
\@mfu@domakefirstuc	\@warn@nomakeglossaries	129
\@mfu@nocaplist	\@xdy@main@language	128

\@xdycrossrefhook	46	\ACLP	19
\@xdylanguage	128	\Aclp	19
\@xdylocationclassorder	46	\aclp	19
\\	127	\ACP	19
		\Acp	19
\_	52, 130, 131	\acp	18
		\ACRfullfmt	112
		\Acrfullfmt	112
		\acrfullfmt	112
		\ACRfullplfmt	113
		\Acrfullplfmt	113
		\acrfullplfmt	113
		\acronymentry	112
		\acronymfont	75–79, 90, 91, 113, 114
		\acronymname	17
		\acronymsort	112
		\acronymtype	17, 112, 113
		\acrpluralsuffix	112, 134
		\ACS	19
		\Acs	19
		\acs	18
		\ACSP	19
		\Acsp	19
		\acsp	19
		\actualchar	184
		\addtolength	394
		\advance	102, 111, 133, 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, 44–49, 79, 84, 100, 108, 139, 141, 180, 191, 193, 194, 365
		\arabic	23, 405
		\AS	18
		\As	18
		\as	18
		\ASP	18
		\Asp	18

<code>\asp</code> .....	18	<code>\cGLSformat</code> .....	103
<code>\AtBeginDocument</code> .....	24, 27, 56, 57, 135	<code>\cGlsformat</code> .....	103
<code>\AtEndDocument</code> .....	49, 101, 109, 128	<code>\cglformat</code> .....	102, 104
<b>B</b>			
<code>babel</code> package .....	181, 182, 191	<code>\cGLSpl</code> .....	18, 19, 99, 111
<code>\begin</code> .....	125, 130,	<code>\cGlspl</code> .....	18, 19, 99, 111
	139, 367, 369–376, 379, 381, 396–400, 403	<code>\cglspl</code> .....	18, 99, 111
<code>\begingroup</code> ....	8, 9, 29, 30, 83, 136, 138, 151	<code>\cGlsplformat</code> .....	104
<code>\bgroup</code> .....	39, 119	<code>\cGlsplformat</code> .....	103
<code>bib2gls</code> .....	23, 30, 141, 146, 147, 329–331	<code>\cglsplformat</code> .....	103, 105
<b>C</b>			
<code>\c@wrglossary</code> .....	23	<code>\changes</code> .....	315, 379, 381
<code>\catcode</code> .....	51, 133	<code>\char</code> .....	123
category attributes:		<code>\columnwidth</code> .....	57
<code>accessinsertdots</code> .....	161	<code>\count@</code> .....	102, 110, 111
<code>apospplural</code> .....	195	<code>\csappto</code> .....	38
<code>discardperiod</code> .....	190	<code>\csdef</code> .....	28, 33, 34,
<code>entrycount</code> .....	96, 99–101, 111		38, 79–81, 101, 106, 107, 109, 166, 178,
<code>firstshortaccess</code> .....	162		188, 189, 213–215, 223, 245, 260, 281,
<code>firstuc</code> .....	176		285, 287–289, 299, 301, 304, 306, 366, 384
<code>glossdesc</code> .....	172	<code>\cseappto</code> .....	43
<code>glossdescfont</code> .....	172	<code>\csedef</code> .....	107
<code>glossname</code> .....	174	<code>\csgdef</code> .....	34,
<code>glossnamefont</code> .....	174, 176		40–42, 51, 58, 101, 107, 109, 110, 384, 406
<code>headuc</code> .....	313	<code>\cslet</code> .....	34, 39, 119
<code>indexname</code> .....	181	<code>\csletcs</code> .....	34, 215
<code>indexonlyfirst</code> .....	83	<code>\csname</code> .....	6, 29, 41, 47,
<code>insertdots</code> .....	195		51, 56, 60, 62, 64, 67, 75–81, 83, 92, 107,
<code>linkcount</code> .....	151		116, 125, 128, 131, 132, 139, 144, 145,
<code>linkcountmaster</code> .....	151		147, 151, 152, 172, 193, 200–210, 216, 393
<code>markshortwords</code> .....	195	<code>\cspreto</code> .....	38
<code>markwords</code> .....	195, 196, 292, 293, 301	<code>\csuse</code> .....	9, 30, 41,
<code>nameshortaccess</code> .....	162		49, 58, 80, 81, 93–95, 106–108, 110, 118,
<code>nohyper</code> .....	82		123, 125, 126, 136, 137, 141, 142, 167,
<code>nohyperfirst</code> .....	68–70		178, 188, 189, 214, 215, 384, 385, 387, 388
<code>noshortplural</code> .....	195, 196	<code>\csxdef</code> .....	44, 47, 48, 107, 110
<code>regular</code> ....	60, 105, 216–221, 223, 226,	<code>\currentglossary</code> .....	119, 136, 138, 405
	228, 229, 231, 232, 235, 236, 238, 239,	<code>\CurrentOption</code> .....	24, 365, 366
	241, 243–245, 249–253, 256–258, 260,	<code>\CurrentTrackedLanguage</code> .....	363
	263–269, 271, 273, 275, 277–279, 281,	<code>\CurrentTrackedLanguageTag</code> .....	134
	284, 290–292, 294–297, 302, 303, 307, 309	<code>\CurrentTrackedScript</code> .....	363
<code>textformat</code> .....	65	<code>\CurrentTrackedTag</code> .....	329, 363
<code>textshortaccess</code> .....	162	<code>\CustomAbbreviationFields</code> .....	197,
<code>\cdot</code> .....	24		216, 218–221, 223, 225, 227, 230, 232–
<code>\centering</code> .....	402		238, 240, 244, 245, 248–252, 255, 258,
<code>\cGLS</code> .....	18, 19, 99, 111		259, 262, 263, 265–270, 272, 274, 277,
<code>\cGls</code> .....	18, 19, 99, 111		279, 281, 284, 285, 287, 289, 290, 292,
<code>\cgl</code> .....	18, 99, 111		294, 295, 297, 299, 300, 302, 304, 306–308
<b>D</b>			
<code>\DeclareAcronymList</code> .....	112		



<code>\forall</code> forallglsentries .....	102, 110	<code>listdottedstyle</code> .....	367
<code>\forall</code> ForEachTrackedDialect .....	329, 364	<code>listgroup</code> .....	368
<code>\forall</code> foreignlanguage .....	35, 37	<code>listhypergroup</code> .....	368
<code>\forall</code> forglsentries .....	6, 49, 169–171, 386–392	<code>mcolalmtree</code> .....	398
<code>\forall</code> forlistcsloop .....	31, 110, 125	<code>mcolalmtreegroup</code> .....	399
<code>\forall</code> forlistloop .....	98, 122, 186	<code>mcolalmtreehypergroup</code> .....	399
<code>\forall</code> futurelet .....	191	<code>mcolalmtreespannav</code> .....	400
<b>G</b>			
<code>\gdef</code> .....	59, 183	<code>mcolindexgroup</code> .....	395
<code>\Genacrfullformat</code> .....	112, 113	<code>mcolindexhypergroup</code> .....	395
<code>\genacrfullformat</code> .....	112, 113	<code>mcolindexspannav</code> .....	396
<code>\GenericAcronymFields</code> .....	112	<code>mcoltreegroup</code> .....	396
<code>\Genplacrfullformat</code> .....	113	<code>mcoltreehypergroup</code> .....	397
<code>\genplacrfullformat</code> .....	113	<code>mcoltreenonamegroup</code> .....	397
<code>\GetTrackedDialectFromLanguageTag</code> ..	35	<code>mcoltreenonamehypergroup</code> .....	398
<code>\GetTrackedDialectToMapping</code> .....	36	<code>mcoltreenonamespannav</code> .....	398
<code>\glo@grabfirst</code> .....	142	<code>mcoltreespannav</code> .....	397
<code>\glo@name</code> .....	175, 176, 179, 180	<code>sublistdotted</code> .....	367
<code>\gloaliaslabel</code> .....	86	<code>tree</code> .....	379
<code>\global</code> .....	10, 39, 119, 142	<code>treegroup</code> .....	380
<code>\glolinkprefix</code> .....	64, 66, 86, 87, 121, 135	<code>treehypergroup</code> .....	381
<code>glossaries</code> package .....		<code>treenoname</code> .....	381
.....	14, 25, 26, 37, 44, 46–48, 118, 366	<code>treenonamegroup</code> .....	382
<code>glossaries-accsupp</code> package ....	21, 24, 152, 196	<code>treenonamehypergroup</code> .....	382
<code>glossaries-extra</code> package .....	2, 364	<code>glossary-bookindex</code> package .....	366
<code>glossaries-extra-bib2gls</code> package	14, 27, 329, 364	<code>glossary-hypernav</code> package .....	85
<code>glossaries-extra-stylemods</code> package	21, 188, 329	<code>glossary-long</code> package .....	371
<code>glossaries-stylemods</code> package .....	401	<code>glossary-longbooktabs</code> package .....	371
<code>glossaries.sty</code> package .....	39	<code>glossary-tree</code> package .....	377, 378
<code>\GlossariesExtraWarning</code> .....	6,	<code>\glossaryentrynumbers</code> ...	60, 119, 142, 143
16, 36–38, 53, 55, 65, 113, 116, 126,		<code>\glossaryheader</code> .....	125,
130, 132, 133, 139, 172–177, 179, 186, 215		139, 367–376, 378–382, 393, 395–399, 403	
<code>\GlossariesExtraWarningNoLine</code>	16, 102, 111	<code>\glossaryname</code> .....	118
<code>\GlossariesWarning</code> 58, 116, 118, 122, 123, 213		<code>\glossarypostamble</code> .....	125, 139, 141
<code>\GlossariesWarningNoLine</code> .....	116, 129	<code>\glossarypreamble</code> .....	125, 138
<code>glossary</code> styles:		<code>\glossarysection</code> ...	125, 131, 132, 138, 141
altlist .....	367	<code>\glossarytitle</code> 41, 118, 119, 125, 131, 132, 138	
altlistgroup .....	368	<code>\glossarytoctitle</code> 41, 118, 125, 131, 132, 138	
altlisthypergroup .....	369	<code>\glossentry</code> .....	
almtree .....	383, 384, 393	119, 143, 366–376, 378, 380, 382, 393, 403	
almtreegroup .....	394	<code>\glossentrydesc</code> .....	
almtreehypergroup .....	395	366, 367, 369–376, 379–381, 383	
index .....	378	<code>\glossentryname</code> .....	
indexgroup .....	379	136, 366–376, 378, 380, 382, 393, 394, 401	
indexhypergroup .....	379	<code>\glossentrynameother</code> .....	138
inline .....	377	<code>\glossentrysymbol</code> 370–374, 376, 380, 381, 383	
list .....	367	<code>\glossxtrsetpopts</code> .....	187
listdotted .....	366	<code>\glostyle</code> .....	379, 381
		<code>\GLS</code> .....	99, 111, 146
		<code>\Gls</code> .....	54, 99, 111, 146

<code>\gls</code> . . . . .	37, 38, 54, 55, 99, 111, 116, 129, 146	<code>\GLSaccessfirstplural</code> . . . . .	71
<code>\gls@assign@desc</code> . . . . .	39	<code>\Glsaccessfirstplural</code> . . . . .	70
<code>\gls@assign@field</code> . . . . .	12, 13, 28, 79	<code>\glsaccessfirstplural</code> . . . . .	70
<code>\gls@checkseeallowed</code> . . . . .	52, 115	<code>\Glsaccesslong</code> . . . . .	78, 197, 205, 217, 226, 230, 231, 234, 240–243, 248, 254–257, 262, 264, 272–276, 278, 285, 286, 294, 296, 297, 300, 302, 307, 308
<code>\gls@codepage</code> . . . . .	128	<code>\glsaccesslong</code> . . . . .	. . . 77, 78, 197, 205, 206, 217, 219, 220, 222, 224, 225, 227, 229–231, 234, 236, 237, 239–245, 247, 248, 250–259, 261, 262, 264, 266, 268–282, 284, 286, 288, 289, 291, 294, 296, 297, 300, 302, 307, 308
<code>\gls@defdocnewglossaryentry</code> . . . . .	51, 100, 108	<code>\Glsaccesslongpl</code> . . . . .	79, 198, 209, 217, 226, 230, 231, 234, 240, 241, 243, 248, 254–257, 263, 264, 272– 278, 285, 286, 294, 296, 297, 300, 302, 308
<code>\gls@defglossaryentry</code> . . . . .	39, 54, 55	<code>\glsaccesslongpl</code> 78, 79, 198, 209, 210, 217, 219, 220, 222–225, 227–231, 234, 236– 245, 247, 248, 250, 252–259, 261, 262, 264, 266, 268, 269, 271–283, 285, 286, 289, 291, 294, 296, 297, 300, 302, 307, 308	
<code>\gls@dotocitle</code> . . . . .	119	<code>\GLSaccessname</code> . . . . .	71
<code>\gls@glossary</code> . . . . .	47	<code>\Glsaccessname</code> . . . . .	71
<code>\gls@grplabel</code> . . . . .	85	<code>\glsaccessname</code> . . . . .	45, 71
<code>\gls@ifnotmeasuring</code> . . . . .	99	<code>\GLSaccessplural</code> . . . . .	70
<code>\gls@level</code> . . . . .	142	<code>\Glsaccessplural</code> . . . . .	70
<code>\gls@noidxglossary</code> . . . . .	116	<code>\glsaccessplural</code> . . . . .	69
<code>\gls@org@glossaryentryfield</code> . . . . .	119	<code>\Glsaccessshort</code> . . . . .	76, 204, 213, 219, 222, 224, 227, 229, 236, 237, 239, 244– 247, 250, 252, 253, 259–261, 266, 268, 269, 271, 280–282, 288, 289, 291, 299, 305
<code>\gls@org@glossarysubentryfield</code> . . . . .	119	<code>\glsaccessshort</code> 75, 76, 197, 203, 204, 212, 217, 219, 222, 224–229, 231, 234, 236– 241, 243–248, 250–262, 264, 266, 267, 269–273, 275–278, 280, 282, 285, 286, 288, 291, 294, 296, 297, 299, 302, 305, 308	
<code>\gls@orgTrackLangRequireDialectPrefix</code> . . . . .	363	<code>\Glsaccessshortpl</code> 77, 207, 213, 220, 222– 224, 228, 229, 236, 238, 239, 245–247, 250, 252, 253, 259, 261, 266, 268, 269, 271, 280–282, 288, 289, 291, 300, 305, 308	
<code>\gls@save@numberlist</code> . . . . .	57, 58, 60	<code>\glsaccessshortpl</code> . . . . .	. . . . . 76, 77, 198, 207, 208, 212, 217, 219, 222, 224–229, 231, 234, 236–241, 243–250, 252–257, 259–264, 266, 268– 273, 275–278, 280, 282, 285, 286, 288, 289, 291, 294, 296, 297, 299, 302, 305, 308
<code>\gls@set@xr@key</code> . . . . .	47	<code>\GLSaccesssymbol</code> . . . . .	72
<code>\gls@tmplen</code> . . . . .	386–392, 394	<code>\Glsaccesssymbol</code> . . . . .	72, 185
<code>\gls@type</code> . . . . .	116		
<code>\glsabbrvdefaultfont</code> . . . . .	199, 217, 219, 222, 224, 225, 227, 230, 283, 293, 296, 306		
<code>\glsabbrvmfont</code> . . . . .	. . . . . 261–270, 272, 274, 276, 278–282		
<code>\glsabbrvfont</code> . . . . .	. . . . . 89, 90, 113, 199, 203, 204, 207, 208, 210, 212, 213, 216–225, 227, 230, 232, 234, 235, 237, 238, 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, 290– 292, 294, 296, 298–300, 302, 304, 305, 307		
<code>\glsabbrvhyphenfont</code> . . . . .	. . . . . 293–295, 299, 300, 302–304, 306		
<code>\glsabbrvonlyfont</code> . . . . .	306–309		
<code>\glsabbrvscfont</code> . . . . .	233–238, 241, 242, 244–246		
<code>\glsabbrvsmfont</code> . . . . .	. . . . . 247–251, 253, 255, 256, 258, 260		
<code>\glsabbrvuserfont</code> . . . . .	283–289, 291		
<code>\GLSaccessdesc</code> . . . . .	71		
<code>\Glsaccessdesc</code> . . . . .	71, 172, 185		
<code>\glsaccessdesc</code> . . . . .	71, 173, 189, 190		
<code>\GLSaccessdescplural</code> . . . . .	72		
<code>\Glsaccessdescplural</code> . . . . .	72		
<code>\glsaccessdescplural</code> . . . . .	72		
<code>\GLSaccessfirst</code> . . . . .	69		
<code>\Glsaccessfirst</code> . . . . .	69		
<code>\glsaccessfirst</code> . . . . .	69		

<code>\glsaccesssymbol</code> .....	72, 185, 189, 190	<code>\glsdosanitizesort</code> .....	117
<code>\GLSaccesssymbolplural</code> .....	73	<code>\glsenableentrycount</code> .....	99, 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
<code>\Glsaccesstext</code> .....	68	<code>\glsentrycurrcount</code> .....	100, 102, 108
<code>\glsaccesstext</code> .....	45, 68	<code>\Glsentrydesc</code> .....	157, 165, 173
<code>\glsacrshortcutstrue</code> .....	20, 21	<code>\glsentrydesc</code> .....	157, 164, 165, 174
<code>\glsacspacemax</code> .....	114	<code>\Glsentrydescplural</code> .....	157, 165
<code>\glsadd</code> .....	30, 50, 67, 129	<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, 154, 155, 163, 164, 325	
<code>thevalue</code> .....	9, 10	<code>\Glsentryfirstplural</code> ...	105, 150, 155, 164
<code>\glsaddpostsetkeys</code> .....	10, 67	<code>\glsentryfirstplural</code> .....	
<code>\glsaddpresetkeys</code> .....	10, 67	.....	105, 150, 155, 164, 325, 326
<code>\glsaddstoragekey</code> .....	48, 49, 166, 331	<code>\glsentryfmt</code> .....	40–42
<code>\glsbackslash</code> .....	53	<code>\Glsentryfull</code> .....	113
<code>\glscapscase</code> .....	62, 67–81, 199–212	<code>\glsentryfull</code> .....	113
<code>\glscategory</code> ..	60, 68, 81, 89, 90, 167–169, 172–179, 185, 188, 189, 199–204, 206–208	<code>\Glsentryfullpl</code> .....	113
<code>\glscategorylabel</code> .....		<code>\glsentryfullpl</code> .....	113
..	81, 160–162, 192, 195, 196, 223, 245, 260, 281, 285, 287–289, 299, 301, 304, 306	<code>\glsentryitem</code> .....	
<code>\glsclosebrace</code> .....	46, 130–132	.....	136, 138, 366–376, 378, 380, 382, 393, 404
<code>\glscounter</code> .....	9	<code>\Glsentrylong</code> ....	90, 91, 105, 150, 159, 165
<code>\glscurrententrylabel</code> .....		<code>\glsentrylong</code> .	90, 91, 105, 150, 159, 165, 166, 223, 246, 260, 281, 288, 289, 304, 326
..	57–59, 119, 127, 128, 136–140, 187, 188	<code>\Glsentrylongpl</code> .....	90, 91, 105, 159, 166
<code>\glscurrentfieldvalue</code> .....		<code>\glsentrylongpl</code> 90, 91, 105, 159, 160, 166, 327	
.....	30, 32, 33, 35, 36, 283, 330, 331	<code>\Glsentrylongplural</code> .....	150
<code>\glscustomtext</code> ..	61, 62, 75–79, 199–210, 212	<code>\glsentrylongplural</code> .....	150
<code>\glsdefaulttype</code> .	6, 17, 38, 118, 129, 138, 140	<code>\Glsentryname</code> .....	153, 163, 174, 176, 177
<code>\glsdescriptionaccessdisplay</code> ..	157, 173	<code>\glsentryname</code> .....	
<code>\glsdescriptionpluralaccessdisplay</code>		.....	136, 137, 153, 163, 181, 323, 386–392, 406
.....	157, 158	<code>\glsentrynumberlist</code> ....	117, 123, 390–392
<code>\glsdescwidth</code> .....	369–376	<code>\Glsentryplural</code> .....	154, 163
<code>\glsdetoklabel</code> ...	8, 9, 31–34, 37–39, 43– 45, 50, 51, 53, 64, 67, 83, 86, 100, 101, 106–110, 116, 119, 122, 123, 136, 138, 142, 145, 147, 172, 175, 176, 179, 387, 388	<code>\glsentryplural</code> .....	154, 163, 324, 325
<code>\glsdisplaynumberlist</code> .....	117, 121	<code>\glsentryprevcount</code> .....	100, 102, 108
<code>\glsdohyperlink</code> .....	85, 87, 88	<code>\glsentryprevmaxcount</code> .....	108
<code>\glsdohypertarget</code> .....	87, 120	<code>\glsentryprevtotalcount</code> .....	108
<code>\glsdoifexists</code> .....	15, 25, 33, 42, 45, 46, 61, 62, 67, 75–79, 87, 99, 115, 122, 123, 136, 138, 199–210	<code>\Glsentryshort</code> .....	89, 90, 158, 165
<code>\glsdoifexistsordo</code> .....	29, 30, 62	<code>\glsentryshort</code> .....	89, 90, 114, 158, 165, 285, 287, 298, 322, 323
<code>\glsdoifexistsorwarn</code> ....	15, 172, 173, 185	<code>\Glsentryshortpl</code> .....	90, 91, 159, 165
<code>\glsdoifnoexists</code> .....	39	<code>\glsentryshortpl</code> .....	
<code>\glsdonohyperlink</code> .....	66, 87, 88	.....	89–91, 158, 159, 165, 322, 323
		<code>\Glsentrysymbol</code> .....	156, 164
		<code>\glsentrysymbol</code> ....	155, 156, 164, 389–391
		<code>\Glsentrysymbolplural</code> .....	156, 164
		<code>\glsentrysymbolplural</code> .....	156, 164
		<code>\Glsentrytext</code> .....	153, 163



<code>\glsentrytext</code> .....	87, 153, 154, 163, 324	<code>\glsfirstlongfootnotefont</code> .....	
<code>\glsentrytype</code> .....	137	.....	221–224, 244–247, 258–261, 279–283
<code>\Glsentryuseri</code> .....	73	<code>\glsfirstlonghyphenfont</code> .....	293–304
<code>\glsentryuseri</code> .....	73	<code>\glsfirstlongonlyfont</code> .....	307, 308
<code>\Glsentryuserii</code> .....	74	<code>\glsfirstlonguserfont</code> .....	284–292
<code>\glsentryuserii</code> .....	74	<code>\GLSfirstplural</code> .....	318
<code>\Glsentryuseriii</code> .....	74	<code>\Glsfirstplural</code> .....	318
<code>\glsentryuseriii</code> .....	74	<code>\glsfirstplural</code> .....	318
<code>\Glsentryuseriv</code> .....	74	<code>\glsfirstpluralaccessdisplay</code> .....	155
<code>\glsentryuseriv</code> .....	74	<code>\glsforeachincategory</code> .....	213
<code>\Glsentryuserv</code> .....	74	<code>\glsgenentryfmt</code> .....	60
<code>\glsentryuserv</code> .....	75	<code>\glsgetattribute</code> .....	65, 86, 102, 106–108, 128, 146, 151, 172–177, 179, 181
<code>\Glsentryuservi</code> .....	75	<code>\glsgetcategoryattribute</code> .....	167
<code>\glsentryuservi</code> .....	75	<code>\glsgetgroupitle</code> .....	
<code>\glsextrapostnamehook</code> .....	178	.....	368, 369, 379, 381–383, 394–400
<code>\glsfieldfetch</code> .....	86	<code>\glsgetwidestname</code> .....	384
<code>\glsfieldxdef</code> .....	171	<code>\glsgroupheading</code> .....	
<code>\glsfindwidesttoplevelname</code> .....	385	.....	142, 367–376, 378–383, 393–400, 405
<code>\GLSfirst</code> .....	317	<code>\glsgroupskip</code> .....	
<code>\Glsfirst</code> .....	317, 318	.....	142, 367, 369–377, 379, 380, 382, 394, 405
<code>\glsfirst</code> .....	317	<code>\glshasattribute</code> .....	
<code>\glsfirstabbrvdefaultfont</code> .....		..	65, 86, 102, 106, 107, 109, 110, 127, 146, 151, 172–177, 179, 180, 217–221, 223, 226, 228, 229, 232, 233, 235, 236, 244, 246, 248–251, 258, 260, 262–265, 267, 268, 275, 279–281, 284, 285, 287– 292, 294–297, 299, 301–304, 306, 307, 309
.....	198, 217, 219, 222, 224, 225, 227, 230, 296	<code>\glshascategoryattribute</code> .....	167
<code>\glsfirstabbrvmfont</code> .....	262–282	<code>\glshex</code> ..	334–340, 342–348, 350–353, 355–363
<code>\glsfirstabbrvfont</code> .....	113, 197, 198, 217–231, 234, 235, 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>\glshyperlink</code> .....	87, 331
<code>\glsfirstabbrvhyphenfont</code> .....		<code>\glshypernavsep</code> .....	125
.....	293–295, 298, 299, 301–306	<code>\glshypernumber</code> .....	128, 180
<code>\glsfirstabbrvonlyfont</code> .....	307, 308	<code>\glsifattribute</code> ..	63, 64, 68, 82, 84, 93, 151, 170, 172–176, 179, 187, 190, 191, 313–321
<code>\glsfirstabbrvscfont</code> .....	233–247	<code>\glsifcategory</code> .....	169
<code>\glsfirstabbrvsmfont</code> .....	248–261	<code>\glsifcategoryattribute</code> .....	
<code>\glsfirstabbrvuserfont</code> .....	284–292	.....	81, 160–162, 168, 169, 195, 196
<code>\glsfirstaccessdisplay</code> .....	154, 155	<code>\glsifnotregular</code> .....	68
<code>\glsfirstlongdefaultfont</code> .....		<code>\glsifnotregularcategory</code> .....	169
.....	217, 219, 225, 227, 230, 233–243, 248– 258, 262, 263, 265, 266, 269–273, 276, 277	<code>\glsifplural</code> .....	
<code>\glsfirstlongemfont</code> .....		.....	62, 67, 69, 70, 72, 73, 75–79, 190, 199–211
.....	263–265, 267, 268, 274, 275, 277–279	<code>\glsifregular</code> .....	60, 68, 105, 150
<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>\glsifregularcategory</code> .....	169
		<code>\glsifusetranslator</code> .....	41
		<code>\glsignore</code> .....	58, 59
		<code>\glsinlinedescformat</code> .....	377
		<code>\glsinlinesubdescformat</code> .....	377



<code>\glsinsert</code> .....	62,	<code>\glslongfootnotefont</code> .....	
	67, 75–79, 199–212, 292, 299, 301, 304, 306		221, 222, 224, 244, 246, 258, 260, 280, 282
<code>\glskeylisttok</code> .....	112, 194, 197	<code>\glslonghyphenfont</code> .....	
<code>\glslabel</code> .....	8, 9,		293, 294, 296, 297, 299, 302, 304
	29, 43, 45, 60, 63–66, 81–83, 86, 87, 114,	<code>\glslongonlyfont</code> .....	306, 307
	147, 151, 188–190, 210–212, 223, 246,	<code>\glslongpltok</code> .....	
	260, 281, 285, 287–289, 299, 301, 304, 306		196, 197, 217–221, 230, 232–236, 240,
<code>\glslabeltok</code> .	112, 194, 196, 217–221, 223,		244, 248, 249, 251, 255, 258, 262–268,
	225–230, 232–238, 240, 244, 246, 248–		272, 274, 277, 279, 284, 285, 287, 289–
	251, 253, 255, 258, 260, 262–270, 272,		292, 294–297, 299, 300, 302, 303, 307, 308
	274, 275, 277, 279–281, 284, 285, 287–	<code>\glslongpluralaccessdisplay</code> ...	159, 160
	292, 294–297, 299, 301–304, 306, 307, 309	<code>\glslongtok</code> .....	112,
<code>\glsletentryfield</code> .....	181		194, 195, 197, 217–221, 223, 225, 227,
<code>\glslink</code> .....	112, 113		230, 232–238, 240, 244, 245, 248, 249,
<code>\glslink options</code>			251, 253, 255, 258, 260, 262–270, 272,
counter .....	10		274, 277, 279, 281, 284, 285, 287, 289–
format .....	180		292, 294–297, 299, 300, 302–304, 307, 308
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, 143	<code>\glsnameaccessdisplay</code> ..	153, 174, 175, 177
wrgloss .....	8, 63	<code>\glsnamefont</code> .....	174, 175, 177, 179
<code>\glslinkcheckfirsthyperhook</code> .....	82	<code>\glsnavhyperlink</code> .....	125
<code>\glslinkpostsetkeys</code> .....	10, 65, 147	<code>\glsnavhyperlinkname</code> .....	85
<code>\glslinkpresetkeys</code> .....	10, 65, 147	<code>\glsnavhypertarget</code> .....	
<code>\glslinkvar</code> .....	84, 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>\glsnoidxdisplaylocclishandler</code> ...	122
<code>\glslistnavigationitem</code> .....	368, 369	<code>\glsnoidxloclist</code> .....	123, 142, 143
<code>\glslistprelocation</code> .....	367, 368	<code>\glsnoidxnumberlistloophandler</code> ...	122
<code>\glslocalunset</code> .....	62, 147	<code>\glsnonextpages</code> .....	119
<code>\glslongaccessdisplay</code> .....	159	<code>\glsnonumberlistfalse</code> .....	57
<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, 283, 293, 306	<code>\glsnumberlistloop</code> .....	117
<code>\glslongemfont</code> ..	261, 264, 267, 274, 277, 278	<code>\glsnumlistlastsep</code> .....	122
<code>\glslongfont</code> .....	90, 199, 205, 206,	<code>\glsnumlistsep</code> .....	122
	209, 210, 217–220, 222, 224, 225, 227,	<code>\glsopenbrace</code> .....	46, 130–132
	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> .....	99, 111, 146
	288, 291, 294, 296, 299, 302, 304, 307, 308	<code>\Glspl</code> .....	55, 99, 111, 146

<code>\glsp1</code> .....	54, 99, 111, 146	<code>\glstextaccessdisplay</code> .....	153
<code>\GLSplural</code> .....	316, 317	<code>\glstextformat</code> .....	62, 65, 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> .....	134, 195, 199	<code>\glstreechildprelocation</code> ...	378, 380, 382
<code>\glspostdescription</code> .....	16, 17, 119, 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, 75–79, 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, 206–208, 210	<code>\glstreenavigationfmt</code>	379, 381, 383, 395–400
<code>\glsssetattribute</code> .....	217–221, 223, 225–230, 232–238, 240, 244, 246, 248–251, 253, 255, 258, 260, 262–265, 267–270, 272, 274, 275, 277, 279–281, 284, 285, 287, 288, 290– 292, 294–297, 299, 301–304, 306, 307, 309	<code>\glstreenonamechilddesc</code> .....	382
<code>\glsssetcategoryattribute</code> .....	100, 111, 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, 233, 235–238, 244, 245, 248, 249, 251, 253, 258, 260, 262– 270, 279, 281, 284, 285, 287, 289–292, 294, 295, 299, 300, 302–304, 306, 307, 309	<code>\glstreesubitem</code> .....	378, 404
<code>\glssshortpluralaccessdisplay</code> ..	158, 159	<code>\glstreesubsubitem</code> .....	378, 405
<code>\glssshorttok</code> .....	112, 194–197, 216–221, 223, 225, 227, 232–238, 240, 244, 245, 248–251, 253, 255, 258–260, 262, 263, 265–270, 272, 274, 279, 281, 284, 285, 287, 289–292, 294, 295, 297, 299, 300, 302–304, 306–308	<code>\glstreesymbol</code> .....	378, 380
<code>\glsssubentryitem</code> .....	137, 366–376, 378, 380, 382, 394, 404	<code>\GLstrLetField</code> .....	34
<code>\glsssymbolaccessdisplay</code> .....	155, 156	<code>\glstype</code> .....	62, 64, 75–79, 147, 200–210
<code>\glsssymbolpluralaccessdisplay</code> .....	156	<code>\glsunset</code> .....	50, 62, 102–104, 147
<code>\glstarget</code> .....	136, 138, 366–376, 378, 380, 382, 393, 394, 404, 405	<code>\glssuserdescription</code> ...	284, 285, 287, 290
<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>\glsxtr</code> .....	55
		<code>\glsxtr@@do@wrglossary</code> .....	8, 10, 12, 13
		<code>\glsxtr@addloccllistfield</code> .....	14
		<code>\glsxtr@addunused</code> .....	50
		<code>\glsxtr@applyabbrvfmt</code> .....	210
		<code>\glsxtr@applyabbrvstyle</code> ...	193, 195, 213
		<code>\glsxtr@counterrecord</code> .....	136
		<code>\glsxtr@do@alsoindex@wrglossary</code> ...	14
		<code>\glsxtr@doooption</code> .....	5, 16, 17, 23, 24, 26
		<code>\glsxtr@fields</code> .....	134
		<code>\glsxtr@headentry@p</code> .....	93, 94
		<code>\glsxtr@hyperoutsidefalse</code> .....	64
		<code>\glsxtr@hyperoutsidettrue</code> .....	63, 64
		<code>\glsxtr@ifnextpunc</code> .....	191
		<code>\glsxtr@ifpunctoken</code> .....	191
		<code>\glsxtr@inc@linkcount</code> .....	65, 152
		<code>\glsxtr@inc@wrglossaryctr</code> ...	8, 9, 23, 27

\glxtr@indexonly@saveentrycounter	\glxtrAltTreeSetHangIndent ... 383, 393
..... 13, 14, 27	\glxtrAltTreeSetSubHangIndent ... 394
\glxtr@keylist ..... 53–55	\glxtralttreeSubSymbolDescLocation 394
\glxtr@label ..... 406	\glxtralttreeSymbolDescLocation ..
\glxtr@langtag ..... 134	..... 383, 394
\glxtr@linkprefix ..... 134, 135	\glxtrassignfieldfont ..... 68–75
\glxtr@loaddialect ..... 329, 364	\glxtrautoindex ..... 181
\glxtr@makeglossaries ..... 115	\glxtrautoindexassignsort ..... 181
\glxtr@newabbreviation ..... 113, 194	\glxtrautoindexentry ..... 181
\glxtr@next ..... 192	\glxtrbibaddress ..... 331
\glxtr@org@odo@wrglossary ..... 27	\glxtrbibauthor ..... 331
\glxtr@org@dohyperlink ..... 85	\glxtrbibbooktitle ..... 331
\glxtr@org@getgrouptitle ..... 124	\glxtrbibchapter ..... 331
\glxtr@org@newignoredglossary ..... 40	\glxtrbibedition ..... 331
\glxtr@org@makenoidxglossaries ..... 51	\glxtrbibhowpublished ..... 331
\glxtr@pluralsuffixes ..... 134	\glxtrbibinstitution ..... 331
\glxtr@process ..... 139, 140	\glxtrbibjournal ..... 331
\glxtr@provideignoredglossary ..... 41	\glxtrbibmonth ..... 331
\glxtr@punclist ..... 191	\glxtrbibnote ..... 331
\glxtr@record ..... 11, 134	\glxtrbibnumber ..... 331
\glxtr@record@nr ..... 13	\glxtrbiborganization ..... 331
\glxtr@recordsee ..... 12	\glxtrbibpages ..... 331
\glxtr@resource ..... 133, 134	\glxtrbibpublisher ..... 331
\glxtr@s@newignoredglossary ..... 40	\glxtrbibschool ..... 331
\glxtr@s@provideignoredglossary ... 41	\glxtrbibseries ..... 331
\glxtr@saveentrycounter ..... 8, 9, 12, 83	\glxtrbibtitle ..... 331
\glxtr@setaccessdisplay ..... 179	\glxtrbibtype ..... 331
\glxtr@setbookindexmark ..... 406	\glxtrbibvolume ..... 331
\glxtr@setup@record ..... 13, 14, 26, 27	\glxtrbookindexatendgroup ..... 404
\glxtr@shortcutsval ..... 134, 135	\glxtrbookindexatsubendgroup ..... 404
\glxtr@texencoding ..... 135	\glxtrbookindexatsubsubendgroup .. 404
\glxtr@undefaction@nr ..... 7	\glxtrbookindexbetween ..... 404
\glxtr@undefaction@val ..... 7	\glxtrbookindexbookmark ..... 405
\glxtr@usesee ..... 45	\glxtrbookindexcols ..... 403
\glxtr@warnonexistsordo . 7, 13, 14, 43, 44	\glxtrbookindexcolspread ..... 403
\glxtr@writefields ..... 133	\glxtrbookindexfirstmarkfmt ..... 406
\glxtrabbreviationfont ..... 60	\glxtrbookindexformatheader ..... 405
\glxtrabbrvfootnote .....	\glxtrbookindexgroupskip ..... 405
.... 221–223, 244–246, 258–260, 279–281	\glxtrbookindexlastmarkfmt ..... 406
\glxtrabbrvpluralsuffix ..... 134,	\glxtrbookindexmulticolseenv ..... 403
199, 217, 219, 222, 224, 225, 227, 230,	\glxtrbookindexname ..... 401, 404
233, 247, 261, 284, 293, 296, 299, 304, 307	\glxtrbookindexparentchildsep 401, 403
\glxtrabbrvtype ..... 17, 197	\glxtrbookindexparentschildsep .
\glxtractivatenopost ..... 119	..... 403, 404
\glxtraddallcrossrefs ..... 49	\glxtrbookindexprelocation ... 401, 404
\glxtralias ..... 83	\glxtrbookindexsubbetween ..... 404
\glxtrAltTreeIndent ..... 383, 384	\glxtrbookindexsubname ..... 405
\glxtralttreeInit ..... 393, 399, 400	\glxtrbookindexsubprelocation .... 405
\glxtrAltTreePar ..... 383	\glxtrbookindexsubsubbetween ..... 404

<code>\glxtrbookindexthepage</code> .....	406	<code>\glxtrfull</code> .....	18, 19, 320
<code>\glxtrcat</code> .....	54, 55	<code>\Glsxtrfullformat</code> .....	
<code>\glxtrchecknohyperfirst</code> .....	69–71	.....	198, 212, 214, 215, 217, 219, 222,
<code>\glxtrcombingdiacriticIIrules</code> ..	335		224, 226, 228, 231, 234, 236, 238, 239,
<code>\glxtrcombingdiacriticIrules</code> ..	335		242–244, 246, 248, 250, 252, 253, 256,
<code>\glxtrcombingdiacriticIVrules</code> ..	335		257, 259, 260, 262, 264, 266, 268, 270,
<code>\glxtrComputeTreeIndent</code> .....	393, 394		271, 273, 275, 277, 279, 280, 282, 285,
<code>\glxtrComputeTreeSubIndent</code> .....	394		286, 288, 291, 294, 297, 300, 302, 305, 307
<code>\glxtrcounterprefix</code> .....	127	<code>\glxtrfullformat</code> .....	
<code>\glxtrcurrencyrules</code> .....	338	.....	198, 212, 214, 215, 217, 219,
<code>\Glsxtrdefaultsubsequentfmt</code> ...	213, 214		222, 224, 226, 228, 231, 234, 236, 238,
<code>\glxtrdefaultsubsequentfmt</code> ...	212, 214		239, 242–244, 246, 248, 250, 252, 253,
<code>\Glsxtrdefaultsubsequentplfmt</code> .	213, 214		256–258, 260, 262, 264, 266, 267, 269,
<code>\glxtrdefaultsubsequentplfmt</code> .	213, 214		271, 273, 275, 277, 278, 280, 282, 284,
<code>\GlsXtrDefineAbbreviationShortcuts</code> .	20		286, 288, 291, 294, 297, 300, 302, 305, 307
<code>\GlsXtrDefineAcShortcuts</code> .....	20, 21	<code>\GLSxtrfullpl</code> .....	18, 19, 321, 322
<code>\GlsXtrDefineOtherShortcuts</code> .....	20	<code>\Glsxtrfullpl</code> .....	18, 19, 322
<code>\glxtrdetoklocation</code> .....	145	<code>\glxtrfullpl</code> .....	18, 19, 321
<code>\glxtrdiscardperiod</code> .....	188	<code>\Glsxtrfullplformat</code> .....	
<code>\glxtrdisplayendloc</code> .....	126	.	198, 212, 214, 215, 217, 220, 222, 224,
<code>\glxtrdisplayendloohook</code> .....	126		226, 228, 231, 234, 236, 238, 239, 242,
<code>\glxtrdisplaysingleloc</code> .....	126		243, 245, 246, 248, 250, 252, 254, 256,
<code>\glxtrdisplaystartloc</code> .....	126		258, 259, 261, 263, 264, 266, 268, 270,
<code>\glxtrdoautoindexname</code> .....	84, 178		271, 273, 275, 277, 279, 280, 282, 285,
<code>\glxtrdopostpunc</code> .....	223, 246, 260, 281		286, 288, 291, 294, 297, 300, 302, 305, 308
<code>\glxtrdowrglossaryhook</code> .....	84	<code>\glxtrfullplformat</code> .....	
<code>\GlsXtrDualField</code> .....	331	.....	211, 212, 214, 215, 217, 219,
<code>\glxtrremsuffix</code> .....	262, 264, 266,		222, 224, 226, 228, 231, 234, 236, 238,
	267, 269, 270, 272, 274, 276, 277, 280, 282		239, 242–244, 246, 248, 250, 252, 253,
<code>\GlsXtrEnableEntryCounting</code> .....	111		256–258, 260, 262, 264, 266, 268, 270,
<code>\GlsXtrEnableEntryUnitCounting</code> ....	100		271, 273, 275, 277, 279, 280, 282, 285,
<code>\GlsXtrEnableOnTheFly</code> .....	53, 55, 56		286, 288, 291, 294, 297, 300, 302, 305, 307
<code>\glxtrendfor</code> .....	32	<code>\glxtrfullsep</code> 197, 198, 217–220, 222–229,	
<code>\glxtrfieldlistgadd</code> .....	136		231, 233–241, 243, 245, 247–257, 259,
<code>\glxtrfieldtitlecase</code> ..	173, 174, 176, 179		261–278, 280–283, 293–298, 301–304, 308
<code>\glxtrfieldtitlecasecs</code> .....	172	<code>\glxtrrgenabbrvfmt</code> .....	60
<code>\glxtrfieldxifinlist</code> .....	141	<code>\glxtrgeneralpuncIrules</code> .....	338
<code>\glxtrfirstscfont</code> .....	233	<code>\glxtrgeneralpuncIrules</code> .....	338
<code>\glxtrfirstsmfont</code> .....	247	<code>\glxtrgetgroupitle</code> .....	125, 405
<code>\GlsXtrFmtDefaultOptions</code> .....	30	<code>\glxtrgroupfield</code> .....	142
<code>\glxtrfmtdisplay</code> .....	30	<code>\Glsxtrheadfirst</code> .....	312
<code>\GlsXtrFmtField</code> .....	30	<code>\glxtrheadfirst</code> .....	312
<code>\glxtrfootnotename</code> .....		<code>\Glsxtrheadfirstplural</code> .....	312
.....	221, 223, 244, 245, 258, 259, 279, 281	<code>\glxtrheadfirstplural</code> .....	312
<code>\GlsXtrForeignTextField</code> .....	35	<code>\Glsxtrheadfull</code> .....	312
<code>\GlsXtrFormatLocationList</code> 57, 60, 390–392		<code>\glxtrheadfull</code> .....	312
<code>\GLSxtrfull</code> .....	18, 19, 320, 321	<code>\Glsxtrheadfullpl</code> .....	312
<code>\Glsxtrfull</code> .....	18, 19, 321	<code>\glxtrheadfullpl</code> .....	312
		<code>\Glsxtrheadlong</code> .....	312

<code>\glxtrheadlong</code> .....	312	259, 261, 269–271, 273, 274, 276, 278,
<code>\Glsxtrheadlongpl</code> .....	312	280, 282, 286, 288, 296, 300, 305, 308, 327
<code>\glxtrheadlongpl</code> .....	312	<code>\Glsxtrinlinefullplformat</code> 198, 202, 214,
<code>\Glsxtrheadname</code> .....	312	215, 223, 224, 226, 228, 229, 231, 238–
<code>\glxtrheadname</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>\glxtrheadplural</code> .....	312	281, 282, 286, 289, 297, 300, 305, 308, 328
<code>\Glsxtrheadshort</code> .....	312	<code>\glxtrinlinefullplformat</code> .....
<code>\glxtrheadshort</code> .....	311	..... 198, 201, 202, 214, 215,
<code>\Glsxtrheadshorttpl</code> .....	312	222, 224, 225, 227, 229, 231, 237, 239–
<code>\glxtrheadshorttpl</code> .....	311	241, 243, 245, 247, 252–255, 257, 259,
<code>\Glsxtrheadtext</code> .....	312	261, 269, 270, 272, 273, 275, 276, 278,
<code>\glxtrheadtext</code> .....	312	280, 282, 286, 289, 296, 300, 305, 308, 328
<code>\glxtrhyperlink</code> .....	23, 24, 86	<code>\glxtrininsertinsidefalse</code> .....
<code>\glxtrhyphensuffix</code> .....	294, 302	216
<code>\glxtrifcounttrigger</code> .....	102–104	<code>\GlsXtrInternalLocationHyperlink</code> 23, 127
<code>\glxtrifcustomdiscardperiod</code> .....	188	<code>\glxtrLatinA</code> .....
<code>\glxtrifemptyglossary</code> .....	125, 131, 138	340–345
<code>\GlsXtrIfFieldCmpNum</code> .....	32	<code>\glxtrLatinAELigature</code> .....
<code>\GlsXtrIfFieldEqNum</code> .....	137	342, 344, 345
<code>\GlsXtrIfFieldNonZero</code> .....	330	<code>\glxtrLatinE</code> .....
<code>\glxtrifhasfield</code> ....	35, 83, 330, 331, 401	340–345
<code>\glxtrifhyphenstart</code> .....	293, 295, 298, 301, 303, 304	<code>\glxtrLatinEszettSs</code> .....
<code>\glxtrifindexing</code> .....	84	341–343, 345
<code>\glxtrifinmark</code> .....	67, 93–96, 310–312	<code>\glxtrLatinEszettSz</code> .....
<code>\glxtrifnextpunc</code> .....	191, 192	341, 344
<code>\glxtrifperiod</code> .....	188, 190, 191	<code>\glxtrLatinEth</code> .....
<code>\glxtrifrecordtrigger</code> .....	148–150	340–344
<code>\glxtrifwasfirstuse</code> .....	67, 69, 70, 75–79, 81, 114,	<code>\glxtrLatinH</code> .....
	189, 190, 200, 203–210, 223, 246, 260,	340–345
	281, 282, 285, 287–289, 299, 301, 304, 306	<code>\glxtrLatinI</code> .....
<code>\glxtrinclinkcounter</code> .....	151	340–345
<code>\glxtrindexaliased</code> .....	83	<code>\glxtrLatinInsularG</code> .....
<code>\glxtrindexseealso</code> .....	48, 49	340–345
<code>\glxtrinithyperoutside</code> .....	65	<code>\glxtrLatinK</code> .....
<code>\glxtrinitwrgloss</code> .....	65, 147	340–345
<code>\glxtrinitwrglossbeforefalse</code> .....	63	<code>\glxtrLatinL</code> .....
<code>\glxtrinitwrglossbeforetrue</code> .....	63	340–345
<code>\Glsxtrinlinefullformat</code> 198, 200, 214,		<code>\glxtrLatinM</code> .....
215, 222, 224, 225, 227, 229, 231, 237,		340–345
239–241, 243, 245, 247, 252–255, 257,		<code>\glxtrLatinN</code> .....
259, 261, 269, 271–273, 275, 276, 278,		340–345
281, 282, 286, 289, 297, 300, 305, 308, 327		<code>\glxtrLatinO</code> .....
<code>\glxtrinlinefullformat</code> .....		340–345
..... 198, 199, 201, 214, 215,		<code>\glxtrLatinOELigature</code> .....
222, 224, 225, 227, 229, 231, 237, 239–		342, 344, 345
241, 243, 245, 247, 251, 253–255, 257,		<code>\glxtrLatinP</code> .....
		340–345
		<code>\glxtrLatinS</code> .....
		340–345
		<code>\glxtrLatinT</code> .....
		340–345
		<code>\glxtrLatinThorn</code> .....
		344
		<code>\glxtrLatinX</code> .....
		340–345
		<code>\glxtrlocationhyperlink</code> .....
		127
		<code>\glxtrlocrangefmt</code> .....
		126
		<code>\GLSxtrlong</code> .....
		18, 19, 319
		<code>\Glsxtrlong</code> .....
		18, 19, 319, 320
		<code>\glxtrlong</code> .....
		18, 19, 319
		<code>\glxtrlonghyphen</code> .....
		300
		<code>\glxtrlonghyphennoshort</code> .....
		296, 297
		<code>\glxtrlonghyphenshort</code> .....
		294
		<code>\glxtrlongnoshortdescname</code> .
		230, 277, 296
		<code>\glxtrlongnoshortname</code> .....
		..... 232, 240, 255, 272, 274, 297
		<code>\GLSxtrlongpl</code> .....
		18, 19, 319, 320
		<code>\Glsxtrlongpl</code> .....
		18, 19, 320
		<code>\glxtrlongpl</code> .....
		18, 19, 319

<code>\glxtrlongshortdescname</code> .....	<code>\glxtronlyname</code> .....	307
..... 218, 234, 249, 263, 265, 294, 300	<code>\glxtronlysuffix</code> .....	307
<code>\glxtrlongshortdescsort</code> .....	<code>\glxtrorg@ifKV@glslink@hyper</code> .....	61
.... 218, 234, 249, 263, 265, 290, 295, 300	<code>\glxtrorglong</code> .....	195, 218, 296
<code>\glxtrlongshortname</code> .....	<code>\glxtrorgshort</code> .....	195, 218
216, 233, 248, 262, 263, 284, 285, 294, 299	<code>\GLSxtrp</code> .....	93
<code>\glxtrlongshortuserdescname</code> ..	<code>\Glsxtrp</code> .....	92
287, 290	<code>\glxtrp</code> .....	92, 94
<code>\glxtrmarkhook</code> .....	<code>\glxtrparen</code> .....	189, 190,
309, 310	197, 198, 217–220, 222–229, 231, 233–	
<code>\glxtrMathItalicAlpha</code> .	241, 243, 245, 247–257, 259, 261–273,	
349, 350, 353, 354	275–278, 280–283, 293–298, 301–304, 308	
<code>\glxtrMathItalicBeta</code> ..	<code>\Glsxtrpl</code> .....	55
349, 350, 353, 354	<code>\glxtrpl</code> .....	55
<code>\glxtrMathItalicChi</code> .....	<code>\glxtrpostdescription</code> .	120, 170, 187, 377
350, 354, 355	<code>\glxtrposthyphenlong</code> .....	304, 306
<code>\glxtrMathItalicDelta</code> .	<code>\glxtrposthyphenshort</code> .....	299, 301
349, 350, 353, 354	<code>\glxtrposthyphensubsequent</code> .....	
<code>\glxtrMathItalicEpsilon</code> 349, 350, 353, 354	299, 301, 304, 306	
<code>\glxtrMathItalicEta</code> ...	<code>\glxtrpostlink</code> .....	188
349, 350, 353, 354	<code>\glxtrpostlinkendsentence</code> .....	188
<code>\glxtrMathItalicGamma</code> .	<code>\glxtrpostlinkhook</code> .....	188
349, 350, 353, 354	<code>\glxtrpostlocalreset</code> .....	99, 101, 109
<code>\glxtrMathItalicIota</code> ..	<code>\glxtrpostlocalunset</code> .....	98, 101, 109
349, 350, 353, 354	<code>\glxtrpostlongdescription</code> .....	39
<code>\glxtrMathItalicKappa</code> .	<code>\glxtrpostnamehook</code> .....	175–177, 180
349, 350, 353, 354	<code>\GlsXtrPostNewAbbreviation</code> .	197, 214,
<code>\glxtrMathItalicLambda</code> 349, 350, 353, 354	215, 217–221, 223, 225–230, 232, 233,	
<code>\glxtrMathItalicMu</code> ....	235–238, 240, 244, 245, 248–251, 253,	
349, 350, 353, 354	255, 258, 260, 262–265, 267–270, 272,	
<code>\glxtrMathItalicNu</code> ....	274, 275, 277, 279, 281, 284, 285, 287–	
349, 350, 353, 354	292, 294–297, 299, 301–304, 306, 307, 309	
<code>\glxtrMathItalicOmega</code> ....	<code>\glxtrpostreset</code> .....	98, 101, 109
350, 354, 355	<code>\glxtrpostunset</code> .....	97, 101, 109
<code>\glxtrMathItalicOmicron</code> 349, 350, 353, 355	<code>\glxtrprelocation</code> .....	
<code>\glxtrMathItalicPhi</code> ...	367, 369, 371, 373, 375, 378, 401	
349, 350, 354, 355	<code>\glxtrprotectlinks</code> .....	86, 87
<code>\glxtrMathItalicPsi</code> .....	<code>\GlsXtrRecordCounter</code> .....	11
350, 354, 355	<code>\glxtrrecordtriggervalue</code> .....	146
<code>\glxtrMathItalicRho</code> ...	<code>\GlsXtrRecordWarning</code> .....	132
349, 350, 354, 355	<code>\glxtrregularfont</code> .....	60, 68
<code>\glxtrMathItalicSigma</code> .	<code>\glxtrresourcecount</code> .....	133
349, 350, 354, 355	<code>\glxtrresourcefile</code> .....	133
<code>\glxtrMathItalicTau</code> ...	<code>\glxtrresourceinit</code> .....	133
349, 350, 354, 355	<code>\glxtrrestoremarkhook</code> .....	310
<code>\glxtrMathItalicTheta</code> .	<code>\glxtrrestorepostpunc</code> .....	120
349, 350, 353, 354	<code>\glxtrscfont</code> .....	233
<code>\glxtrMathItalicUpsilon</code> 349, 350, 354, 355	<code>\glxtrscsuffix</code> .....	
<code>\glxtrMathItalicXi</code> ....	234, 235, 237, 238, 241, 242, 244, 246	
349, 350, 353, 354		
<code>\glxtrMathItalicZeta</code> ..		
349, 350, 353, 354		
<code>\glxtrnewabbrevpresetkeyhook</code> ....		196
<code>\glxtrnewnumber</code> .....		19
<code>\glxtrnewsymbol</code> .....		19
<code>\glxtrNoGlossaryWarning</code> ....		14, 21, 128
<code>\GlsXtrNoGlsWarningAutoMake</code> .....		132
<code>\GlsXtrNoGlsWarningBuildInfo</code> .....		132
<code>\GlsXtrNoGlsWarningCheckFile</code> .....		132
<code>\GlsXtrNoGlsWarningEmptyMain</code> ..		131, 132
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...		131
<code>\GlsXtrNoGlsWarningEmptyStart</code> ....		131
<code>\GlsXtrNoGlsWarningHead</code> .....		131
<code>\GlsXtrNoGlsWarningMisMatch</code> .....		132
<code>\GlsXtrNoGlsWarningNoOut</code> .....		132
<code>\GlsXtrNoGlsWarningTail</code> .....		132
<code>\glxtrnopostpunc</code> .....		119
<code>\glxtronlydescname</code> .....		308
<code>\glxtronlydescsort</code> .....		308



<code>\GlsXtrSetActualChar</code> .....	184	<code>\Glsxtrtitlefull</code> .....	311–313, 327, 328
<code>\glxtrsetaliasnoindex</code> .....	13, 14, 83	<code>\glxtrtitlefull</code> .....	311, 312, 327
<code>\GlsXtrSetEncapChar</code> .....	184	<code>\Glsxtrtitlefullpl</code> .....	311–313, 328
<code>\GlsXtrSetEscChar</code> .....	184	<code>\glxtrtitlefullpl</code> .....	311–313, 328
<code>\glxtrsetfieldifexists</code> .....	34	<code>\Glsxtrtitlelong</code> .....	311, 312, 326
<code>\GlsXtrSetLevelChar</code> .....	184	<code>\glxtrtitlelong</code> .....	311, 312, 326
<code>\glxtrsetpopts</code> .....	91	<code>\Glsxtrtitlelongpl</code> .....	311, 312, 327
<code>\glxtrsetupfulldefs</code> .....		<code>\glxtrtitlelongpl</code> .....	311, 312, 327
.....	200–202, 223, 246, 260, 282	<code>\Glsxtrtitlename</code> .....	311, 312, 323, 324
<code>\GLSxtrshort</code> .....	18, 19, 95, 96, 313, 314	<code>\glxtrtitlename</code> .....	311, 312, 323
<code>\Glsxtrshort</code> .....	18, 19, 314	<code>\glxtrtitleorpdforheading</code> .....	
<code>\glxtrshort</code> .....	18, 313	.....	25, 136, 137, 311, 312
<code>\glxtrshortdescname</code> ...	227, 238, 252, 270	<code>\Glsxtrtitleplural</code> .....	311, 312, 325
<code>\glxtrshorthyphen</code> .....	305	<code>\glxtrtitleplural</code> .....	311, 312, 324
<code>\glxtrshorthyphenlong</code> .....	302	<code>\Glsxtrtitleshort</code> .....	311, 312, 323
<code>\glxtrshortlongdescname</code> .....		<code>\glxtrtitleshort</code> .....	311, 312, 322
.....	220, 236, 250, 266, 268, 303, 306	<code>\Glsxtrtitleshortpl</code> .....	311, 312, 323
<code>\glxtrshortlongdescsort</code> .....		<code>\glxtrtitleshortpl</code> .....	311, 312, 322
....	220, 236, 250, 266, 268, 292, 303, 306	<code>\Glsxtrtitletext</code> .....	311, 312, 324
<code>\glxtrshortlongname</code> .....		<code>\glxtrtitletext</code> .....	311, 312, 324
	219, 235, 249, 265, 267, 287, 290, 302, 304	<code>\GlsXtrTotalRecordCount</code> .....	146
<code>\glxtrshortlonguserdescname</code> ..	289, 292	<code>\glxtrtreetopindent</code> .....	384, 392
<code>\glxtrshortnolongname</code> .	225, 237, 251, 269	<code>\glxtrundefaction</code> ..	7, 13, 14, 28, 40, 43, 44
<code>\GLSxtrshortpl</code> .....	18, 19, 313, 314	<code>\glxtrundeftag</code> .....	27, 122, 123
<code>\Glsxtrshortpl</code> .....	18, 19, 314	<code>\GlsXtrUnknownDialectWarning</code> .....	37
<code>\glxtrshortpl</code> .....	18, 19, 313	<code>\glxtrunsrtdo</code> .....	140, 141
<code>\glxtrsmfont</code> .....	247	<code>\glxtrUpAlpha</code> .....	348, 353, 354
<code>\glxtrsmsuffix</code> .....		<code>\glxtrUpBeta</code> .....	348, 353, 354
....	248, 250, 251, 253, 255, 256, 258, 260	<code>\glxtrUpChi</code> .....	348, 349, 354, 355
<code>\GlsXtrStandaloneGlossaryType</code> .	136, 138	<code>\glxtrUpDelta</code> .....	348, 349, 353, 354
<code>\GlsXtrStandaloneSubEntryItem</code> .	136, 138	<code>\glxtrUpDigamma</code> .....	348, 349, 353
<code>\Glsxtrsubsequentfmt</code> .....		<code>\glxtrUpEpsilon</code> .....	348, 349, 353, 354
.....	211, 214, 230, 241, 242,	<code>\glxtrUpEta</code> .....	348, 349, 353, 354
	255, 257, 273, 274, 276, 278, 296, 299, 305	<code>\glxtrUpGamma</code> .....	348, 349, 353, 354
<code>\glxtrsubsequentfmt</code> .....		<code>\glxtrUpIota</code> .....	348, 349, 353, 354
.....	211, 214, 230, 241, 242,	<code>\glxtrUpKappa</code> .....	348, 349, 353, 354
	255, 256, 272, 274, 276, 278, 296, 299, 305	<code>\glxtrUpLambda</code> .....	348, 349, 353, 354
<code>\Glsxtrsubsequentplfmt</code> .....		<code>\glxtrUpMu</code> .....	348, 349, 353, 354
.....	211, 214, 230, 241, 242,	<code>\glxtrUpNu</code> .....	348, 349, 353, 354
	255, 257, 273, 274, 276, 278, 296, 299, 305	<code>\glxtrUpOmega</code> .....	348, 349, 354, 355
<code>\glxtrsubsequentplfmt</code> .....		<code>\glxtrUpOmicron</code> .....	348, 349, 353, 355
.....	211, 214, 230, 241, 242,	<code>\glxtrUpPhi</code> .....	348, 349, 354, 355
	255, 257, 272, 274, 276, 278, 296, 299, 305	<code>\glxtrUpPi</code> .....	348, 349, 354, 355
<code>\glxtrsupplocationurl</code> .....	127, 128	<code>\glxtrUpPsi</code> .....	348, 349, 354, 355
<code>\glxtrtagfont</code> .....	187	<code>\glxtrUpRho</code> .....	348, 349, 354, 355
<code>\Glsxtrtitlefirst</code> .....	311, 312, 325	<code>\glxtrUpSigma</code> .....	348, 349, 354, 355
<code>\glxtrtitlefirst</code> .....	311, 312, 325	<code>\glxtrUpTau</code> .....	348, 349, 354, 355
<code>\Glsxtrtitlefirstplural</code> ....	311, 312, 326	<code>\glxtrUpTheta</code> .....	348, 349, 353, 354
<code>\glxtrtitlefirstplural</code> .....	311, 312, 325, 326	<code>\glxtrUpUpsilon</code> .....	348, 349, 354, 355

<code>\glstrUpXi</code> .....	348, 349, 353, 355	<code>\ifdef</code> .....	14, 19, 26, 30, 35, 37, 43, 44, 46, 47, 50, 56, 57, 82, 85, 87, 93–95, 118, 122, 123, 134, 135, 143, 170, 171, 184, 187, 283, 311, 322–328, 330, 363, 364, 366–369, 377–383, 394–400, 402, 405, 406
<code>\glstrUpZeta</code> .....	348, 349, 353, 354	<code>\ifdefempty</code> 7–9, 32, 33, 36, 37, 40, 42, 45, 46, 65, 67, 100, 111, 112, 115, 118, 126, 139, 142, 146, 147, 161, 162, 186, 194, 210, 403	
<code>\GlsXtrUseAbbrStyleFmts</code> .....	218, 220, 227–229, 232, 233, 235, 237, 240, 249, 251, 254, 263, 265, 267, 268, 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, 231, 232, 240, 242, 254, 256, 271, 275, 276, 279	<code>\ifdefstring</code> .....	6, 23, 35, 181, 186, 402
<code>\glstruserfield</code> .....	283	<code>\ifdefvoid</code> 44, 47–50, 86, 106, 123, 127, 142, 143	
<code>\glstruserparen</code> .....	284–292	<code>\ifdim</code> .....	56, 57, 114, 384–392
<code>\glstrusersuffix</code> .....	284, 286, 288, 291	<code>\IfFileExists</code> 22, 128, 132, 133, 135, 363, 365	
<code>\glstruseseealsoformat</code> .....	46	<code>\ifglossaryexists</code> .....	44
<code>\glstruseseeformat</code> .....	45	<code>\ifglshasfield</code> .....	30, 283
<code>\GlsXtrWarnDeprecatedAbbrStyle</code> 193, 215		<code>\ifglshaslong</code> .....	105, 150
<code>\GlsXtrWarning</code> .....	54, 55	<code>\ifglshasparent</code> .	136, 138, 139, 142, 386–388
<code>\glstrword</code> .....	194	<code>\ifglshashashshort</code> .....	45, 60, 68
<code>\glstrwordsep</code> 194, 293, 295, 298, 301, 303, 304		<code>\ifglshassymbol</code> ....	189, 190, 380, 381, 383
<code>\glstrwrglossmark</code> .....	24	<code>\ifglshasindexonlyfirst</code> .....	84
		<code>\ifglshasgroupskip</code> .....	367, 369–377, 379, 380, 382, 394, 402
		<code>\ifglshasnonnumberlist</code> .....	60
		<code>\ifglshasnopostdot</code> .....	16, 119
		<code>\ifglssanitizesort</code> .....	117
		<code>\ifglssubentrycounter</code> .....	37, 38
		<code>\ifglshasused</code> .....	49, 50, 81, 84, 102, 110, 114, 210, 386, 387, 389, 390, 392
		<code>\ifglshasxindy</code> .....	128, 130
		<code>\ifglstr@hyperoutside</code> .....	66
		<code>\ifglstr@trinitwrglossbefore</code> ...	63, 66, 147
		<code>\ifglstr@insertinside</code> ..	203–210, 212, 213, 217, 219, 220, 222–231, 234, 236– 264, 266–282, 284–286, 288, 289, 291, 293, 295, 298, 300, 301, 304, 305, 307, 308
		<code>\ifHy@hyperindex</code> .....	180
		<code>\ifinlist</code> .....	98
		<code>\ifinlistcs</code> .....	31, 51
		<code>\ifinner</code> .....	25
		<code>\ifKV@glslink@hyper</code> .....	61, 64, 66
		<code>\ifKV@glslink@local</code> .....	62, 147
		<code>\ifKV@glslink@noindex</code> ....	8, 9, 12, 30, 83

## H

<code>\hangindent</code> .	380, 382, 383, 393–395, 399, 400
<code>\hbox</code> .....	366
<code>\hfill</code> .....	366
<code>\href</code> .....	86
<code>\hsize</code> .....	56, 57
<code>\hss</code> .....	366
<code>\hyperlink</code> .....	87
<code>\hyperpage</code> .....	180
<code>\hyperref</code> .....	86, 127, 330
<code>hyperref package</code> .....	87, 180, 309, 322

## I

<code>\if</code> .....	53
<code>\if@glstr@autoseeindex</code> ....	25, 26, 44, 48
<code>\if@glstr@format@override</code> .....	181
<code>\if@glstr@docdefrestricted</code> .....	51
<code>\if@glstr@indexcrossrefs</code> .....	15, 49
<code>\ifblank</code> .....	28, 54, 55, 115
<code>\ifcase</code> ..	7, 13, 20, 21, 24, 52, 63, 120, 378, 404
<code>\ifcsdef</code> ..	27, 38, 40–43, 63, 65, 80, 92–96, 106, 118, 124, 125, 137, 141, 144, 145, 151, 172–179, 189, 192, 210, 214, 369–376
<code>\ifcsstring</code> .....	28, 168, 213
<code>\ifcsundef</code> ....	33, 36, 38, 40–42, 51, 56, 58, 88, 100, 106–110, 124, 128, 141, 151, 152, 168, 213–216, 366, 384, 385, 393, 406
<code>\ifcsvoid</code> .....	49, 167



<code>\ifmmode</code> .....	25	<code>\levelchar</code> .....	184
<code>\ifnum</code> .....	15, 33, 50, 102, 110, 111, 123, 133, 146, 380, 382, 393, 394	<code>\listadd</code> .....	106
<code>\ifstrempy</code> .....	137, 144, 152	<code>\listbreak</code> .....	186
<code>\ifstrequal</code> .....	16, 17, 21, 22	<code>\listcsadd</code> .....	31
<code>\ifthenelse</code> .....	131, 132, 189	<code>\listcseadd</code> .....	31, 107
<code>\IfTrackedDialectHasMapping</code> .....	36	<code>\listcsgadd</code> .....	31, 51
<code>\IfTrackedLanguageFileExists</code> .....	329	<code>\listcsxadd</code> .....	31, 107
<code>\ifundef</code> ....	23, 32, 33, 36, 115, 186, 187, 363	<code>\listxadd</code> .....	97, 98
<code>\ifx</code> .....	8–11, 46, 56, 57, 65, 114, 115, 118, 119, 121, 126, 127, 135, 181, 182, 184, 192–194, 196, 292, 293, 400	<code>\loadglsentries</code> .....	52, 129
<code>\immediate</code> .....	101, 102, 110, 128, 135	<code>\long</code> .....	39
<code>\index</code> .....	181	<b>M</b>	
<code>\indexspace</code> .	367, 379–383, 394–400, 402, 405	<code>\MakeAcronymsAbbreviations</code> .....	113
<code>\input</code> .....	328	<code>\makeatletter</code> .....	51, 128, 133, 183
<code>\inputencodingname</code> .....	135	<code>\makeatother</code> .....	183
<code>\InputIfFileExists</code> .....	51	<code>\makebox</code> .....	366, 393, 394
<code>\istfilename</code> .....	115	<code>\makefirstuc</code> .....	186
<code>\item</code> ....	130, 131, 366–369, 378, 379, 395, 396	<code>makeglossaries</code> .....	121
<b>J</b>		<code>\makeglossaries</code> .....	114, 129–132, 135
<code>\jobname</code> .....	51, 128, 130–133, 135	<code>\makeglossary</code> .....	115
<b>K</b>		<code>makeindex</code> .....	407
<code>\key@ifundefined</code> .	12, 13, 28, 29, 79, 139, 142	<code>makeindex</code> .....	14, 114
<code>\KV@glslink@hyperfalse</code> ..	68, 81, 82, 87, 88	<code>\makenoidxglossaries</code> .....	130
<code>\KV@glslink@hypertrue</code> .....	87	<code>\MakeTextUppercase</code> .....	311
<code>\KV@glslink@noindexfalse</code> .....	82	<code>\MakeUppercase</code> .....	311, 312
<code>\KV@glslink@noindextrue</code> .....	82, 88	<code>\marginpar</code> .....	25
<b>L</b>		<code>\markboth</code> .....	310
<code>\L</code> .....	345, 348	<code>\markright</code> .....	310
<code>\l</code> .....	348	<code>\mathit</code> .....	333
<code>\label</code> .....	23	<code>\mathrm</code> .....	332–334
<code>\LaTeX</code> .....	130, 131	<code>\maxdimen</code> .....	56, 57
<code>\leaders</code> .....	366	<code>\mbox</code> .....	368, 369, 393
<code>\leavevmode</code> .....	40, 64	<code>\medskip</code> .....	131, 132, 141
<code>\let</code> .....	5, 7– 14, 16, 18, 19, 25–27, 30, 32, 35, 36, 39, 51, 52, 56, 58, 59, 61, 62, 64–85, 87, 88, 92, 97, 98, 100, 101, 109–116, 118–125, 133, 135, 139, 140, 142, 147, 152, 161, 172–175, 177–182, 186, 187, 191–196, 199–210, 212–214, 223, 246, 260, 282, 309–313, 363, 378, 383, 385, 396, 403–406	<code>\MessageBreak</code> .....	52, 55, 102, 111, 115, 117, 118, 132, 213
<code>\letabbreviationstyle</code> ..	223, 225, 226, 228, 231, 232, 238, 239, 252, 254, 270, 271	<code>mfirstuc package</code> .....	186
<code>\letcs</code> .....	28, 32, 33, 45, 50, 63, 65, 79, 122–124, 141, 142, 172–179, 406	<code>\mfirstucMakeUppercase</code> .....	68–79, 81, 89–91, 93, 95, 96, 104, 105, 113, 150, 153–160, 163–166, 175, 176, 179, 201, 202, 204, 206, 208, 210–212
		<code>\mfu@checkword@arg</code> .....	186
		<code>\mfu@checkword@do</code> .....	186
<b>N</b>		<b>N</b>	
		<code>\NeedsTeXFormat</code> .....	5, 329, 365, 401
		<code>\new@glossaryentry</code> .....	52, 117
		<code>\new@ifnextchar</code> .....	29, 80, 81, 104, 105, 144, 147–149, 191, 199–209
		<code>\newabbr</code> .....	18, 19
		<code>\newabbreviation</code> .....	18, 19

<code>\newabbreviationhook</code> .....	196	<code>text</code> ....	86, 153, 163, 216, 218, 315, 316, 324
<code>\newabbreviationstyle</code> .....	216, 218–	<code>textaccess</code> .....	162
	221, 223, 225–238, 240, 242, 244, 245,	<code>user2</code> .....	37
	248–252, 254, 256, 258, 259, 262–272,	<code>\newglossarystyle</code> .....	403
	274–277, 279, 281, 284, 285, 287, 289,	<code>\newif</code> .....	63, 180, 216
	290, 292–295, 297, 299–302, 304, 306–308	<code>\newlength</code> .....	383, 384
<code>\newacronym</code> .....	112, 113	<code>\newnum</code> .....	19
<code>\newacronymhook</code> .....	112	<code>\newrobustcmd</code> .....	29,
<code>\newacronymstyle</code> .....	113, 114		30, 32, 34, 35, 46, 47, 64, 67, 80, 81, 91–
<code>\newcommand</code> .....			93, 104, 105, 119, 124, 136, 137, 140,
... 5–13, 15–33, 35, 37–42, 44–46, 48–			141, 144, 145, 147–149, 179, 186, 187,
50, 52–56, 58–60, 63, 64, 66–68, 79–85,			199–210, 292, 310, 311, 313–322, 385–392
87, 88, 91, 93–100, 102, 104–108, 110,		<code>\newsym</code> .....	19
111, 113, 114, 118–124, 126, 127, 129–		<code>\newterm</code> .....	170
161, 163–172, 177, 178, 180–194, 197–		<code>\newtoks</code> .....	193
210, 212–216, 218, 220, 221, 225, 227,		<code>\newwrite</code> .....	115
230, 232, 233, 247, 261, 262, 283, 284,		<code>\nobreak</code> ....	368, 369, 379, 381–383, 395–398
286, 289, 293, 295, 298, 301, 303, 306–		<code>\NoCaseChange</code> .....	93–96, 137, 313–321
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			133, 139, 140, 142, 151, 182, 196, 365, 404
<code>\newcount</code> .....	133, 143	<code>\nofiles</code> .....	131
<code>\newcounter</code> .....	23, 151	<code>\noindent</code> ...	131, 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, 196		<code>\ns@Glsxtrfull</code> .....	200
<code>\newglossaryentry options</code>		<code>\ns@glxtrfull</code> .....	199
<code>access</code> .....	162	<code>\ns@GLSxtrfullpl</code> .....	202
<code>alias</code> .....	16, 44, 47–50	<code>\ns@Glsxtrfullpl</code> .....	201
<code>desc</code> .....	157, 164, 165	<code>\ns@glxtrfullpl</code> .....	201
<code>descplural</code> .....	157, 165	<code>\ns@GLSxtrlong</code> .....	206
<code>first</code> .....	86,	<code>\ns@Glsxtrlong</code> .....	205
	154, 155, 163, 164, 216, 317, 318, 325, 407	<code>\ns@glxtrlong</code> .....	204
<code>firstaccess</code> .....	162	<code>\ns@GLSxtrlongpl</code> .....	209
<code>firstplural</code> .....	155, 164, 216, 318, 325, 407	<code>\ns@Glsxtrlongpl</code> .....	209
<code>group</code> .....	141, 142	<code>\ns@glxtrlongpl</code> .....	208
<code>loclist</code> .....	31	<code>\ns@GLSxtrshort</code> .....	204
<code>long</code> .....	159, 166, 326	<code>\ns@Glsxtrshort</code> .....	203
<code>longplural</code> .....	160, 166, 327	<code>\ns@glxtrshort</code> .....	202, 203
<code>name</code> ...	45, 152, 153, 163, 181, 314, 315, 323	<code>\ns@GLSxtrshortpl</code> .....	207
<code>plural</code> .....	154, 163, 216, 316, 317, 324	<code>\ns@Glsxtrshortpl</code> .....	207
<code>see</code> .....	15, 16, 26, 44, 45, 47, 50, 52, 115	<code>\ns@glxtrshortpl</code> .....	206
<code>seealso</code> .....	16, 44, 45, 47, 49, 50, 418	<code>\null</code> .....	21
<code>short</code> .....	158, 165, 193	<code>\number</code> .....	51, 107, 108, 110, 133, 144
<code>shortaccess</code> .....	161, 162	<code>\numexpr</code> .....	107, 110
<code>shortaccessplural</code> .....	161		
<code>shortplural</code> .....	159, 165, 193		
<code>symbol</code> .....	155, 156, 164		
<code>symbolplural</code> .....	156, 164		

## O

<code>\O</code> .....	345, 348
<code>\o</code> .....	348

\or	7, 13, 14, 20, 21, 24, 52, 63, 378, 404	xindy	46, 47
\org@glossaryentrynumbers	57, 119	\PackageError	6, 11, 22, 26, 52, 55, 56, 63, 80, 81, 83, 92, 93, 100, 101, 108, 110, 111, 113, 115, 117, 125, 135, 140, 141, 143, 189, 213–216, 365, 366
\org@glossarytitle	118, 119	\PackageWarning	16
\org@ifKV@glslink@hyper	64, 66	\PackageWarningNoLine	16
<b>P</b>			
\p@gl@hyp@opt	85	\pageref	23, 37, 38
package options:		\par	131, 132, 368, 369, 378, 380–383, 393–400, 402
abbreviations	17, 18	\parindent	378, 380, 382–384, 393–400, 403
accsupp	21, 152	\parskip	378, 380, 382, 396–398, 403
acronym	17	\PassOptionsToPackage	5, 22
automake	118, 130, 135	\pdfbookmark	402
true	135	\preglossarypreamble	38
autoseeindex	26	\preto	82
false	25	\print@noop@unsrtglossaryunit	12, 13
counter		\print@op@unsrtglossaryunit	14
wrglossary	23	\printabbreviations	17
debug		\printglossaries	116, 130
showtargets	87	\printglossary	17, 116, 130–132
docdef	14, 15, 51, 52, 100, 108	\printglossary options	
false	51, 52	nonumberlist	59
restricted	15	type	118
true	50, 52	\printnoidxglossaries	130
docdefs		\printnoidxglossary	116, 130
restricted	51	\printnumbers	19, 171
indexcounter	330	\printsymbols	19, 170
nonumberlist	57	\printunsrtglossary	131, 132, 138
nopostdot	16	\printunsrtglossaryentryprocesshook	139, 140
false	16	\printunsrtglossaryhandler	140
numbers	19	\printunsrtglossarypredoglossary	139
postdot	16	\printunsrtglossaryskipentry	139
record	7, 13, 51, 61, 114, 132, 133, 199, 201–210, 416	\printunsrtglossaryunit	13, 14, 141
alsoindex	8, 10	\printunsrtglossaryunitsetup	140
only	8, 131	\ProcessOptions	366
shortcuts	20	\ProcessOptionsX	24
ac	20	\protect	93– 96, 163–166, 194, 197, 198, 216–223, 225–227, 229–238, 240–246, 248–260, 262–282, 284, 285, 287, 289–292, 294– 297, 299, 300, 302–304, 306–309, 313–321
all	20	\protected@csedef	33, 34, 124, 384, 385
false	20	\protected@csxdef	34, 124, 384, 385
none	20	\protected@edef	35, 56, 64, 85, 112, 123, 124, 140–142, 181, 196
true	20	\protected@write	11, 12, 51, 59, 115, 116, 133–136, 406
sort			
use	67		
style	22		
stylemods	22		
symbols	19, 170		
undefaction	43		
error	6		
warn	6		

<code>\providecommand</code> .....		<b>S</b>	
..... 17–19, 28, 46, 59, 81, 82, 101, 110, 115, 128, 134, 330, 332–334, 366, 401		<code>\s@glstrfmt</code> .....	29
<code>\ProvidesFile</code> .....	328	<code>\s@glshyp@opt</code> .....	84
<code>\ProvidesPackage</code> .....	5, 329, 365, 401	<code>\s@glstr@enabletagging</code> .....	185
		<code>\s@glstrfmt</code> .....	29
<b>Q</b>		<code>\s@glstrifhasfield</code> .....	32
<code>\quotechar</code> .....	184	<code>\s@GlsXtrStartUnsetBuffering</code> .....	97
		<code>\s@GlsXtrStopUnsetBuffering</code> .....	98
<b>R</b>		<code>\s@printunrtglossary</code> .....	138, 140
<code>\raggedright</code> .....	371, 372, 375, 376, 403	<code>\sealsoname</code> .....	46, 47
<code>\refstepcounter</code> .....	23	<code>\seename</code> .....	45
<code>\relax</code> .....	7, 13–15, 18– 21, 24, 26, 30, 33, 50–52, 56, 57, 59, 62– 65, 84, 92, 101, 102, 110, 115, 116, 118, 119, 122, 123, 126, 133, 135, 142, 146, 182, 184, 186, 189, 193–195, 292, 293, 378, 380, 382, 385–395, 399, 400, 403–406	<code>\setabbreviationstyle</code> .....	113, 218, 226
<code>relsize package</code> .....	247	<code>\setacronymstyle</code> .....	113, 114
<code>\renewcommand</code> .....	6, 7, 13–17, 20–24, 26, 39–45, 50–53, 55–62, 79, 81–87, 91, 98–101, 105, 108–121, 123–125, 127– 129, 140, 141, 146, 170, 172–177, 185– 188, 198, 214–282, 284–292, 294–297, 299–310, 363, 366–383, 393–400, 403–405	<code>\setentrycounter</code> .....	125
<code>\renewenvironment</code> .....	367, 369– 376, 378, 380, 382, 393, 396–398, 400, 403	<code>\SetGenericNewAcronym</code> .....	114
<code>\renewglossarystyle</code> .....	366–376, 378–382, 393–400	<code>\setglossarystyle</code> .....	22, 118, 366, 368, 369, 379, 381, 382, 394–400, 403
<code>\renewrobustcmd</code> .....	67, 87	<code>\setkeys</code> .....	9, 22, 26, 30, 65, 67, 83, 112, 119, 147, 195, 196
<code>\RequireGlossariesExtraLang</code> ...	329, 363	<code>\setlength</code> .....	56, 57, 378, 380, 382, 383, 394, 396–398, 403
<code>\RequirePackage</code> ...	5, 14, 21, 22, 24, 365, 401	<code>\settowidth</code> .....	114, 384–393
<code>\reserved@a</code> .....	191	<code>\setupglossaries</code> .....	5, 26
<code>\reserved@b</code> .....	191	<code>\sfcode</code> .....	16, 17, 189, 377
<code>\reserved@d</code> .....	192	<code>\small</code> .....	25
<code>\RestoreAcronyms</code> .....	113	<code>soul package</code> .....	97
<code>\rGLS</code> .....	146	<code>\space</code> 6, 11, 46, 52, 53, 55, 83, 100–102, 108, 110, 111, 113–118, 129, 131, 132, 135, 140, 141, 143, 189, 190, 194, 198, 218, 366, 367, 377, 380, 381, 383, 384, 393, 401	
<code>\rGls</code> .....	146	<code>\spacefactor</code> .....	16, 17, 189, 195, 377
<code>\rgls</code> .....	146	<code>\stepcounter</code> .....	151
<code>\rGLSformat</code> .....	149	<code>\string</code> .....	6, 11, 12, 46, 47, 51–53, 55, 59, 65, 80, 81, 83, 92, 93, 100–102, 108, 110, 111, 113, 115– 118, 128–136, 140, 141, 143, 174, 175, 177, 179, 181, 189, 329, 330, 334–363, 406
<code>\rGlsformat</code> .....	148	<code>\strut</code> .....	366–376, 382
<code>\rglsformat</code> .....	148, 150	<code>\subglossentry</code> .....	119, 142, 366–376, 378, 380, 382, 394, 404
<code>\rGLSpl</code> .....	146	<code>\subitem</code> .....	378
<code>\rGlspl</code> .....	146	<code>\subsubitem</code> .....	378
<code>\rglspl</code> .....	146		
<code>\rGLSplformat</code> .....	150	<b>T</b>	
<code>\rGlsplformat</code> .....	149	<code>\tablehead</code> .....	373–376
<code>\rglsplformat</code> .....	148, 150	<code>\tabletail</code> .....	373–376
<code>\romannumeral</code> .....	384, 385, 393	<code>\tabularnewline</code> .....	369–377
		<code>\TeX</code> .....	130
		<code>\texorpdfstring</code> ....	30, 93–95, 311, 322–328

